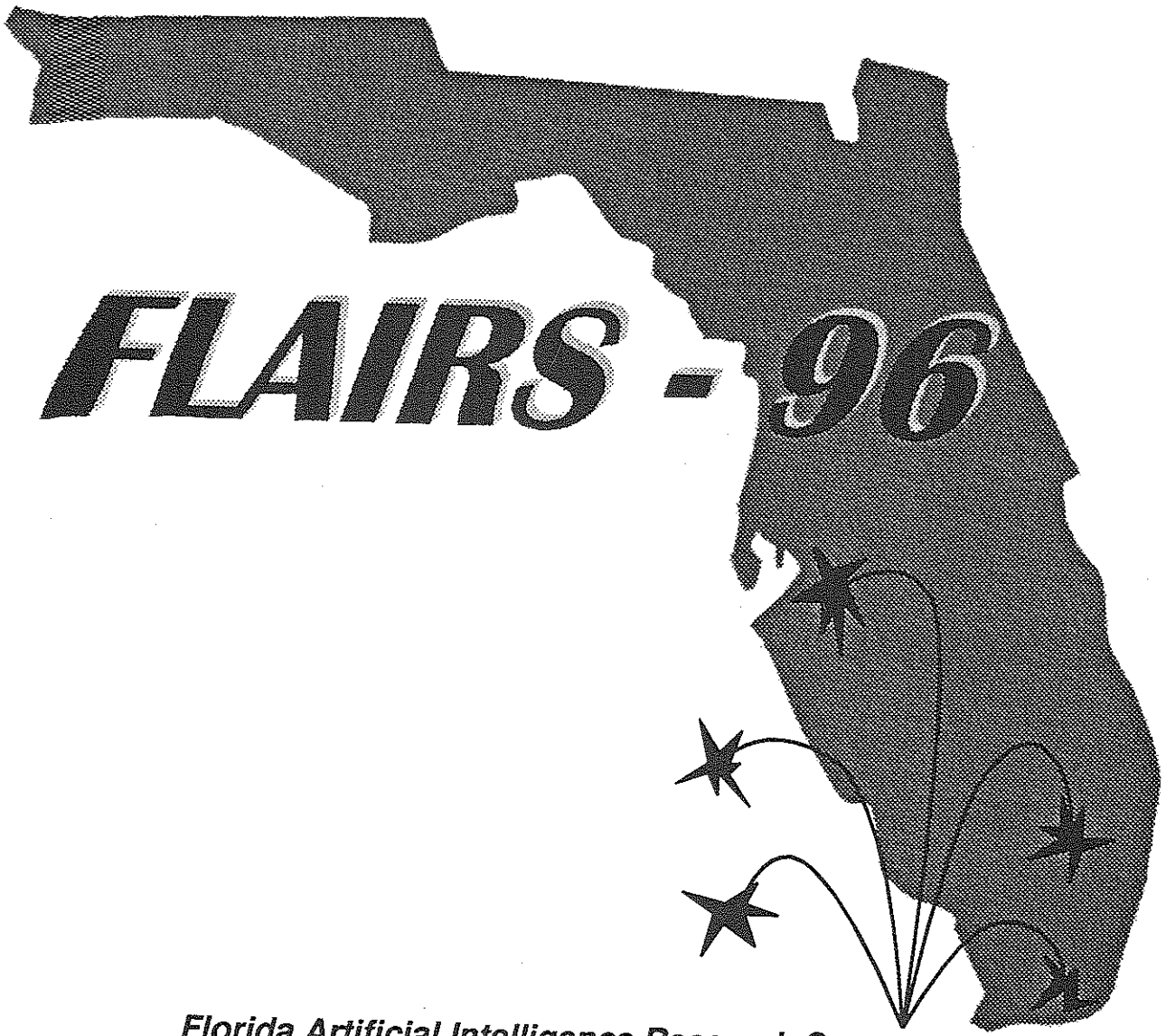
Proceedings of the Ninth Florida Artificial Intelligence Research Symposium



Key West, Florida
May 20 - 22, 1996

John H. Stewman
Editor

Proceedings

The Ninth Florida Artificial Intelligence Research Symposium

FLAIRS-96

May 20 - 22, 1996

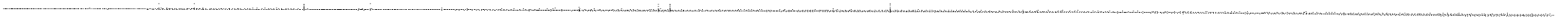
Key West, Florida

ISBN 0-9620-1738-8

Additional copies of this conference proceedings or any of the previous proceedings can be ordered from:

John H. Stewman
Eckerd College
P.O.Box 12560
St. Petersburg, FL 33733

Price: $35 + S&H



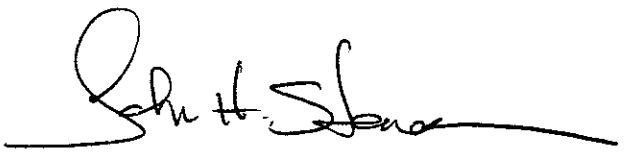
Message from the Proceedings Editor

Welcome to FLAIRS-96, our ninth annual symposium on artificial intelligence. This promises to be an exciting gathering. We have numerous sessions dealing with leading-edge AI research and ground-breaking applications of AI in areas as diverse as constraint-based reasoning, planning, autonomous agents, vision, learning, tutoring, expert systems, knowledge acquisition and representation, logic and optimization, knowledge-based systems, natural language processing, neural networks, information retrieval, and software engineering. Eight special topic tracks have been organized by their own committees and cover real world natural language understanding, real-time planning and reacting, information interchange, uncertain reasoning, reasoning about function, expert systems applications for agriculture, controlling search in automated reasoning systems, and AI in education. We also have a number of distinguished invited speakers who promise stimulating talks on a diverse set of topics.

The Florida AI Research Society was organized in 1987 to promote AI research and to foster collaboration between AI researchers and developers in Florida academic institutions, government and industry. Each year since 1987 we have held a research conference — the Florida AI Research Symposium or FLAIRS. In nine years, these symposia have grown from Florida only events into successful international gatherings of AI researchers. No doubt, our sunny Florida climate helps, but our growth has also been accompanied by a steady increase in the quality of the research presented. This year we have researchers attending from Italy, France, Great Britain, Germany, Australia, New Zealand, Canada, and Mexico to name a few. There are also many in attendance from all over the United States. I welcome you all and hope you enjoy and profit from your attendance of FLAIRS-96.

I would like to thank all members of the organizing committee for their many efforts throughout the year. Without them this conference would not be a reality. I would also like to thank all referees for their willingness to share their experience and expertise and for their participation in the selection of an outstanding program. Most of all I would like to thank the invited speakers, the panelists and the authors of the many fine papers to be presented over the next few days. The real credit for the quality of this symposium goes to them.

As always, the officers of FLAIRS and the organizers for this and future symposia, welcome your feedback on the conference, it's structure, events, and invited speakers. Your views help us improve the quality of the event from year to year. Also, if you are personally interested in becoming more involved in the planning or running of any FLAIRS event in the future, please let me know.



John H. Stewman
Program Chair & Proceedings Editor

FLAIRS-96

Florida AI Research Society Officers

President
John H. Stewman, Eckerd College

Vice President
Massood Towhidnejad, Embry-Riddle Aeronautical University

Secretary
Robert A. Morris, Florida Institute of Technology

Treasurer
Avelino Gonzalez, University of Central Florida

Symposium Organizing Committee

General Co-Chairs
Douglas D. Dankel II, University of Florida
Michael J. Prietula, University of Florida

Program Committee Chair & Proceedings Editor
John H. Stewman, Eckerd College

Tracks Coordinator & Reasoning About Function Track Co-Chair
Amruth Kumar, Ramapo College of New Jersey

Reasoning About Function Track & TIME-96 Workshop Program Co-Chair
Luca Chittaro, Universita' di Udine, Italy

Real-World Natural Language Understanding Track Co-Chairs
Ashwin Ram, Georgia Institute of Technology
Kenneth Moorman, Georgia Institute of Technology

AI Education Track Chair
Deepak Kumar, Bryn Mawr College

Real-Time Planning and Reacting Track Chair
Rhonda Eller-Meshreki, Randolph-Macon College

Information Interchange Track Co-Chairs
Susan Haller, University of Wisconsin - Parkside
Syed S. Ali, Southwest Missouri State University

Controlling Search in Automated Reasoning Systems Track Chair
B. Spencer, U. New Brunswick

Uncertainty Track Chair
Eric Neufeld

Expert System Applications in Agriculture Track Chair
Fedro S. Zazueta, University of Florida

Workshops Coordinator & TIME-96 Workshop Co-Chair
Howard Hamilton, University of Regina

TIME-96 Workshop Co-Chair
Scott Goodwin, University of Regina

TIME-96 Workshop Program Co-Chair
Angelo Montanari, Universita' di Udine, Italy

CONSTRAINT-96 Workshop Co-Chairs
David Leasure, Texas A&M University at Corpus Christi
Martina Shollmeyer, Texas A&M University at Corpus Christi

ROBOLEARN-96 Workshop Co-Chairs
Henry Hexmoor, SUNY Buffalo
Lisa Meeden, Swarthmore College

Symposium Sponsor
Florida AI Research Society

FLAIRS-96 T-Shirt Design
Mario A. Pennycooke, University of Florida

Symposium Program Committee

Syed S. Ali, Southwest Missouri State University
Kevin W Bowyer, University of South Florida
Alberto J. Canas, University of West Florida
Douglas D. Dankel II, University of Florida
Cary deBessonet, Louisiana State Law Institute
David Eggert, University of Edinburgh
Mark Fishman, Eckerd College
Ken Ford, University of West Florida
Janice Glasgow, Queen's University
Clark Glymour, Carnegie Mellon
Avelino J. Gonzalez, University of Central Florida
Scott Goodwin, University of Regina
Nicolas Groleau, NASA/Recom Technologies
Larry Hall, University of South Florida
Susan Haller, University of Wisconsin - Parkside
Howard J. Hamilton, University of Regina
Mehdi T. Harandi, University of Illinois at Champaign-Urbana
Patrick Hayes, University of Illinois
Rattikorn Hewett, Florida Atlantic University
Fred Hoffman, Florida Atlantic University
Ken Hughes, University of the Pacific
Michael N. Huhns, MCC
Ibrahim F. Imam, SRA
Rainer Knauf, Technical University of Ilmenau
Amruth Kumar, Ramapo College of New Jersey
Henry E. Kyburg, Jr., University of Rochester
David B. Leake, Indiana University
David E. Leasure, Texas A&M University - Corpus Christi
Rasiah Logananthara j, University of Southwestern Louisiana
Robert A. Morris, Florida Institute of Technology
Robert Plant, University of Miami
Michael J. Prietula, University of Florida
Abdul Sattar, Griffith University
Ethan Scarl, Boeing
Peter G. Selfridge, AT&T Bell Laboratories
Dorothy E. Setliff, University of Pittsburgh
Valerie Shalin, University of Buffalo
Evangelos Simoudis, IBM
Louise Stark, University of the Pacific
Dan E. Tamir, Florida Institute of Technology
Massood Towhidnejad, Embry-Riddle Aeronautical University
Steven Walczak, University of South Florida
Ronald R. Yager, Iona College
Neli P. Zlatareva, Central Connecticut State University

Real-World Natural Language Understanding Track Committee

Ashwin Ram, Georgia Institute of Technology
Kenneth Moorman, Georgia Institute of Technology
Eugene Charniak, Brown University
Scott Huffman, Price Waterhouse Technology Center
Yael Ravin, T.J. Watson Research Center
Chris Riesbeck, Northwestern University
Stuart C. Shapiro, SUNY Buffalo

Reasoning About Function Track Committee

Amruth Kumar, Ramapo State College
Luca Chittaro, Universita' di Udine, Italy
Ameen Abu-Hanna, Utrecht University, Netherlands
Dean Allemang, PTT Telecom, Switzerland
Kevin Bowyer, University of South Florida
Matthew Brand, M.I.T.
B. Chandrasekaran, Ohio State University
Jack Hodges, San Francisco State University
Yumi Iwasaki, Stanford University
Morten Lind, Technical University of Denmark, Denmark
James McDowell, Michigan State University
Chris Price, University of Wales, U.K.
Yasushi Umeda, University of Tokyo, Japan

Artificial Intelligence Education Track Committee

Deepak Kumar, Bryn Mawr College
Robert Aiken, Temple University
Giorgio Ingargiola, Temple University

Real-Time Planning and Reacting Track Committee

Rhonda Eller-Meshreki, Randolph-Macon College
David Musliner, Honeywell Technology Center
Marcel Schoppers, Stanford University
Jennifer Chu-Carroll, University of Delaware
Linda Suri, Central Institute for the Deaf, St. Louis MO
Lynn Lambert, Christopher Newport University

Information Interchange Track Committee

Susan Haller, University of Wisconsin - Parkside
Syed S. Ali, Southwest Missouri State University
Alistair Campbell, State University of New York at Buffalo
Hans Chalupsky, Information Sciences Institute

Controlling Search in Automated Reasoning Systems Track Committee

B. Spencer, University of New Brunswick
O. Astrachan, Duke University, North Carolina
W. Bibel, Technische Hochschule Darmstadt, Germany
J. Dix, Universitaet Koblenz-Landau, Germany
U. Furbach Universitaet Koblenz-Landau, Germany
C. Goller, Universitaet Muenchen, Germany
J. Lobo, University of Illinois, Chicago
R. Letz, Universitaet Muenchen, Germany
K. Mayr, Universitaet Muenchen, Germany
W. McCune, Argonne National Labs, Argonne, Illinois
E. Neufeld, University of Saskatchewan, Canada
D. Reed, Dickinson College, Pennsylvania
D. Seipel, Universitaet TueBingen, Germany
M. Stickel, Stanford Research Institute
G. Sutcliffe, James Cook University
C. Suttner, Universitaet Muenchen, Germany

Uncertainty Track Committee

Eric Neufeld, University of Saskatchewan

Expert System Applications in Agriculture Track Committee

Fedro S. Zazueta, University of Florida
Bob Peart, University of Florida
Poliotro Martinez, IMTA Mexico
Eduardo Holzapfel, Universidad de Concepcion, Chile

The Symposium's Distinguished Invited Speakers

Welcome Talk — Monday, 8:10 - 8:35 a.m.

Speaker TBA

Opening Plenary Talk — Monday, 8:40 - 9:20 a.m.

Computational Social Science: An Opportunity for AI *Michael Fehling, Stanford university*

Increasingly, social scientists rely upon computational methods as a core component of their theoretical and empirical work. The appearance of a new archival journal, "Computational and Mathematical Organization Theory," suggests that Computational Social Science has now become a distinct, interdisciplinary field. This field presents exciting challenges and opportunities for members of the AI and computer-science research communities. In particular, much of the theory and research in Computational Social Science draws extensively from AI for both concepts and modeling techniques.

My talk offers an overview of this emerging interdisciplinary field and a discussion of opportunities presented to computer scientists. Some critical reflexctions upon history help to shape my remarks: AI's role in the evolution of cognitive science offers some useful lessons for those who would contribute to the new field of Computational Social Science.

Plenary Talk — Monday, 9:20 - 10:00 a.m.

Causation: What Artificial Intelligence Has Done for Us Lately
Clark Glymour, Carnegie Mellon University

For most of this century many statisticians have claimed that no causal inferences can be reliably made from uncontrolled, non-experimental data. But just such inferences are responsible for the discovery of the effects of smoking, the discovery of AIDS, and the discovery of the effects of low concentrations of lead. Skeptics may say that these discoveries are rare examples of good fortune amidst a larger background of unnoticed erroneous inferences.

In the last decade, the computer science community has developed a number of rigorous, feasible, demonstrably reliable methods for causal inference from uncontrolled data. This lecture will describe some recent results of applying those methods, including 1) identifying components of psychometric tests, 2) finding the causes of college dropout rates, 3) recalibrating instruments on orbiting satellites, and 4) reanalyzing the effects of lead on children's IQ.

Banquet Talk — Tuesday Luncheon

The Logic of Enterprise Models
Mark Fox, University of Toronto

As information systems play a more active role in the management and operations of an enterprise, the demands on these systems have also increased. Departing from their traditional role as simple repositories of data, information systems must now provide more sophisticated support to manual and automated decision making; they must not only answer queries with what is explicitly represented in their Enterprise Model, but must be able to answer queries with what is implied by the model. The goal of the TOVE Enterprise Modeling project is to create the next generation Enterprise Model, a Common Sense Enterprise Model. By common sense we mean that an Enterprise Model has the ability to deduce answers to queries that require relatively shallow knowledge of the domain.

Keynote Address — Wednesday, 8:00 - 8:40 a.m.

What Case-Based Reasoning Really Means: Now and the Future

David Leake, Indiana University

Case-based reasoning (CBR) is now an established artificial intelligence paradigm, but there is still confusion about what case-based reasoning really means. This talk delineates the fundamental tenets of case-based reasoning, identifying key differences between CBR and other paradigms. It then shows how those differences reveal opportunities for CBR itself and for synergy between CBR and other artificial intelligence methods. It examines how the opportunities suggest promising new roles for case-based reasoning systems and for the application of CBR principles. The talk identifies challenges, such as the case adaptation problem, that must be addressed in order to realize the full potential of CBR. It describes ongoing research that addresses those challenges to move towards a new generation of case-based reasoning systems.

Closing Plenary Talk — Wednesday, 11:20 a.m. - Noon

Symbolophobia

Geoffrey LaForte, University of West Florida
Kenneth M. Ford, University of West Florida
Patrick J. Hayes, Illinois University

Computers are the only machines which both manipulate symbols and are manipulated by them; they are "physical symbol systems." As we know, AI uses this insight to implement systems capable of thought. However, some people find this idea of a symbolic machine very unsettling. This tension can be a clinical precursor of a disorder called symbolophobia. Those suffering from symbolophobia take the word "symbol" to be symbolic of all that is wrong with AI. Symbols are blamed for rigidity, ungroundedness, disembodiment, formality, seriality, logicality, etc. Victims of symbolophobia often find reassurance by imagining computers that are subsymbolic, analog, biological, socially-situated, or quantum. In this paper we will examine symbolophobia from both scientific and clinical perspectives, outlining several syndromes indicative of (but not exclusive to) its initial and advanced stages. No full cure has yet been found, but we will suggest some promising treatment protocols. While the etiology and epidemiology of this worrisome affliction are not fully understood, we will offer some speculations for further investigation.

Table of Contents

Session II A – Special Track on Real-World Natural Language Understanding (II)
Monday, 1:20 - 3:00 p.m.

Session II B – Special Track on Real-time Planning and Reacting
Monday, 1:20 - 3:00 p.m.

Session II C – Reasoning
Monday, 1:20 - 3:00 p.m.

Session III A – Multiple Agents, Vision and Genetics
Monday, 3:20 - 4:20 p.m.

Session III B – Learning (I)
Monday, 3:20 - 4:20 p.m.

Session III C – Tutoring and Expert Systems
Monday, 3:20 - 4:20 p.m.

Session IV A – Special Track on Information Interchange (I)
Tuesday, 8:00 - 9:20 a.m.

Session IV B – Learning (II)
Tuesday, 8:00 - 9:20 a.m.

Session IV C – Logic and Optimization
Tuesday, 8:00 - 9:20 a.m.

Session V A – Knowledge Acquisition and Representation
Tuesday, 9:40 - 11:20 a.m.

Session V B – Knowledge-based Systems
Tuesday, 9:40 - 11:20 a.m.

Session V C – Special Track on Uncertain Reasoning
Tuesday, 9:40 - 11:20 a.m.

Session VI A – Special Track on Information Interchange (II)
Tuesday, 1:20 - 2:40 a.m.

Session VI B – Natural Language Processing (I)
Tuesday, 1:20 - 2:40 p.m.

Session VI C – Special Track on Reasoning about Function (I)
Tuesday, 1:20 - 2:40 p.m.

Session VII A – Information Retrieval and Data Mining
Tuesday, 3:00 - 4:20 p.m.

Session VII B – Special Track on Expert Systems Applications for Agriculture
Tuesday, 3:00 - 4:40 p.m.

Session VII C – Special Track on Reasoning about Function (II)
Tuesday, 3:00 - 4:20 p.m.

Session VIII A – Neural Networks and Vision
Wednesday, 8:50 - 9:50 a.m.

Session VIII B – Special Track on Controlling Search in Automated Reasoning Systems
Wednesday, 8:50 - 9:50 a.m.

Session VIII C – Special Track on AI Education (I)
Wednesday, 8:50 - 9:50 a.m.

Session IX A – Software Engineering
Wednesday, 10:10 a.m. - 11:10 a.m.

Session IX B – Special Track on AI Education (II)
Wednesday, 10:10 a.m. - 11:10 a.m.

Session IX C – Natural Language Processing (II)
Wednesday, 10:10 - 11:10 a.m.

AGGREGATION AS A SUBTASK OF TEXT AND SENTENCE PLANNING

Hercules Dalianis
Department of Computer and Systems Sciences
The Royal Institute of Technology and
Stockholm University
hercules@dsv.su.se

Abstract

Natural language generation is the technique of letting a computer automatically create natural language, e.g. English, Chinese or Greek, out of a computational representation. To generate natural language from computational representations, a number of processes must be carried out. Part of the process called sentence planning is the task of aggregation. Aggregation is the process which removes redundancies during generation of a natural language discourse without losing any information. Aggregation, which has been called ellipsis or coordination in Linguistics, makes text more fluent and easily read. While people do aggregation all the time without thinking about it, the contents of software engineering tools, data bases and expert systems, etc., is often highly redundant and needs aggregation before paraphrased to natural language.

This paper summarizes a larger work [Dalia96] which address various aspects of aggregation. When do we need to carry out aggregation ? What type of aggregations are there? Are there any general rules for how to aggregate? How are the rules related to each other? Aggregation may give rise to ambiguities: How can we solve them? How is aggregation related to the other generation processes?

1. INTRODUCTION

Aggregation, which is a subtask of Text and Sentence planning in Natural Language Generation (NLG), has received very little attention to date. We define aggregation to be the process that removes redundancies during generation of a natural language discourse without (ideally) losing any information. While people do aggregation all the time without thinking about it, the contents of software engineering tools, data bases and expert systems, etc., are often highly redundant, and therefore need aggregation in order to deliver high quality natural language. In this paper we develop the concept of aggregation and describe how and when it should be done.

2. WHAT IS AGGREGATION?

Aggregation[1] is the process of removing redundant information in a text without, losing any information. People do aggregation all the time to make natural language expressions shorter, non-redundant and easy to read.

For example

John has a book (a)

Mary has a book (b)

aggregation =>

John and Mary have a book (c)

We can see in the above example that in the two sentences (a) and (b) the objects which are different are, i.e. *John* and *Mary* , the rest of the sentences (a) and (b), are the same, i.e. *has a book* . Therefore we can aggregate the parts which are the same into one unit and then use the coordinator *and* between *John* and *Mary* and we obtain sentence (c).

In newspapers, books, articles you find various types of aggregation. For example the ratio (syntactic aggregation cases)/(total sentences) is approximately 33 % i.e. one third of the sentences has syntactic aggregation. The aggregation makes texts 10-20% shorter than it should have been without grouping as well as it is easier to read

We have studied aggregation from a "generation" view where we "generate ellipsis". The term ellipsis originates from the Greek word *ellípsis*, meaning missing or omission. We are using the term ellipsis in its more original general form.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/1 ©1996 FLAIRS

[1] The term *aggregation* used in this thesis and in the Natural Language generation community is not he same as the term *aggregation* used in the conceptual modelling community.

Aggregation has many different aspects. Consider the example below:

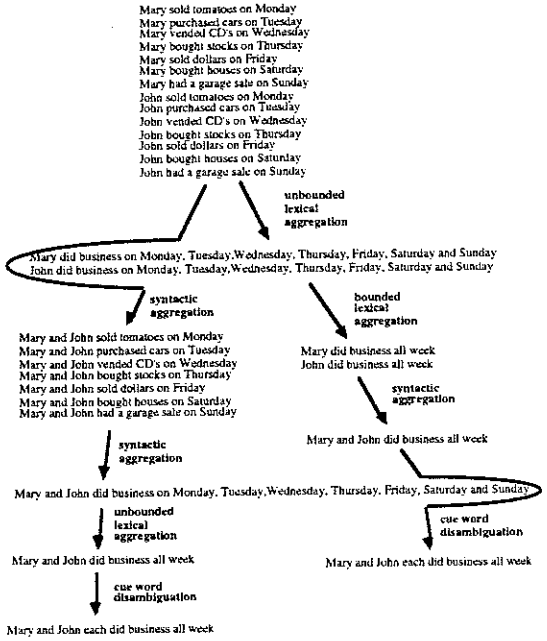


Figure 1 A text which is aggregated by syntactic and lexical aggregation rules and disambiguated by a cue word.

As we see there are several types of aggregation carried out during aggregation and there are also various orders to apply them. This example gives us a hint of the complexity of the aggregation processes.

In this paper we discuss aggregation during the sentence planning stage of NLG, in which we can distinguish four principal types:

1. *Syntactic* aggregation removes redundant information, but leaves (at least) one item in the text to carry the meaning explicitly. This is carried out at a pure syntactic level with no information loss about the content of the aggregated items.
2. *Elision* removes information that can be inferred and leaves no items in the text to carry the information explicitly, but the information remains there implicitly.
3. *Lexical aggregation* replaces a set of items with a new item, while the overall meaning is kept intact. This is carried out at a lexico-semantic level where information about the content of the aggregated items is needed.
   3.1 *Bounded lexical aggregation* keeps the overall meaning intact and the aggregated information is retrievable. Bounded lexical aggregation requires a known set with a fixed number of elements.
   3.2 *Unbounded lexical aggregation* may not keep the overall meaning intact and the aggregated information is not retrievable. Unbounded lexical aggregation requires an open set of elements.
4. *Referential aggregation* replaces redundant information with some sort of trace, such as a pronoun, to carry the information explicitly.

Syntactic aggregation have been discussed in [Dalia93], lexical aggregation in [Dalia95c] (in prep); referential aggregation have been discussed in [Wilki95]. Elision has been investigated in [McDon94]. Referential aggregation and elision lie outside the scope of this paper.

3. TYPES OF AGGREGATION

Linguists and Computational Linguists working with natural language analysis and parsing are interested in ellipsis or coordination because of finding the omitted pieces of a text, e.g., in [Dahl83,Sigur90]. They want to catch the real meaning of a text since they want to represent the meaning in some non-ambiguous representation. Computational Linguists working with Natural Language Generation are interested in "inverse ellipsis" or "inverse coordination" or more exactly what we here call aggregation. We want to generate ellipted sentences. The objective is to reduce redundancy and to avoid repetition.

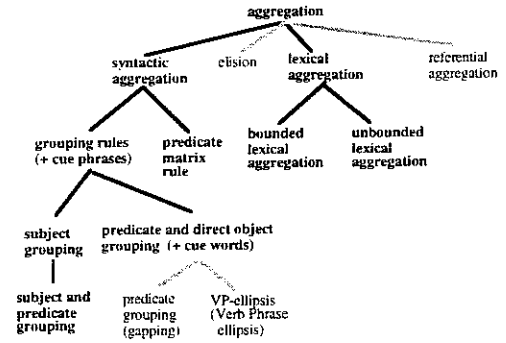


Figure 2 The tree describes the relation of the different aggregation types. The ones with the grey lines have not been investigated in this paper.

Figure 2, above, contains a tree clarifying the relation between various aggregation types. The bold text in the figure 2. shows the syntactic aggregation phenomena which have been treated previously in [Dalia93], where also a study of previous work on aggregation can befound. The syntactic aggregations found in [Dalia93] have been suggested to improve a NLG system and is described in [Dalia95a] and also implemented and described in [Dalia95b]. Syntactic aggregation has also been implemented in [Sigur92, Shaw95].

In linguistics, the results of aggregation is called ellipsis or coordination. [Quirk72] defines the strict sense of ellipsis when words are elided only if they are recoverable. Some type of ellipsis can function as ø-anaphora [Webbe79].

The motivation of ellipsis is to reduce redundancy and avoid repetition. [Quirk72] also includes a careful study of ellipsis; naming combined and segregatory coordination what we call (predicate and direct object grouping), as well

as ellipsis of subject and auxiliaries what we call (subject and predicate grouping). Neither Lexical aggregation nor Elision is mentioned at all. Authors who also have treated coordination, so-called syntactic aggregation, are [Oirso87,Gooda87].

The well-known term *gapping* corresponds to ellipsis of the first part of a predication in [Quirk72] which is redefined as *predicate grouping* according to this paper and not the predicate grouping defined in, [Dalia93].

4. GENERAL ISSUES: AGGREGATION AND SENTENCE PLANNING

Why should one carry out aggregation and what types of aggregations are there?

In corpora studies, [Dalia93c], we have studied in total 11 texts. The total amount of words in the nine first texts were 6452 words and the ratio (syntactic aggregation cases)/(total words) is 1.8% if you include the two last texts, the ratio (syntactic aggregation cases)/(total sentences) is approximately 33%; i.e. one third of the sentences includes syntactic aggregation.

If each aggregation saves approximately six words, this will make the text 1.8% aggregations x 6 words = 11% shorter, in some cases up to 20% shorter, than it would have been without aggregation in addition the text becomes easier to read. Eight of ten are subject and predicate grouping and the rest are predicate direct object grouping.

In further analysis of two additional texts (*Wall Street Journal 1992, March 24,* 60862 words and *Asiatisk Dagbok 1984,* 23860 words, [Dalia93c] containing together 84722 word and 5807 sentences in both English and Swedish, the ratio (Bounded Lexical aggregation cue words)/(total sentences) is 0.5%. I.e., we have at least 0.5% BL-aggregations, because the ones with no BL aggregation cue word are not visible or easy to find, when scanning a text automatically.

We estimate that aggregation shortens texts by 10-20%.

Why should one not carry out aggregation?

Syntactic aggregation and bounded lexical aggregation should always be carried out since the resulting text is shorter and no information is lost during the aggregation. One exception occurs when the Hearer's goal says it specifically to obtain certain information, e.g., the Hearer asks if the Speaker knows if *Mary did business this Sunday,* to which the Speaker should answer:

Yes, Mary did business this Sunday.

and not

Mary and John did business all week.

When losing information which is not retrievable, in for example unbounded lexical aggregation, one requires a good reason to carry out that aggregation, e.g., the Hearer wants to know if the Speaker knows if Mary did business, then the Speaker should answer:

Yes, Mary did business all week.

and not

Mary sold tomatoes, purchased cars, vended CD's.....and had a garage sale all week.

Another case preventing syntactic aggregation is when there exists a temporal relation between two mutually exclusive states which may be broken:

The wall is red. Tom paints the wall. The wall is yellow.

should not be aggregated to

The wall is red and yellow. Tom paints the wall.

Where should aggregation be carried out during natural language generation?

According to [Wilki95], aggregation can take place during every phase of NLG except during content selection and surface form generation.

In this work we take a slightly simplified view of the text generation process as a pipeline of three stages. Text planning (which determines the content and overall discourse structure of the text material), is followed by sentence planning (which decides on the sentence structure and scope), which in turn is followed by surface form realization (which is based on syntax).

In our view and also in [Dalia93,95a,95b,96] aggregation takes place after text planning, but before sentence planning. Aggregation operates mainly at sentence clause level, but it may also operate at discourse structure level, where however it may distort the discourse structure, so that the text becomes incoherent; in such cases new text planning is required.

How should one order input propositions before aggregation?

As shown in [Dalia93], ordering the input propositions is essential before applying aggregation rules. Certain combinations of input propositions give the optimal aggregation (the most consumed input propositions per aggregation rule), while other orderings do not permit aggregation.

The aggregation rules themselves may be ordered in various ways as well. This is treated in [Dalia95c]. In general first one should use the unbounded lexical aggregation rule (since that is

the most powerful) followed by the syntactic aggregation rules, then bounded lexical aggregation, and finally pronominalization (which is not aggregation but must be carried out as well).

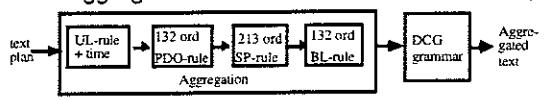


Figure 3. Overview of the input clauses ordering

What does a good text look like?

The concept *easily read text* has to be defined so it can be measured. Given a specific set of input propositions, to which a set of aggregations rules and a set of ordering rules in various combinations are applied, the result is a number of differently aggregated texts. Is it the case that the shortest text is best? Or is the Rhetorical Structure [Mann88] of the generated text the most important factor?

To answer these questions we performed a number of experiments on permuting and applying ordering rules and aggregation rules on a set of text plan clauses during generation, which is described in [Dalia95c]. We found that the Rhetorical Structure is a more important factor on deciding on which text is easily read than sentence length.

What happens after aggregation?

A side-effect as mentioned earlier is when applying aggregation at the discourse level the discourse structure may be distorted and new text planning is required. Also if aggregation occurs at sentence level the aggregated clauses may become unordered, i.e., incoherent, and reordering according to theme and focus may be required.

Aggregation may give rise to ambiguities: How can we prevent them?

One of the side-effects of aggregation is that an ambiguity can arise, because for example of problems with quantifier scoping. This is solved by using cue words such as*both, each, separately,* etc., which perform the disambiguation (see Fig. 1). The nature and use of cue words is discussed in [Dalia95d].

5. CONCLUDING REMARKS AND FUTURE DIRECTIONS

This paper have summarized some aspects of a longer work [Dalia96]. In that work the concept of aggregation has been defined and investigated. Aggregation contributes a novel part of the sentence planning phase of natural language generation, it clarifies the task of sentence planning and text realization in generation as well as discourse analysis and text linguistics.

The topic of aggregation is an important area since without aggregation automatically generated text is often very poor. This area has a direct application to real world problems in computational linguistics.

The work presented in this paper is a complement to research carried out by computational linguists and linguists working in the related areas as parsing and analysis of texts, since we have investigated different aspects of the ellipsis phenomena.

The concept of aggregation has been established in the Natural Language Generation community and the concept of aggregation has been used and referred to by other researchers [Kölln95,Shaw95,Wilki95]. Work summarized in this paper have been presented both at conferences for computational linguistics as well as for requirements engineering.

Implementation is important to reveal the nature of the aggregation phenomena and is a complement to empirical studies. Implementations in [Dalia95a,b,c,d] have been of great value to prove the findings in [Dalia93,95c,d] and have also lead to new discoveries in aggregation, such as the ordering problem of input text plan clauses, the distortion of discourse relations after aggregation [Dalia95c], and the problem of ambiguity after aggregation [Dalia95d].

Algorithms for carrying out syntactic and lexical aggregation during generation have been defined and implemented [Dalia95a,b,c,d]. A method for using cue words for disambiguation of aggregated text using the *and* -coordinator has been investigated and implemented [Dalia95d].

The solutions of the implementation problems provide guidelines for the architecture of an aggregation component and its relation to other generation components.

Aggregation in the sentence planning phase of natural language generation is required to make text non-redundant, short, and easily read. Aggregation is a fascinating research topic since it can be carried out at many different phases during natural language generation and it has various ways to be expressed, e.g., various syntactic and lexical aggregation types.

Many issues in the research area aggregation in natural language generation remain to be investigated. Our work is just the beginning. However, one problem is that research in aggregation is dependent on advances in other sentence planning areas such as sentence scoping, grouping and lexical choice.

In addition the use of cue words to disambiguate aggregated text is a topic which needs further investigation, for other coordinators, as for example, the *or* - and *but* - coordinators.

We have used the domain of requirements engineering for our aggregation generation. It would now be interesting to use other domains, as for example Medical Informatics [Ranki89a,89b, DiMar95] and Documentation and Technical manual generation [Rösne92, Svenb94].

Acknowledgements

Many thanks to Eduard Hovy for advising me, for stimulating discussions, and for fun both over the internet as well as in person at Information Sciences Institute/University of Southern California.

Reference

Dahl83 V. Dahl & M. McCord: Treating Coordination in Logic Grammars. In *American Journal of Computational Linguistics* Vol 9 No 2 April-June , 1983.

Dalia93 H. Dalianis & E. Hovy: Aggregation in Natural Language Generation. EWNLG-93, *Proceedings of the 4th European Workshop on Natural Language Generation*, Pisa, Italy 1993. Also in *Trends in Natural Language Generation: an Artificial Intelligence Perspective*, Adorni, G. & Zock, M. (eds.),Springer Verlag Lecture Notes in Computer Science (forthcoming 1996).

Dalia95a H. Dalianis: Aggregation in the NL-generator of the VIsual and Natural language Specification Tool. In *Proceedngs of The Seventh International Conference of the European Chapter of the Association for Computational Linguistics* (EACL-95), Student Session, pp 286-290, Dublin, Ireland, March 27-31, 1995.

Dalia95b H. Dalianis: Aggregation, Formal Specification and Natural Language Generation. In *Proceedings of the NLDB'95, First International Workshop on the Applications of Natural Language to Data Bases,*135-149, Versailles, France, June 28-29, 1995.

Dalia95c H. Dalianis & E. Hovy: On Lexical Aggregation and Ordering.(submitted to AAAI-96 and to INLG-96 workshop), 1995.

Dalia95d H. Dalianis: Natural Language Aggregation and Disambigation Using Cue Words. (submitted to ECAI-96), 1995.

Dalia96 H. Dalianis: Concise Natural Language Generation from Formal Specifications. (Forthcoming) Ph.D. dissertation,

DiMar95 C. DiMarco et al.: Healthdoc: Customizing patient information and health education by medical condition and personal characteristics. In *Proceedings of the Workshop on Patient Education*, Glasgow, 1995.

Gooda87 G. Goodall: *Parallel Structures in Syntax, Coordination, Causatives, and Restructuring.* Cambridge University Press, 1987.

Kölln95 M. Kölln: Employing user attitudes in text planning. In *Proceedings of the 5th European Workshop on Natural Language Generation*, pp. 163-179, Leiden, the Netherlands, 1995

Oirso87 R. R. van Oirsouw: *The Syntax of Coordination*, Croom Helm, 1987.

Quirk72 R. Quirk et al: *A grammar of contemporary English.* Longman Group, Limited, 1972.

Ranki89a I. Rankin: Deep generation of a critique. In*Proceedings of the Second European on Natural Language Generation Workshop*, Edinburgh, April 6-8th 1989.

Ranki89b I. Rankin: The Deep Generation of Text in Expert Critiquing Systems, Licentiate Thesis No 184, Linköping University, 1989.

Rösne92 D. Rösner & M. Stede: Customizing RST for the Automatic Production of Technical Manuals. In *Aspects of Automated Natural Language Generation*, R. Dale et al. (eds.). Springer Verlag Lecture Notes in Artifical Intelligence no. 587, pp. 199-214, 1992.

Shaw95 J. Shaw: Conciseness through Aggregation in Text Generation. In *Proceeding of the 33rd Annual Meeting of Association of Computational Linguistics*, 26-30 June 1995, (ACL-95), Student Session, MIT, Camebridge, Massachussetts, USA, pp 329-331, 1995.

Sigur90 B. Sigurd & P. Warter: Understanding Coordination by Means of Prolog. Working Papers 36, pp 151-162, 1990, Department of Linguistics and Phonetics, Lunds University, 1990.

Sigur92 B. Sigurd et. al.: Automatic translation in specific domains: weather (Weathra) and stock market (Stocktra, Vectra). *Praktisk Lingvistik 15, 1992*, Department of Linguistics, Lund University, 1992.

Svenb94 S. Svenberg: Representing Conceptual and Lingusitic Knowledge for Multilingual Generation in a Technical Domain, *Proceedings of the 7th International Workshop on Natural Language Generation*, Kennebunkport,June, 1994.

Swart82 B.Swartout: GIST English Generator, *Procedings of AAAI-82.*, American Association of Artificial Intelligence, Pittsburg, Pensylvania, 1982.

Webbe79 B. Webber: *A Formal Approach to Discourse Anaphora.* Garland Publishing Company, Inc, 1979.

Wilki95 J. Wilkinson: Aggregation in Natural Language Generation: Another Look, Computer Science Department, University of Waterloo, Canada, (unpublished M.Sc. thesis) 1995.

Extracting Software Test Cases Directly from Documentation

Patricia Lutsky
Brandeis University
pel@cs.brandeis.edu

Abstract

This paper describes SIFT, a tool for automatically generating software tests by extracting information from software documentation. Although SIFT uses domain-oriented techniques, they are sufficiently powerful for this application.

1. INTRODUCTION

Extracting information from texts is a central part of software testing because frequently testing involves comparing a software system to a reference document. If the system behavior does not match the written description, there is a defect in the software, the document, or both. Depending on the development organization, the reference document may be a formal functional specification, a user's guide, or an informal document.

For each entity under test (each API routine or each command), certain testing techniques establish conformance to the documentation. Often, once the basic techniques are identified, repeatedly applying them to each interface can be tedious, so important progress in automating software testing has been made. [5][12] These systems allow for partial automation of generating and running tests. A natural extension to further automation of testing is to process the documents automatically and extract test-related information from them using natural language parsing techniques. This addition is generally overlooked because of the potential complexity of parsing unrestricted natural language texts. However, the texts of software documents are a restricted sublanguage [4][6], and the SIFT document parser [7][8] has demonstrated progress in automatic information extraction by partial parsing; it only extracts specific information relevant to testing.

SIFT, which stands for specification information from text, allows software development organizations to use texts found in existing artifacts rather than requiring special-purpose encodings of testing-related information. This text, while naturally produced, is stylistically limited and therefore easier to parse than many other texts. The cultural context of documentation writing leads to stylistic conformity. Documents for external use, such as user's guides or reference manuals, generally have editors who enforce writing style. Even informal documents such as database dictionaries show stylistic conformity because as different writers add or update sections, they mimic the existing style.

SIFT is successful because it looks for specific types of information in the texts. It does not try for full-text understanding, thereby bypassing issues of discourse structure. It works on individual sentences, so only concepts that can be expressed in individual sentences can be extracted. However, many important testable conditions are expressed this way, including allowable value ranges for integer parameters, legal input formats, prerequisites such as required privileges for operations, and parameter interdependencies. Furthermore, if information is available in the texts but is not expressed simply enough, SIFT's inability to extract this information will not adversely affect the test system. It will just require that more tests must be added manually.

A tester using SIFT analyzes a text to identify those concepts which would enhance testing and which can be extracted. Then he or she looks in detail at how those concepts are expressed in the document. Specific sentence types are encoded in a grammar and SIFT uses this grammar to extract the information from the document. As the document is updated over time, the grammar continues to extract the latest information from new versions of the document.

Software documents are usually structured, with separate sections describing different entities. Each text section has a fixed-format header. This partially-structured nature of the text assists in anaphora resolution and ellipses expansion. For example, SIFT keeps track of the name of the entity being described and uses it to fill in missing sentence components.

2. SIFT PROCESSING STEPS

SIFT text processing is done in four steps, and a SIFT grammar has four corresponding parts:

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/6

1. Preprocessing directives
2. Phrase structure grammar
3. Heuristics for identifying sentence types
4. Heuristics for generating canonical forms

Preprocessing varies depending on the tool used to produce the document; it includes actions such as stripping out characters the parser does not handle or identifying appropriate sections of the document to pass to the parser. For instance, if the document was developed using a SGML-type markup language, formatting commands such as "<ARGITEM>" would signal the beginning of each argument section, and the preprocessor would look for these commands to locate the argument descriptions.

The phrase structure grammar describes the syntax of the sentence types using a LISP-based language. The grammar also describes the format of the phrase structure tree that should be output as a result of this step. The format of this tree can be tailored to the type of information being extracted.

This phrase structure tree is passed to heuristics which identify the sentence type. During the text analysis phase, the tester has defined sentence types which describe testable conditions. These heuristics analyze the phrase structure tree to decide which sentence type(s) it can be. For some grammars, the phrase structure tree may contain specific clues about the sentence type.

The last processing step uses heuristics to extract relevant semantic information from the phrase structure trees. These heuristics can take advantage of the sentence classification done in the previous step. As with the rest of the sublanguage grammar, they are expressly tailored to the type of information being extracted and can therefore be quite simple. The output of this step is a canonical representation of the semantic content of the sentence, and this final representation is domain-specific.

3. APPLICATION AREAS

SIFT's potential has been demonstrated for two domains: OpenVMS operating system [3] testing and XCON expert system [1] database testing. Both of these had automated test systems developed for them, and SIFT was able to augment these test systems.

3.1. OpenVMS

The TESTGEN system [5][9] was developed to test the OpenVMS API, such as the system services. It allows a tester to define frames [2] and reusable software components (RSC's). The RSC's are merged into the frames, thereby generating complete tests. A database of testcase RSC's for various scenarios is maintained, and system service parameters are identified as types such as character string descriptor, longword address, or integer that translate to indices into the testcase database. Basic indices were automatically extracted from the definition files of the routines; SIFT then allowed the indices to be more fine-grained. For instance, rather than just identifying a parameter as a "character string," SIFT allows for identification as a character string with a minimum and maximum length.

Information about length restrictions for character strings was expressed in various ways in the system services reference manual [3], but a finite number of sentence types were identifiable. Examples of these are

```
1. The maximum length of the
name is 15 characters.

2. The name string may be from
1 to 31 bytes in length.

3. An identifier name consists
of 1 to 31 alphanumeric
characters including dollar
signs and underscores,
containing at least one
nonnumeric character.

4. The BRKTHRU service permits
the message text to be as long
as 16,350 bytes; however, both
the SYSGEN parameter MAXBUF
and the caller's available
process space may affect the
maximum length of the message
text.
```

Sentences of types 1 and 2 are short and straightforward. Sentences of type 3 are more complex, yet appeared several times in the document and warranted inclusion in the grammar. Sentence 4 is an example of a sentence that is specific to a parameter and was not included in the grammar.

A SIFT grammar for eight sentence types was written and run on the whole document. The information extracted allowed TESTGEN to generate an additional 96 system service testcases.

3.2. XCON

For the XCON expert configuration system, a system called CARPER [11] has been developed for database testing. It uses machine learning techniques to identify patterns in the XCON database that can be used to detect anomalous entries in the database. The XCON development organization maintains an internal document, the database dictionary, that includes information about each class and each attribute in the database. Specific extractable test-related information in the database dictionary is the following: attributes that must be used together, attributes that must not be used

together, attributes that must be used for a class, and attributes that are illegal for a class. Example sentences for the sentence types identified include

```
5. If BUS-DATA is defined then
BUS must also be defined.

6. This attribute is
appropriate only for class
SYNC-COMM.

7. Should not be defined for
class typesetter-interface.
```

These were translated into one of the following canonical forms:

```
ATTRIBUTE must [not] be
defined if ATTRIBUTE is
[not]defined.

ATTRIBUTE must [not] be
defined for CLASS.

ATTRIBUTE can only be defined
for CLASS.
```

CARPER could then use this information to augment or verify the tests found by machine learning.

4. EXAMPLE HEURISTIC

SIFT's heuristics for semantic processing and for sentence classification can have a narrow, domain-specific focus. For example, in OpenVMS testing heuristics were developed to distinguish conditional sentences where the antecedent and the consequent both concern values of parameters from sentences where the antecedent or consequent concern the actual execution of the routine. The type of testcase hinges on this distinction: whether it is a simple testcase of setting a parameter a certain way and calling the routine, or if the testcase involves elaborate system setup before the routine is invoked.

Sentences in the first group express testable conditions such as default values, as in

```
8. If not specified, the
default value 0 is used,
```

restrictions on allowable values, as in

```
9. If you specify a delta
time, it must be less than
10,000 days,
```

and relationships between parameters, as in

```
10. If you specify OBJNAM, you
must omit CHAN or set it to 0.
```

After analyzing the first twenty-three OpenVMS system service routine descriptions, the heuristics developed for identifying conditionals about parameter restrictions were the following:

```
If the verb phrase used in the
consequent is a modal indicating
necessity ("must" or "should" but
not "can") then it is this type.
                OR
If the consequent is expressed as a
generic with "use" as its verb then
it is this type.
```

The first heuristic shows that modal necessity is only used to refer to parameter values, not to execution of the routine. The second shows that "the routine uses X" is the generally-accepted way to describe a routine's handling of parameters.

These heuristics were then tested on the descriptions of the last 24 system service descriptions. Despite their simplicity, they were quite accurate, correctly identifying five conditionals that concerned parameter restrictions. They did not return any inappropriate conditionals, but they did miss one additional conditional sentence about a parameter value. This led to testing them for more general usefulness, on a manual written by a different software vendor [10]. The heuristics did not work well for this manual. Several appropriate conditionals were missed and irrelevant sentences were identified. This result indicates that SIFT can be powerful within a domain but that since each grammar is specifically tailored to the sublanguage used in a set of manuals, it depends heavily on specific stylistic conventions.

5. SUMMARY

SIFT has been effective in augmenting two test systems by extracting information from the texts of documents. The SIFT architecture has domain-dependent modules and domain-independent modules so grammars for different domains can be plugged into the SIFT framework. The parsing techniques used are simple and domain-specific yet sufficient for the task. Current work concerns investigating what is needed to allow more complex types of information to be extracted from documents.

References

[1]Barker, Virginia, & O'Connor, Dennis (1989). Expert systems for configuration at DIGITAL: XCON and beyond. Communications of the ACM, 32. 298-318.

[2]Bassett, Paul (1987). Frame-based software engineering. IEEE Software. 7/89. 9-16.

[3]Digital Equipment Corporation (1988). OpenVMS System Services Reference Manual Version 5.0.

[4]Grishman, R. & Kittredge R. (Eds.). (1986). Analyzing language in restricted domains: Sublanguage description and processing. Hillsdale, NJ:Lawrence Erlbaum Associates.

[5]Jones, Larry (1984). Regression testing of VMS. Proceedings of the Digital Equipment Computer Users Society. 611-618.

[6]Kittredge, R., & Lehrberger, J. (Eds.). (1982). Sublanguage: Studies of language in restricted semantic domains. New York:Walter de Gruyter.

[7]Lutsky, Patricia (1992). Document parser to extract software test conditions. Proceedings of the Annual Meeting of the Association for Computational Linguistics. 294-296.

[8]Lutsky, Patricia (1994). Using a document parser to automate software testing. Proceedings of the 1994 ACM Symposium on Applied Computing. 59-63.

[9]Lutsky, Patricia & Jones, Larry (1988). Improving software reuse in software testing. Digital Equipment Corporation Internal Memo.

[10]Microsoft Corporation (1992). Microsoft Windows Software Development Kit Programmer's Reference, Volume 2: Functions.

[11]Schlimmer, Jeffrey (1991). Learning meta knowledge for database checking. Proceedings of AAAI 91. 335-340.

[12]Ziman, L. & Dickau, M. (1988). Project management of the VAX DEC/Test Manager Software Version 2.0. Digital Technical Journal. 6.

Address Recognition with Robust NLU Technology

Michael H. Malburg
German Research Center for Artificial Intelligence (DFKI GmbH), Kaiserslautern/Germany
malburg@dfki.uni-kl.de

Abstract

An analysis component recognizing addresses on printed documents and identifying the respective sender or recipient is presented. The component's kernel is a standard chart parser for feature-based context-free grammars. The parsing procedure is enhanced with the island-parsing strategy and the ability for partial parsing. The needs of the real-world application within document analysis of printed documents are met by several pre-processing steps. Additionally, a robust address matcher compares the parsing result with the entries of an address data base. Thus, understanding in this context is the identification of a database entry appropriate to a printed address. The described work has been implemented as a component within a large document analysis system. First evaluation results for 230 documents have recently been worked out.

INTRODUCTION

The application of address recognition may be best known from systems used to sort mail pieces for postal services, e.g., the United States Postal Service (USPS). Common goal of such systems (e.g., [Srihari &al 1993], [Vayda &al 1993]) is the identification of the ZIP+4 code which can be used for routing the mail pieces. In order to reach this goal, one has to recognize ZIP code, city name, state name, street, and post office box number, and to retrieve the ZIP+4 code from the USPS database by matching it against these data.

In our approach, we go beyond this by analyzing the whole address including the addressee as a person or organization. This identification of person, organization, and address is our realization of "understanding", meaning, we perform a shallow understanding in the same sense as "understanding" is used throughout document analysis (opposed to deep natural language understanding).

There are two further differences to typical systems for postal applications. Firstly, we focus on machine printed characters which eases the OCR and simplifies the word recognition component. Secondly, our application scenario is an in-house mail handling rather than mail piece dispatch in a postal service.

The major challenges in the field of address recognition are:

- several pre-processing steps such as noise reduction and layout segmentation;
- handwriting recognition (cf. [Srihari &al 1993]) or printed character recognition (cf. [Vayda &al 1993]) respectively;
- word recognition, e.g., by means of dictionary access or Hidden Markov Models;
- parsing of geometric objects or words respectively;
- some kind of database matching for the best hypotheses yielded.

Although this paper describes some aspects of the first three items, the central topic of the described work are the latter two items. Main difference to existing approaches, such as those mentioned above, is the use of robust language technology for this task. The work described continues the development of [Malburg&Dengel 1993] where a simpler parsing technique was used and less efforts in database matching were undertaken.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/10

Overview to this Paper

Since document analysis may not be known as a typical field for use of robust NLU technology, a short introduction will give an impression of this research area building the context of the described work. The succeeding description of our current application domain will motivate why the analysis of printed addresses is a relevant sub-task outside the scope of mail delivery systems. Afterwards, our solution for this sub-task is described in detail. The component description is concluded with the evaluation of strengths and weaknesses of the approach and a short résumé. Suggestions on further improvement and a comparison of the robust approach to its "classical" kernel also are given.

Document Analysis

This work has been undertaken in the context of the OfficeMAID document analysis system at DFKI Kaiserslautern (German Research Center for AI). OfficeMAID is a prototypical system for the (integrated) analysis of printed documents. It starts from a scanned document image and generates an electronic document representation of a certain level of coarse document understanding. Most important phases of this system are document segmentation, logical labeling, character and word recognition (for these cf. [Fein&al 1995]) and several steps of partial text analysis and understanding (cf. [Baumann&al 1995]).

Document segmentation provides the layout structure of the document with blocks, lines, words etc. Starting from this layout structure, a so-called logical labeling may be performed where labels representing the expected contents of these blocks are attached to some layout blocks. For a scientific paper such labels typically are "title", "author", "heading", "plain text" etc. For a business letter such labels are "recipient", "sender", "subject", "body", and so on. For all layout blocks containing textual elements, an optical character recognition (OCR) is performed usually resulting in alternative readings for each printed character. These OCR-results are then matched against a given dictionary supplying ambiguous word recognition results. At this point, the tasks of document understanding are started. Such tasks are highly application-dependent and therefore a restriction to a certain application domain usually is done.

Business Application

The current application of OfficeMAID is – as the name may suggest – the handling of printed business letters. More precisely, we are actually analyzing the documents of the local university's purchasing department. Such documents are demand's announcements, inquiries, offers, orders, confirmations, delivery notes, invoices, and (sometimes) demands. Our "final" goal is the electronic archiving of roughly analyzed documents in order to support the department's work flow. Thus, a support of work flow management systems or groupware systems could be done as well as a simple notification of a letter's arrival for the clerk concerned.

Correspondent Identification

One central sub-task of this system is the identification of correspondence clerks for letters, i.e. the recognition of sender and recipient. The respective analysis "specialist" is designed as follows.

First of all, the text block expected to hold the relevant sender or recipient address has to be found. This step is referred to in the following as direct labeling since special heuristics are implemented which "directly" lead analysis to the respective text blocks – at least in case they work correctly.

Presumed, the text is already recognized, now (secondly) a so-called re-segmentation starts. This can be compared to the scanner in classical NLU systems, but is slightly different (more about this later).

The resulting lattice of word hypotheses is, thirdly, lexically pre-processed by standard means (lexical access) and heuristic means (default values).

Fourthly, the component's kernel is started: the address parser which terminates in yielding one or more complete or partial parses.

Fifthly and finally, the resulting parses are matched against the entries of a given address data base.

In the following, we put together the steps direct labeling, re-segmentation and lexical pre-processing as pre-processing steps. Afterwards, we describe the parsing and the database matching tasks. But, before we will do so, we have to spend a few words on preparatory steps, i.e., making lexicon, grammar, and address data base. These are required to be performed before the analysis task can start off.

PREPARATORY STEPS

As usual for real-world applications, acquisition and design of lexicon, grammar, and – in this case – the address data base needs some relevant effort and thus, although scientifically almost boring, will shortly be acknowledged.

In document analysis, recording of so-called Ground Truth (GT) data is performed for such results the system is expected to generate automatically later-on. Such GT data can easily be understood as a collection of exactly that information a small jury of humans agree to be central and correct for a given document. For the given application, GT is a template of information to be extracted. This GT data is similar to those templates defined in the context of the MUC-conferences (cf. [MUC-3 1991], [MUC-4 1992]; also [Lehnert & Sundheim 1991]). Consequently, the evaluation described below is undertaken in analogy to the evaluation of MUC.

Starting from this GT data which comprises both message-level and person-specific data, a lexicon and a address data base are compiled. In order to reduce manual effort, GT data is recorded at some level between word and phrase. For address GT this means in concrete that the slots to be filled are organization name, company type (e.g., "Ltd."), department name and type, person's first name and surname, etc. In other words, GT is not always given at word level and, thus, not part-of-speech tagged. Therefore, several heuristics have to be used within the lexicon generation step. A typical example are such cases, where a company's name is constructed by its owner plus the company type, e.g., "Peter Stuyvesant & Sons, Ltd.". In this particular case, the occurrence of the keywords "Sons" and "Ltd." is sufficient for the hypothesis that "Peter" is the forename and "Stuyvesant" is the surname of the company's owner. Anyway, such heuristics often fail, since there are several similar "meanings" possible for one writing type; for example compare the two names "Peter Stuyvesant Corporation" and "General Electric Company".

Additionally and together with the hand-crafted grammar, a small lexicon of "function words" is drawn up manually. Such function words are the German pendants to "P.O.Box", "Mister", "Mistress", etc. In detail, each word having a special role within the address grammar is attached an entry in the manually generated lexicon.

PRE-PROCESSING STEPS

Pre-processing steps are direct labeling, re-segmentation, and lexical assignment.

Within direct labeling, the recipient address and the sender's address (located directly above the recipient) are labelled, i.e., marked.

Our heuristic in-use is mainly like this: look at the left-hand-side of the upper third of the letter's first page. If you find a text block with three to six lines, optionally last line more spaced, lines equally sized (font size) and left adjusted, then suppose this is the recipient address. If you found this and if there is one smaller line right above it, with same left margin, suppose this is the sender address. Please note that any other sender information, esp. in the upper right part or in the bottom line of the letter, is not analyzed at all.

This simplified heuristic yields (approximately) in 3 out of 4 cases the correct recipient block, and in 1 out of three the correct sender block. In some cases, the selected lines only intersect with the correct address block wherefore partial parsing is necessary. Important to mention: in 1 out of 3 cases, there is no sender address on our test documents. In comparison to approaches used for mail sorting (cf. [Palumbo &al 1992]), finding the address block on printed letters is far more complicated which results in the higher error and rejection rate.

The task of re-segmentation is mainly the following. In all cases, where a word ends in a dot, hyphen or something similar, the respective character is chopped off this word. This task is best understood when thinking of layout segmentation as a "stupid" scanner only cutting on spaces. The other way round, numbers are often printed spaced, e.g. in groups of two letters. This is also a job to be done by re-segmentation.

Last pre-processing step is two-fold lexical assignment: lexical access and use of default values. The lexical access is nothing special and should almost ever yield the correct reading. Additionally and especially for those cases where a word is unknown, a default assignment is performed. The respective default categories depend on the character contents of the recognized words, e.g. 5-digit numbers are assumed to be ZIP codes, words with upper-case initial letter are assumed to be names, and so on. More interesting are those cases, where more precise word categories can be assumed which will be better weighted for analysis. Such are street names typically ending in "street", "alley", "way" etc., and town names which also have typical endings, compare the English "town" or "ton".

Such heuristics are important for calculating rejects at the end of a parse. For example, if a good weighted parse is calculated but no match is found then it is better to reject instead of finding some possible match for the next best parse. The heuristics also help to accept addresses which are recorded in the address data base but have not yet been compiled into the lexicon.

PARSING AND DATABASE MATCHING

Subsequently, the parser is invoked to calculate valid parses. The parsing itself is standard and therefore not described in more detail. It works on an feature-based grammar formalism related to PATR-II (cf. [Shieber 1986]) and consists of an island-driven chart parser (i.e. a parser driven by recognition probabilities). The island parsing strategy is adapted from [Woods 1982].

Partial parsing is performed in the following way. Since the components of an address are easy to delimit, a kind of case frame is defined consisting of the parts town, street (alternatively P.O.Box number), person, department, and institution; some minor parts like building, etc., may occur, too. This frame is mapped to the resulting feature structure of a complete parsing result and thus a uniform post-processing can be done for partial and complete parses. For partial parses, strict rules are given how to fit a partial parse into this frame. Thus, several semi-complete frames can be constructed from a set of partial parses.

Data base matching is carried out as follows. For each pair of parse and data-base entry in question, a small formula is generated describing the weights of the parts of the mentioned frame. The principle of this formula is shown in Table 1 for matching of organization (O), department (D), and person (P) between parsing result and database entry. Similar formulas have to be generated for different address types (P.O.Box vs. street, in-house addresses, etc.) as well as for all constituents of these components. For example, a person name itself is constituted by forename, surname, title, sex, and (sometimes) function.

As an example, take the following situation. The parser identified a person, and the actual database entry to match against consists of a person which is associated with an institution. In this case, the resulting formula – in part given by Table 1 – is

1/5 P + 1/5 S + 1/5 T,

where S and T denote the weights for the street and the town matches, respectively. In contrast, a person-person match is weighted

3/5 P + 1/5 S + 1/5 T.

Thus, an exact match between person and person will be better weighted in the overall weighting sum than a (probably partial) match between person and person plus institution.

Table 1: Weighting formulas (sketchy)

Parse / GT	O&D&P	O&P	P	O&D	O
O&D&P	$\frac{O+D+P}{3}$	$\frac{O+P}{2}$	$\frac{P}{2}$	-	-
O&P	$\frac{O+P}{3}$	$\frac{O+P}{2}$	$\frac{P}{2}$	-	-
P	$\frac{P}{3}$	$\frac{P}{2}$	P	-	-
O&D	$\frac{O+D}{3}$	$\frac{O}{3}$	-	$\frac{O+D}{2}$	-
O	$\frac{O}{3}$	$\frac{O}{2}$	-	$\frac{O}{2}$	O

This kind of formulas is used not only for the address level but – in part – also for its parts. At word level only a sub-string match is performed and weighted w.r.t. the length of the string in the database entry. For ZIP codes a partial match (for similarity of numbers) is done since ZIP codes often happen to be "slightly" wrong; i.e., errors in the lower significant digits occur quite frequently.

This way, a list of weighted pairs of parse and database entry is calculated. Actually, the total weight of such a pair is calculated by simple multiplication of parse weight and match weight, whereby the parse weight is given by the bottom-up probabilities yielded by word recognition. Actually, there is no additional knowledge used during database matching beneath that described above. In particular, an exploitation of the database's internal structure is not performed. The database entries are organized in a hierarchical manner using links between persons, department and organizations, and therefore avoiding redundancy in multiple address recordings. This hierarchical structure would enable a more efficient search over the entries by integrating search and match process which currently is not exploited.

EVALUATION

As mentioned above, the evaluation of this component is done using MUC measures and metrics (cf. [Chinchor 1991] and [Chinchor 1992]). In particular, we use the measures recall, precision and overgeneration for evaluation. Table 2 shows the results for identification of sender and recipient with complete and partial parsing, respectively.

The results show complete parsing having a higher precision (up to 100%) but a terribly bad recall (11% for sender and 17% for recipient) while the partial parsing approach is not necessarily less precise (at least for the recipient a precision of 83% is reached) but far better in recall (up to 61%).

Table 2: Results

	Recall	Precision	Overgen.
Sender (Comp.)	11.49	100.00	0.00
Sender (Partial)	16.09	50.00	3.57
Recipient (Comp.)	17.24	95.24	0.00
Recipient (Partial)	61.21	82.56	1.16

The bad precision in partial sender recognition is due to the weak heuristics which cause a reject. Additionally, the sender address is almost ever in a poor printing quality and not ever given an a letter. Since the sender address is typically in a small font, the hypotheses yielded by text recognition are often similarly bad weighted. But the adjustment of the respective threshold for rejects highly depends on the test set and therefore an optimal adjustment has not yet been done.

The overall "bad" recall is due to the restricted heuristics of direct labeling which sometimes cannot find the required address block. In order to improve total recall of this component, a text-based module for logical labeling is under development which will be used in combination to the layout-based one.

RÉSUMÉ

We presented the novel application of robust NLU technology for recognition and analysis of printed addresses. Central novel aspects of this application are the parser and grammar type used as well as the technique for database matching.

For description of addresses, an attributed context-free grammar is used which exceeds the descriptive power of other approaches. The parsing technique incorporated comprises both recognition-driven island parsing and certain heuristics for partial parsing. After parsing is finished, the best hypotheses are matched against the address data base for retrieving the sender or recipient. Therefore, a declarative description is used of how a match between parse and database entry is weighted.

Furthermore, the parameters of this declarative description seem to be learnable which probably will be subject of our future work.

References

[Baumann&al 1995] Stephan Baumann, Michael Malburg; and Hans-Günther Hein, Rainer Hoch, Thomas Kieninger, Norbert Kuhn: *Document Analysis at DFKI – Part 2: Information Extraction.* DFKI Research Report RR-95-03, DFKI Kaiserslautern, Germany, March 1995

[Chinchor 1991] Nancy Chinchor: *MUC-3 Evaluation Metrics.* in: [MUC-3 1991], pp. 17-24

[Chinchor 1992] Nancy Chinchor: *MUC-4 Evaluation Metrics.* in: [MUC-4 1992], pp. 22-29

[Fein&al 1995] Frank Fein, Frank Hönes, Thorsten Jäger, Achim Weigel, and Majdi Ben Hadj Ali: *Document Analysis at DFKI – Part 1: Image Analysis and Text Recognition.* DFKI Research Report RR-95-02, DFKI Kaiserslautern, Germany, February 1995

[Lehnert&Sundheim 1991] Wendy Lehnert and Beth M. Sundheim: *A Performance Evaluation of Text-Analysis Technologies.* AI Magazine, Volume 12/91, Fall 1991, pp. 81 - 94

[Malburg&Dengel 1993] Michael H. Malburg and Andreas R. Dengel: *Address Verification in Structured Documents for Automatic Mail Delivery.* in: Proc. 1st European Conf. dedicated to Postal technologies, Nantes, France, June 1993 (SRTP, Nantes), pp. 447-454

[MUC-3 1991] *Proc. 3rd Message Understanding Conference (MUC-3).* Morgan Kaufmann Publishers Inc., San Mateo, California, 1991

[MUC-4 1992] *Proc. 4th Message Understanding Conference (MUC-4).* Morgan Kaufmann Publishers Inc., San Mateo, California, 1992

[Palumbo&al 1992] Paul W. Palumbo, Sargur N. Srihari, Jung Soh, Ramalingam Sridhar and Victor Demjanenko: *Postal Address Block Location in Real Time.* in: Computer – Transcribing for Reuse – Document Image Analysis Systems (special issue), June 1992, pp. 34-42

[Shieber 1986] Stuart M. Shieber: *An Introduction to Unification-Based Approaches to Grammar.* CSLI Lecture Notes, Number 4, Stanford, California 1986

[Srihari&al 1993] Sargur N. Srihari, Venu Govindaraju, and Ajay Shekhawat: *Interpretation of Handwritten Addresses in US Mail Stream.* in: Proc. 1st European Conf. dedicated to Postal technologies, Nantes, France, June 1993 (SRTP, Nantes), pp. 421-428

[Vayda&al 1993] Alan J. Vayda, Michael P. Whalen, Daniel J. Hepp, and Andrew M. Gillies: *A Contextual Reasoning System for the Interpretation of Machine Printed Address Block Images.* in: Proc. Second Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, Nevada, USA, April 1993, pp. 429-441

[Woods 1982] W. A. Woods: *Optimal Search Strategies for Speech Understanding Control.* in: Artificial Intelligence 18, North-Holland Publishing Company 1982, pp. 295 - 326

Beyond Concept Recognition

Mohsen Rais-Ghasem
Jean-Pierre Corriveau

Carleton University
Ottawa, Ontario
Canada K1S 5B6
{mohsen, jeanpier}@scs.carleton.ca

Abstract

This paper addresses the problem of language comprehension from the viewpoint of conceptual categorization. We argue that existing NLP systems, be they symbolic or connectionist, generally rely on *static* definitions: once trained or hand-coded, the meaning of a concept is fixed. We instead, propose an interpretative model based on dynamic contextual categorization and show how it offers a more context-sensitive account of meaning. Within this framework, we also briefly investigate issues such as sense construction, ad hoc categories, and unmediated and incremental learning.

1 INTRODUCTION

This article considers meaning in natural language processing (NLP) systems. In particular, we address issues such as concept recognition, contextual effects, ad hoc categories, polysemy and idiosyncrasy. We argue for a concept-based approach to meaning (as opposed, for example, to a syntactic one). Our main hypothesis is that "deep understanding" of an utterance is attainable only if we succeed to *properly categorize* each concept in this utterance. More specifically, each concept should be subsumed by a *category* that comprises all relevant *instances* (i.e., usages of concepts) that the system remembers. A category in this sense differs from conventional categories in that a) it is *dynamically* formed, and b) context and background play a major role in its construction. Instantiation, i.e. sense selection, is complete once an object is classified in this fashion.

To get a general idea of our proposal, consider the following sentences :

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/15

1. John raised the canvas.
2. John painted the canvas.

We suggest understanding is not complete unless we manage to categorize *John*, *raised*, *canvas*, and *painted* properly. For instance, *John* should be classified along with entities which are remembered as raising other things in the first sentence (including to some degree, *cranes*); and with painters in the second one. Similarly, *raised* should be associated with all actions involving physical elevation (e.g., *elevate*, *pick up* etc).

Though trivial, these examples raise some interesting issues. First, "entities capable of raising" illustrates an ad hoc category: we probably do not systematically store such a category. Note that we do not challenge the fact that "being capable of raising" is an inferable characteristic for humans. All we are suggesting is that this property is not always (i.e., independently of context) ascribed to humans. Second, *painted* has at least two different, yet related, senses: painting as coloring and as drawing pictures. We argue that static preference of the second sense over the first one, only on the basis of *canvas*, is inappropriate. Instead, we depend disambiguation not only on the context but also on the reader's background knowledge.

The next section briefly reviews related ideas in NLP and cognitive psychology. In section 3, we sketch out our proposal and provide examples of the model in work. Section 4 briefly compares the model with pertinent other work. The last section discusses the proposed model and its adequacy in addressing the intended problems, as well as its shortcomings.

2 BACKGROUND

The importance of categorization in human cognition has been emphasized for a long time (see for example Lakoff, 1987; Medin & Goldstone, 1990). Over the centuries, several theories have attempted to explain human conceptual structure. Roughly put, categorization theories range between two extremes: rule- and similarity-based paradigms (for review, see Oden, 1987).

Recently it has been suggested that both mechanisms are present in human cognition, stirring work on hybrid models (Nosofsky et. al. , 1994; Smith & Sloman 1994; Ahn & Medin 1992).

While the main emphasis of psychological studies of categorization is on item recognition, NLP systems often focus on sentences and higher level structures. Concepts, in most NLP systems consist of unchanging prespecified building blocks used in meaning construction. Therefore, in such models, concept identification is reduced to concept recognition. Such approaches are clearly similar to the classical view of categorization, and thus suffer similar inadequacies (e.g., no graded membership, no prototype effects, no contextual effects, etc.). For example, concepts in LeMICON (Bookman, 1994) are defined based on a number of fixed semantic primitives, or micro-features (Waltz & Pollack, 1985). *Hospital* in LeMICON, for instance, is *always* associated with pain and other unpleasant feelings, regardless of its appearance in pleasant contexts, as in recovering from a fatal disease or giving birth. This view is specially problematic in dealing with context-dependent properties. For example, it does not explain why after reading a sentence about *floating on a basketball*, we do build a new attribute for *basketball* as something you can float on (Barsalou, 1982). Another major deficiency arises in handling polysemyous words: Most computerized lexicons maintain separate entries for distinct senses, ignoring the existing systematicity among them. WordNet for example, lists 6 different senses for *home*; and 10 for *read*. This approach increases the complexity of the model using the lexicon and limits its ability to deal with novel senses. Moreover, the cognitive plausibility of independent separate senses (i.e., strong homonymy) has been questioned (see Lakoff & Johnson, 1980). Also such models are inherently unable of handling sentences like the following (from Rastier, 1991) :

3. A dime is a dime.
4. A father even disciplining his son is a father.

3 THE MODEL

A summary of the problems introduced above will clarify our goals and motivations.

1. Properties forming the "definition" of a concept should vary as the model encounters new relevant evidence. Thus, comprehension is diachronic (see Corriveau, 1995) and requires an incremental knowledge acquisition scheme. In turn, this allows for a personalization of concepts (*Ibid.*). For example, *water* may include, in its definition, the features *trauma* and *danger* for a child who has lost a sibling in a drowning, or *baptism* for a priest, while this is far more unlikely for other persons.

2. Multiplicity of senses has been typically addressed in an ad hoc way at the lexical level. For sentences and texts, it has either been ignored or specialized into comprehension of figurative language (*Ibid.*). We still lack a unified theory for this pervasive facet of language.

3. We agree with Frawley (1992) that "linguistic meaning *precedes* and *enters into* a context of use because speakers bring this meaning with them, in their heads, into the contexts of communication." Thus, in light of examples 3 and 4, we need to separate between a) what is believed to be the acontextual "norm" or default (Franks, 1995) for a concept and b) the contextual interpretation of it.

To address these issues, we argue below in favor of a two-tiered approach to the representation of concepts. This section, though not detailed, will stress the fundamental aspects of our proposal.

3.1 Two-Tiered Concept Structure

We structure concepts in two tiers: the *property* tier and the *extensional* one. The former refers to what the system knows, or more precisely, remembers at a given point in time (given the complexities and limitations of memory retrieval). The latter, formed of a number of simple methods, is responsible for instantiating concepts in context.

A concept is generally defined by its properties. In connectionist models, properties are usually manifested as micro-features and serve to structure the conceptual space by associating concepts with shared properties (Waltz & Pollack, 1985; Bookman, 1994). Properties in our model share the same functionality, but differ from micro-features in two ways: a) there is no fixed, universal set of properties, and b) a property may also bear information on the *use* of a concept.

For each concept, the property tier includes two parts: a default definition[1] and traces (or exemplars) of how the concept has been used in the past. The default definition consists of a set of context-independent, high-diagnostic properties, and indications of how the concept is typically used by people[2] (see Barsalou,1982). The second component of the property tier represents the reader's recollection of the concept in previously seen contexts. For example, *milk* is defined as a liquid with features distinguishing it from other liquids, plus some associated to common sense knowledge such as its typical sources and being drinkable and healthy. As for exemplars, depending on the reader's past readings, *milk* may be remembered in contexts like being an ingredient of ice-

[1] This default definition is different from a prototype and does not represent 'typical' features of the concept. For a detailed discussion, see Franks(1995).
[2] Possibly expressed in terms of simple examples.

cream and an additive to tea, and perhaps in the expression "land of honey and milk". Properties are prioritized on the basis of their centrality and diagnosticity. This classification is by no means fixed, and thus, a frequently visited peripheral property may become central.

The enpansion methods in the second tier are responsible for properly instantiating their associated concepts in context. The process is governed by the first tier and the context. More precisely, each method is coupled with an exemplar. A method investigates the applicability of its exemplar to the current context. This is achieved by having a method propose extensions, inspired by its exemplar, to the corresponding concepts in the input (see examples in section 3.3). Such proposals usually come into three forms: feature promotion, value alteration, and feature introduction. If a suggested expansion does not raise any incompatibility (see next section for more detail), then it is carried out on the corresponding concept and the method continues with the remaining components of its exemplar. Thus, such concept instantiations *gradually* construct the final interpretation(s).

3.2 Comprehension Process

The comprehension process is carried out through the concurrent execution of expansion methods. Once a context is presented to it, the system first forms a category for each concept (word) in the input. These categories are formed only on the basis of their default senses. Concepts in these categories contain exemplars of their previous usages. Coupled with each exemplar is a method which individually attempts to map the input to its exemplar. This is possible only if the method succeeds in expanding all the concepts (words) in the input to corresponding ones in its exemplar. Each concept therefore, receives some suggested expansions, and to verify them, cues memory with a combination of its default attributes and the suggested changes. The primed conceptual instances (if any) will form a new category, which in turn acts as an expansional model for the concept. This process is sketched out below:

Lexical Priming For each word in the input context, the system's memory is queried. This will lead to the activation of all previously seen contexts which contain the given word or closely related words. For example, if *boy* is the current word, not only contexts including it, but those containing concepts such as *man*, *girl*, and *colt* (a graded category of objects sharing features with *boy*) will also be (more or less) activated.

Contextual Expansions Methods associated with the exemplars individually examine the possibility of reusing them.

Compatibility Check To verify how expansions suggested by a method are compatible with a concept, the memory is primed with the default sense plus the suggested modification(s). If successful (through the activation of some previously seen usages in the memory), the concept will be expanded to these usages (see compatibility of *fish* with being agressive in Example1).

Settlement Expansion methods, working in parallel, attempt to interpret the input by instantiating its constituents to corresponding ones in their exemplars. The process continues until each method is either complete or fails (due to inconsistency).

We are currently working to formalize the interactions between concepts and possible expansions.

3.3 Examples

Example 1 *The fish attacked the swimmer*.

Anderson at al. (1976) have found that this sentence is likely to lead to encoding of *fish* in terms of *shark*. We further suggest that *swimmer* similarly will be remembered as *victim*, rather than, for example, as *sportsman*.

Assume the system's memory holds the following exemplar sentences :

S1. A passerby was attacked by the gang.
S2. The fish jumped out of the jar.
S3. Mary saw the swimmers in the pool.
S4. The hikers later on attacked the western face.

From S1, or the default sense, one possible extension path initiates for *attack*, proposing [aggressive attitude] to *fish* and [being victim] to *swimmer*. Another path simultaneously attempts to extend *attack* to another sense as it is available by S4. *Fish* and *swimmer* also individually attempt to expand themselves, by offering "jump" and "see" to *attack*, "jar" to *swimmer*, and "Mary" to *fish* (via S2, and S3).

Each path verifies how compatible its proposals are with the relevant concepts (see below for more details). Extensions motivated by S2, S3, and S4, clearly all fail due to the incompatibilities their proposals cause. However, *fish* could be extended to be aggressive, as *swimmer* can potentially be a victim. However, these pieces of information need not be directly encoded. Suppose the system has seen the following sentences as well:

S5. Sharks slaughtered the ship's survivors.
S6. The accident injured two travelers.

As a result, *shark* and *travelers* are remembered in association with aggression and being hurt respectively.

To verify the compatibility of *fish* and *swimmer* with the suggested properties, the system cues the memory with [aggressive fish] and [swimmer being victim]. S5 provides a strong match for the aggressive fish. And because swimmer and traveler both share being human, S6 also offers a viable match for the harmed swimmer.

Therefore, S1 succeeds in disambiguating *attack* in the input sentence to its sense. This in turn, will classify *fish*, *attack*, and *swimmer* properly[3].

Example 2 This example demonstrates the contextual sense generation capacity offered by our model. The system's repository contains the following sentences:

S7. John studies physics in university.
S8. John read the newspaper.
S9. The jury announced the verdict.

Input 1 : *The judge read the decree.*
For simplicity, we only consider two expansions, one motivated by S8, which attempts to classify *read* as [interpreting written words], and one suggested for *decree* and initiated by S9. The latter is possible if we assume *decree* and *verdict* overlap in their definitions. Most likely, S9 gives a better match than S8, and consequently *read* is classified with *announce*, and *Judge* is assimilated (Rastier, 1991) with *jury* .

Input 2: *The judge read the book.*
Unlike the previous case, *verdict* and *book* don't overlap and, most likely, the input matches S8.

Input 3 : *Mary reads journalism.*
Due to the existing match between *physics* and *journalism*, the input will be mapped to S7.

Example 3 is *John painted the canvas* .

This example illustrates the memory-dependent nature of the model, as well as learning, and the possibility of multiple interpretations. Suppose the system is aware of two different senses for *painting*, as provided in:

S10. Da Vinci painted the Mona Lisa.
S11. John painted the wall.

Assume *canvas* is known only as strong, coarse cloth. Now, since mapping *canvas* to *wall* is more likely than to *Mona Lisa*, *painting* in this example will favor S11 and be grouped together with *to color*. Considering matchs may be incorrect, we suggest a semi-automatic learning process whereby the user can correct a classification. This new evidence will be remembered and used in future.

Now consider a reader aware of both senses of *canvas*, yet seen only S10 (unaware of canvas-painting relation). Two coherent interpretations might be constructed, drawing on a piece of cloth (e.g. as a tent canvas), and creating an art work.

Example 4 *A dime is a dime.*

This example from Rastier (1991) displays the need for a default sense. As Tversky pointed out (Ortony, 1979), in similarity statements, the subject is usually a specialization of the object. This information could be attributed to *be* as part of one of its expansional methods, restricting the topic concept only to its default property (i.e., to the first part of the property tier).

4 RELATED WORK

Word-oriented distributed language comprehension was first introduced in WordExpert (Small, 1980). IDIoT (Corriveau, 1995) takes a similar computational approach. However, they both require extensive hand-coded knowledge.

Because it emphasizes contextuality and flexibility of knowledge, as well as uncanonical learning, Mnemosyne (Haase, 1991) is closest in spirit to our model. However, besides the lack of relativity for categorization in Mnemosyne, the models differ in the approaches they take to address their shared goals. Mnemosyne relies on costly indexing and matching operations, while in our model, relying on associative memory, priming substitutes for both. On the other hand, the property set, even if not fixed, imposes constraints to our model, while Mnemosyne enjoys its holistic approach to contexts.

Our structure for concept is inspired by Michalski's distribution of meaning (1989). Michalski, on the basis of cognitive parsimony, attributes two classes of information to each concept: a base concept representation (BCR) and an Inferential Concept Interpretation (ICI). Barsalou (1982) also divides the properties assigned to a concept into two classes: context-dependent and context independent.

Finally our computational framework resembles structured connectionist networks (e.g. Waltz & Pollack, 1985; Bookman, 1994). It is, however, distinct in two ways. First, the inferential paths are dynamic and learnable rather than static and hand-coded. Second, the

[3] Exemplars such as *Serbs attacked the village* in this case require more sophisticated extensions. We are currently investigating metaphoric and metonymic mapping (Lakoff & Johnson 1980, Lakoff, 1987). For instance, RESIDENCE STANDS FOR ITS RESIDENT metonymy allows a match between *village* and *victims*.

instantiations of concepts are also dynamic in the sense that they may vary from the default definition as a result of contextual interactions.

5 CONCLUSION AND FUTURE WORK

This work has been motivated by our conviction of the need for a 'deeper' account of concept meaning. The proposed model addresses this issue using a categorization framework. Because of its two-tiered concept structure and its pervasive use of exemplars, the model is characterized by its impoverished initial definitions, and its incremental learning strategy. Moreover, the distributed autonomous nature of the extensional tier allows multiple interpretations to be constructed.

Grounding comprehension in relative categorization opens the door to interesting issues that are usually ignored by existing NLP systems. In particular, our model addresses concept formation, multiple sense generation, and treatment of unknown words.

One interesting group of extensions to be investigated concerns conventional metaphors and metonymies as described by Lakoff and Johnson (1980), and Lakoff(1987). Using Ortony's salience imbalance account (1979), we can envision presenting such metaphors as input (e.g. ARGUMENT IS WAR) to the system and let it extract the proper source-target mappings.

Another topic for future research considers the impact of inputs on the default sense, that is, the way in which the accumulation of evidence may alter the default sense of a concept.

REFERENCES

Ahn, W. and Medin, D.L. (1992) *A Two-Stage Model of Category Construction*, Cognitive Science, 16, 81-121.

Anderson, R.C., et. al. (1976) *Instantiation of General Terms*, Journal of Verbal Learning and Verbal Behavior, 15, 667-679.

Barsalou, L. W. (1982) *Context-independent and Context-dependent Information in Concepts*, Memory & Cognition, 10(11), 82-93.

Bookman, L. (1994) *Trajectories Through Knowledge Space*, Amsterdam: Kluwer Academic Publishers.

Corriveau, J-P. (1995) *Time Constrained-Memory*, Hillsdale, NJ: Lawrence Erlbaum Associates.

Franks, B. (1995) *Sense Generation: A "Quasi-Classical" Approach to Concepts and Concepts Combination*, Cognitive Science, 19, 441-505.

Frawley, W. (1992) *Linguistic Semantics*, Hillsdale, NJ: Lawrence Erlbaum Associates.

Haase, K. B. (1991] *Making Clouds from Cement: Building Abstractions out of Concrete Examples*. In Proceedings of the US-Japan Workshop on Integrated Comprehension and Generation in Perceptually Grounded Environments.

Lakoff, G. (1987) *Women, Fire and Dangerous Things: What Categories Reveal about the Mind*, Chicago: The University of Chicago Press.

Lakoff, G. and Johnson, M. (1980) *Metaphors We Live by*, Chicago: The University of Chicago Press.

Medin D. L. and Goldstone, R. L. (1990) *Concepts*, In Eysenck, M.W(Ed.) The Blackwell Dictionary of Cognitive Psychology, Cambridge, MA: Basil Blackwell Ltd.

Michalski, R.S. (1989) *Two-tiered concept meaning, inferential matching and conceptual cohesiveness*, In Vosniadou, S. & Ortony(Eds.) Similarity and Analogical Reasoning, A., New York: Cambridge University Press.

Nosofsky, R.M. et. al. (1994) *Rule-Plus-Exception Model of Classification Learning*, Psychological Review, 101(1), 53-79.

Oden, G.C. (1987) *Concepts, Knowledge, and Thought*, Annual Review of Psychology, 38: 203-227.

Ortony, A. (1979) *Beyond Literal Similarity*, Psychological review, 86(3), 161-180.

Rastier, F. (1991) *Science et Recherches Cognitives*, Presses Universitaires de France.

Small, S.L. (1980) *Word Expert Parsing: A Theory of Distributed Word-Based Natural Language Understanding*, Ph.D. Dissertation, Department of Computer Science, University of Maryland.

Smith, E.E. and Sloman, S.A. (1994) *Similarity versus Rule-Based Categorization*, Memory and Cognition, 1994, 22(4), 377-386.

Waltz, D.L. and Pollack, J.B. (1985) *Massively Parallel Parsing: A Strongly Interactive Model of Natural Language Interpretation*, Cognitive Science, 9, 51-74.

Towards Anytime Constraint Satisfaction

Hans W. Guesgen*
Computer Science Department, University of Auckland
Private Bag 92019, Auckland, New Zealand
hans@cs.auckland.ac.nz

Abstract

Anytime algorithms play a significant role in solving real world problems, as these problems often require a solution within limited time (for example, planning the path of a robot to retrieve explosive materials from a site or diagnosing the injuries of a human after an accident). A class of algorithms that has received more and more attention over recent years is the class of constraint satisfaction algorithms, i.e., algorithms that solve problems stated as a set of constraints over a set of variables. In this paper, we look at some constraint satisfaction algorithms and apply the concept of anytime algorithms to these algorithms.

INTRODUCTION AND MOTIVATION

An important issue in the area of knowledge-based systems has been the development of efficient methods to reason about the knowledge represented by the system. Constraints are one way to represent knowledge, and constraint satisfaction algorithms one way to reason about that knowledge. A constraint satisfaction algorithm is a search algorithm for problems that can be stated in the following way: Given a set of variables over some finite domains and constraints on subset of these variables, find values for the variables such that all constraints are satisfied.

Constraint satisfaction has been applied successfully in various areas, for example:

- Computer vision [19]
- Circuit analysis [17]
- Planning experiments in molecular biology [18]
- Job-shop scheduling [5]
- Diagnosis [2, 3, 7]
- Temporal and spatial reasoning [1, 6, 9, 10, 15]
- Logic programming [4, 12]

Unfortunately, constraint satisfaction in general is NP-complete. Thus it is very unlikely that any deterministic complete constraint satisfaction algorithm has polynomial worst-case time complexity. On the other hand, exponential time complexity is unacceptable for most knowledge-based systems. We would rather accept an approximation of the solution computed in polynomial time than a solution of the problem computed in exponential time. More than that, we often require an algorithm to produce some result in a predefined period of time.

An algorithm that determines or approximates the solution of a problem in a given time period is called an anytime algorithm. Usually, the quality of its result gradually improves as available runtime increases. Anytime algorithms can be classified as follows:

1. If the algorithm knows in advance how much computation time is available for computing a result, the algorithm is called an *informed* anytime algorithm. In the context of AI planning, the notion of an informed anytime algorithm is remotely related to the notion of a contract algorithm [11, 20].
2. If the algorithm doesn't know in advance how much computation time is available for computing a result, the algorithm is called an *uninformed* anytime algorithm.

We will formalize these notions in the following.

*The author performed part of this work while on leave at the International Computer Science Institute (ICSI), Berkeley, California, where he has been supported by the University of Auckland Research Fund under the grant number A18/XXXXX/62090/F3414025.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/20 ©1996 FLAIRS

FORMALIZATION OF ANYTIME CONSTRAINT SATISFACTION

We will start the formalization with standard definitions for constraints, constraint networks, etc. as they can be found in the constraint literature. For a more elaborate introduction of these concepts, see [8] or similar work.

The basis of our formal framework are the notions of constraints and constraint networks. Informally, a constraint network is a set of constraints interconnected by shared variables, each constraint restricting the values that may be assigned to its variables. Formally, constraints and constraint networks can be defined as follows:

Definition 1 (Constraint)
A k-ary constraint C consists of variables $V_1, \ldots, V_k$ over domains $D_1, \ldots, D_k$ and a decidable relation $R \subseteq D_1 \times \cdots \times D_k$, i.e., a k-ary relation the i-th place of which is symbolized by V_i.

Definition 2 (Constraint Network)
A constraint network N is given by a set of constraints with shared variables. This means a constraint network on variables $V_1, \ldots, V_m$ consists of constraints $C_1, \ldots, C_n$, the variables of each C_i being a subset of $V_1, \ldots, V_m$.

We consider a constraint to be satisfied if the assignment of values to its variables is admissible with respect to the constraint relation. If all constraints in the network are satisfied, the assignment of values to the variables in the network is called a solution. The problem is to find such an assignment. Formally:

Definition 3 (Satisfaction)
A tuple $(d_1, \ldots, d_m) \in D_1 \times \cdots \times D_m$ satisfies a constraint C_i if the subsequence of $(d_1, \ldots, d_m)$ that corresponds to the variables of C_i is an element of the relation of C_i.

Definition 4 (Solution)
A solution of a constraint network that is given by constraints $C_1, \ldots, C_n$ on variables $V_1, \ldots, V_m$ over domains $D_1, \ldots, D_m$ is a tuple $(d_1, \ldots, d_m) \in D_1 \times \cdots \times D_m$ that satisfies $C_1, \ldots, C_n$.

Definition 5 (CSP)
The task of finding one, several, or all solutions of a constraint network is called constraint satisfaction problem (CSP).

There are various algorithms that can be used to either solve a given CSP or to transform it into an equivalent but easier-to-solve CSP. (In this context, a CSP is equivalent to another CSP if their solution spaces are identical.) Since we want to look at some of these algorithms from the viewpoint of anytime algorithms, we have to introduce notions that help us to assess the result and also the intermediate states of the algorithms.

Let us assume that $\mathcal{A}$ is a constraint satisfaction algorithm to be applied to a constraint network N. Instead of running the algorithm until it terminates, we interrupt it from time to time and look at the intermediate results of the algorithm. For example, if $\mathcal{A}$ is a backtracking algorithm [16], the intermediate result is the current assignment of values to (some of) the variables; if $\mathcal{A}$ is a filtering algorithm like Waltz filtering [19], the intermediate result is the current assignment of restricted domains to the variables; and so on.

Generally speaking, the intermediate results of an algorithm $\mathcal{A}$ applied to a constraint network N can be extracted directly from the states which N assumes during the application of $\mathcal{A}$. These states are usually defined by the data structures that are used by $\mathcal{A}$ and thus differ from algorithm to algorithm. In the following, $\mathcal{A}(N)$ denotes the set of states that N assumes when A is applied to N, and $\mathcal{A}(N)_{|t}$ the state of N at time t.

To compare different states of a constraint network with each other, we need some measure which assesses these states:

Definition 6 (State Measure)
Let N be a constraint network and A a constraint satisfaction algorithm, then a state measure $\mu : \mathcal{A}(N) \rightarrow [0, 1]$ for N with respect to A is a function from $\mathcal{A}(N)$ to the interval of real numbers between 0 and 1.

Based on the above definition, we can now define the notion of an anytime constraint satisfaction algorithm:

Definition 7 (Anytime Algorithm)
Given a constraint satisfaction algorithm $\mathcal{A}$. Let N be a constraint network and μ a state measure of N with respect to A. Then $\mathcal{A}$ is an anytime constraint satisfaction algorithm if:

$$\forall t, t' \in [0, \infty] : t \leq t' \Longrightarrow \mu(\mathcal{A}(N)_{|t}) \leq \mu(\mathcal{A}(N)_{|t'})$$

Furthermore, $\mathcal{A}$ is an informed anytime constraint satisfaction algorithm, if t and t' are input of $\mathcal{A}$:

$$\forall t, t' \in [0, \infty] : t \leq t' \Longrightarrow \mu(\mathcal{A}(N, t)_{|t}) \leq \mu(\mathcal{A}(N, t')_{|t'})$$

Otherwise, $\mathcal{A}$ is called an uninformed anytime constraint satisfaction algorithm.

Note that in the case of an informed constraint satisfaction algorithm, it may be possible that some successor states are worse than its predecessors during

the run of the algorithm, i.e., the following may hold for an informed constraint satisfaction algorithm:

$$\exists t, t', t'' \in [0, \infty]: \\ t \leq t' \wedge \mu(\mathcal{A}(N, t'')_{|t}) > \mu(\mathcal{A}(N, t'')_{|t'})$$

It is only required that, given two different contract times t and t' with $t \leq t'$, the result of applying $\mathcal{A}$ to N and t' must be at least as good as the result of applying $\mathcal{A}$ to N and t.

Obviously, not every constraint satisfaction algorithm is an anytime algorithm. Backtracking, for example, shows a very bad anytime behavior. On the other hand, there are some constraint satisfaction algorithms with perfect anytime behavior (i.e., that are anytime constraint satisfaction algorithms according to the above definition) or at least with good-on-average anytime behavior.

In the following, we will restrict our discussion to uninformed anytime constraint satisfaction algorithms. We will first introduce a class of constraint satisfaction algorithms that satisfy the condition of an anytime algorithm. We will then look at some algorithms that don't show anytime behavior but can be converted into anytime algorithms.

DISCUSSION OF CONSTRAINT SATISFACTION ALGORITHMS

There are constraint satisfaction algorithms that show perfect anytime behavior. Among these algorithms are filtering algorithms like the Waltz algorithm [19] and the arc and path consistency algorithms [13]. As a representative of the filtering algorithms, we will discuss here the filtering algorithm that was introduced in [8]. The algorithm is shown in Figure 1.

The idea of a filtering algorithm is to iterate through the constraints of the network, comparing each constraint relation with the current domains of the variables and deleting incompatible values from the domains of the variables or the relations of the constraints. When no more values can be deleted from the domains, respectively the relations, the list of constraints to iterate through finally gets empty and the algorithm terminates.

The control flow of the algorithm is influenced by two components:

1. M, the list of constraints to iterate through.
2. $D_1, \ldots, D_m$, the domains of the variables of N.

Since M gets finally empty, it doesn't contribute to the result of the algorithm, and thus we can neglect it. $D_1, \ldots, D_m$, on the other hand, have an impact on the result of the algorithm; actually, they are the result. Thus we use $D_1, \ldots, D_m$ to represent the states that the algorithm assumes when applied to N.

Given a constraint network N.

1. Initialize M with the constraints of N.
2. While $M \neq \emptyset$ do:
   (a) Select a constraint $C \in M$. Let $D_1, \ldots, D_k$ be the domains of the variables of C.
   (b) Remove C from M.
   (c) For each D_i, $i \in \{1, \ldots, k\}$, do:
      i. Delete each value d_i from D_i for which holds: There isn't

$$d_1 \in D_1 \\ d_2 \in D_2 \\ \vdots \\ d_{i-1} \in D_{i-1} \\ d_{i+1} \in D_{i+1} \\ \vdots \\ d_k \in D_k$$

      such that $(d_1, \ldots, d_k)$ is an element of the relation of C.
      ii. If D_i has been changed, then add to M all other constraints that share a variable with C.

Figure 1: A filtering algorithm for constraint networks.

Since the aim of filtering algorithms is to make the domains or the relations tighter (and thereby restricting the search space), an adequate state measure for a constraint network N with respect to filtering algorithms is a measure that takes into account the number of values in the domains of the variables, respectively the number of elements in the constraint relations. There are several ways to do so, one of which is to calculate the total number of values in the variable domains plus the total number of elements in the constraint relations:[1]

$$\mu(\mathcal{A}(N)_{|t}) = \sum_{i=1}^{m} |D_{i\,|t}| + \sum_{j=1}^{n} |C_{j\,|t}|$$

During the execution of a filtering algorithm, values are deleted from the domains of the variables or elements from the relations of the constraints.[2] As a

[1] $|D_{i\,|t}|$ denotes the number of values in the domain D_i at time t and $|C_{i\,|t}|$ the number of elements in the relation of the constraint C_i at time t.

[2] The algorithm shown in Figure 1 deletes values only from the domains of the variables but not from the relations of the constraints. An example of a filtering algorithm that does both can be found in [8].

result, neither the number of values in the domains nor the number of elements in the constraint relations increases over time, and with that the total number of values and elements doesn't increase either, which means:

$$\forall t, t' \in [0, \infty] : t \leq t' \Longrightarrow$$
$$\sum_{i=1}^{m} |D_{i|t}| + \sum_{j=1}^{n} |C_{j|t}| \leq \sum_{i=1}^{m} |D_{i|t'}| + \sum_{j=1}^{n} |C_{j|t'}|$$

This is equivalent to the following (which is the property that must hold for anytime constraint satisfaction algorithms):

$$\forall t, t' \in [0, \infty] : t \leq t' \Longrightarrow \mu(\mathcal{A}(N)_{|t}) \leq \mu(\mathcal{A}(N)_{|t'})$$

Although filtering algorithms have become a substantial part of constraint satisfaction research, they are not the only algorithms research is focussed on in this area, the reason being that filtering algorithms in general compute only an approximation of the solution of a given constraint satisfaction problem rather than the solution itself. Backtracking, on the other hand, does compute a solution (if one exists). However, backtracking algorithms in general are *not* anytime constraint satisfaction algorithms.

The best we can hope for in the case of a backtracking algorithm is a behavior that is close to that of an anytime constraint satisfaction algorithm. It has turned out that some backtracking algorithms are closer to anytime behavior than others. We performed empirical studies on a variety of backtracking approaches (see [16] for information on these approaches) and measured the quality of the intermediate results.

In particular, we performed our experiments with the algorithms shown in Figure 2. Beyond that, we tested simple backtracking and backjumping with various variable and value ordering heuristics.[3] As state measure for the algorithms, we used the number of correctly instantiated variables, i.e.:

$$\mu(\mathcal{A}(N)_{|t}) = |S_{|t}|$$

where $S_{|t}$ denotes the set of variable instantiations at time t.

As expected, none of the above algorithms are anytime constraint satisfaction algorithms, the reason being that any backtracking algorithm retracts values from time to time, which causes the algorithm to assume a state that is worse than the previous state. We can overcome this problem by caching the best instantiation so far and returning this instantiation when the algorithm is interrupted.

From this point of view, every backtracking algorithm can be converted into an anytime algorithm. However, the question that arises then is: How fast do

[3] We used the constraint satisfaction system CSP [14] for our experiments, which provides all the algorithms mentioned here.

1. Simple backtracking (BT)
2. Backjumping (BJ)
3. Conflict-directed backjumping (CBJ)
4. Graph-based backjumping (GBJ)
5. Backmarking with simple backtracking (BM)
6. Backmarking with backjumping (BM_BJ)
7. Backmarking with conflict-directed backjumping (BM_CBJ)
8. Backmarking with graph-based backjumping (BM_GBJ)
9. Forward checking with simple backtracking (FC)
10. Forward checking with backjumping (FC_BJ)
11. Forward checking with conflict-directed backjumping (FC_CBJ)
12. Forward checking with graph-based backjumping (FC_GBJ)
13. Full arc consistency forward checking with simple backtracking (FCarc)
14. Full path consistency forward checking with simple backtracking (FCpath)

Figure 2: Various backtracking algorithms for constraint satisfaction.

the above algorithms make progress, i.e., how quickly does the quality of the states improve over time? We found that the more informed an algorithm is, the faster the improvement over time. This result is in accordance with the results in [16] regarding the efficiency of various backtracking algorithms.

Another way of getting an indication for how fast an algorithm makes progress is to look at the following:

1. How often does the algorithm assume a state that is worse than the previous one?
2. What is the standard deviation of the intermediate states?

Table 1 summarizes our findings for the various backtracking approaches. We checked 30 intermediate states for each algorithm and calculated the standard deviation of the number of correctly instantiated variables in each state (see 'Standard deviation' in Table 1). Beyond that, we counted how many of the 30 intermediate states were worse than previous states (see 'Degradations' in Table 1).

The table shows that BM_GBJ is the most likely can-

Algorithm	Standard deviation	Degradations
BT	1.26	12
BJ	1.65	13
CBJ	1.84	9
GBJ	1.14	9
BM	1.11	13
BM_BJ	2.16	10
BM_CBJ	1.99	12
BM_GBJ	1.04	7
FC	1.20	11
FC_BJ	1.43	11
FC_CBJ	1.96	14
FC_GBJ	2.08	12

Table 1:

didate for an algorithm that makes good progress over time and therefore is worthwhile being converted into an anytime algorithm. Only 7 out of the 30 intermediate states in our experiment were states that were worse than previous states.

SUMMARY

In this paper, we applied the concept of anytime algorithms to constraint satisfaction. We introduced the concept of anytime constraint satisfaction, provided a formalization, and discussed filtering algorithms with respect to their anytime behavior. We also showed that backtracking algorithms in general are not anytime algorithms, but can be converted into anytime constraint satisfaction algorithms very easily.

References

[1] J.F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983.

[2] R. Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24:347–410, 1984.

[3] J. de Kleer and B.C. Williams. Diagnosing multiple faults. In *Proc. AAAI-86*, pages 132–139, Philadelphia, Pennsylvania, 1986.

[4] M. Dincbas, H. Simonis, and P. van Hentenryck. Extending equation solving and constraint handling in logic programming. In *Proc. Colloquium on Resolution of Equations in Algebraic Structures*, Austin, Texas, 1987.

[5] M.S. Fox, B. Allen, and G. Strohm. Job-shop scheduling: An investigation in constraint-directed reasoning. In *Proc. AAAI-82*, pages 155–158, Pittsburgh, Pennsylvania, 1982.

[6] C. Freksa. Qualitative spatial reasoning. In *Proc. Workshop RAUM*, pages 21–36, Koblenz, Germany, 1990.

[7] H. Geffner and J. Pearl. An improved constraint-propagation algorithm for diagnosis. In *Proc. IJCAI-87*, pages 1105–1111, Milan, Italy, 1987.

[8] H.W. Guesgen and J. Hertzberg. *A Perspective of Constraint-Based Reasoning*. Lecture Notes in Artificial Intelligence 597. Springer, Berlin, Germany, 1992.

[9] H.W. Guesgen and J. Hertzberg. A constraint-based approach to spatiotemporal reasoning. *Applied Intelligence (Special Issue on Applications of Temporal Models)*, 3:71–90, 1993.

[10] D. Hernández. *Qualitative Representation of Spatial Knowledge*. Lecture Notes in Artificial Intelligence 804. Springer, Berlin, Germany, 1994.

[11] J. Hertzberg. Anytime-Algorithmen. *KI*, 1/95:40, 1995.

[12] J. Jaffar and J.L. Lassez. Constraint logic programming. In *Conference Record of the 14 th Annual ACM Symposium on Principles of Programming Languages*, pages 111–119, Munich, Germany, 1987.

[13] A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[14] D. Manchak and P. van Beek. A c-library of constraint satisfaction techniques. Technical report, Available by anonymous ftp from ftp.cs.ualberta.ca:pub/ai/csp, 1994.

[15] A. Mukerjee and G. Joe. A qualitative model for space. In *Proc. AAAI-90*, pages 721–727, Boston, Massachusetts, 1990.

[16] P. Prosser. Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 9:268–299, 1993.

[17] R.M. Stallman and G.J. Sussman. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence*, 9:135–196, 1977.

[18] M. Stefik. Planning with constraints (MOLGEN: Part 1). *Artificial Intelligence*, 16:111–140, 1981.

[19] D.L. Waltz. Generating semantic descriptions from drawings of scenes with shadows. Technical Report AI-TR-271, MIT, Cambridge, Massachusetts, 1972.

[20] S. Zilberstein and S.J. Russell. Efficient resource-bounded reasoning in at-ralph. In *Artificial Intelligence Planning Systems: Proc. First International Conference*, pages 260–266, College Park, Maryland, 1992.

INCREMENTAL TEMPORAL CONSTRAINT PROPAGATION

Hartmut Noltemeier[1] and Georg Schmitt[2]

[1] University of Wuerzburg, <noltemei@informatik.uni-wuerzburg.de>
[2] *Deutsche Luft- und Raumfahrtsgesellschaft*, Oberpfaffenhofen, Germany

Abstract

This paper presents the problem of adding intervals and constraints to a temporal constraint network and its efficient propagation. We compare the classic algorithm of Allen with several incremental techniques. All these algorithms were implemented and their runtime complexity was measured in certain scenarios. One of the results was, that the incremental algorithms led to improved runtime complexities which we confirmed by a more rigorous analysis. For incremental algorithms it is shown, that they result in local consistency. Moreover, some selection rules are discussed and resulting complexities are shown to be advantageous. Measurements in some other temporal calculi again confirmed our improvements.

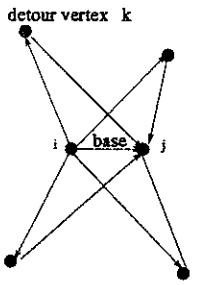
1 INTRODUCTION

J.F. Allen [1] describes a calculus, in which temporal dependencies are defined by qualitative relations between intervals. Besides the algebra an algorithm is introduced additionally, which establishes local consistency in constraint networks. This is done by propagating the constraints by means of transitive consequences. In this paper we consider a constraint network as a graph. The vertices (of a set V) represent intervals and the constraint between the start and the end vertex of an (directed) arc is given by a disjunction of Allen's basic relations. Allen additionally showed that his algorithm doesn't reveal all cases of *global* inconsistency. Vilain, Kautz and van Beek [6] proved, that the determination of inconsistent scenarios is NP-hard. Nevertheless Allen's algorithm is known as an efficient procedure to constrain a temporal network. It is particular meaningful in certain subalgebras (compare [6], [5]), in which it reveals all globally inconsistent scenarios. In the next section we give an informal introduction to Allen's algorithm to enable an easy understanding. We refer to a variant of Allen's algorithm [4], named $3C$. Skipping the initialization phase it looks like follows:

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/25 ©1996 FLAIRS

```
procedure propagate()
While queue is not empty Do
{ get next (i,j) from the queue
   /* propagation begins here */
   For ((k ∈ V) ∧ k ≠ i ∧ k ≠ j)Do
   { temp ← R(i,k) ∩ (R(i,j) ∘ R(j,k))
      If temp = null Then signal contradiction and Exit
      If R(i,k) ≠ temp Then
      { place (i,k) on queue
         R(i,k) ← temp }
      temp ← R(k,j) ∩ (R(k,i) ∘ R(i,j))
      If temp = null Then signal contradiction and Exit
      If R(k,j) ≠ temp Then
      { place (k,j) on queue
         R(k,j) ← temp }
   }
}
```

$R(x,y)$ is the value (the constraint) of an arc (x,y); $x,y \in V$. The procedure incorporates a *queue*. In the initialization phase of $3C$ all arcs (which do not have inverse arcs) with less than 13 relations are pushed onto the queue. Unlike Allen's version, updates of single constraints are not possible. The second difference is, that in the case of an empty relation as constraint inconsistency is reported and the algorithm immediately terminates. If the queue is empty the while-loop finishes. As a first step an element is popped from the queue. If the constraint graph consists of $|V| = n$ vertices (intervals), then the loop iteratively runs through $n-2$ vertices (*detour vertices*). This way all possible triangles (i,k,j) are considered for the base arc (i,j). Therefore we will speak of *triangle-propagation* with base (i,j) and detour vertex k (*transitive propagation*).



The for-loop does the task of constraining the disjunctions. More precisely we attempt to cross out relations at arc (i,k) first, afterwards those at arc (k,j). For this purpose the composition operator $\circ$ (the procedure *constraints* in [1]) is used.

In both cases the value of (i, j) is an argument of this operator, because conclusions from an earlier restriction should be made. Looking on a specific arc, whose value is *constrained* - that means, the associated disjunctions does not contain all 13 basic relations - it followed that all incident arcs will be propagated, too. Finally, there is no arc which can be constrained furthermore by the arc (i, j) and its constraint. As all arcs which are constrained are pushed onto the queue, all possible arcs with all possible detour vertices are considered. Therefore it is guaranteed, that all possible *transitive* propagations are executed. New restrictions are handled in the same way as long as the queue is not empty. Thus, there is no transitive reduction left. This is the reason we have got *local consistency*. By *transitive consistency* we mean, that $3C$ did not detect any contradiction in the final network.

In the next two sections, we refer to the incremental algorithm IC, which was proposed in [4]. IC is used to add one vertex with its new constraints to a constraint graph and to propagate the enlarged network. It can be used iteratively to constrain the whole given graph. Unlike $3C$, the authors [4] supposed up to $\mathcal{O}(n^4)$ basic steps in the worst case. The following considerations improve this result. It will be shown, that IC works correctly, which means that all transitive propagations are fulfilled and ends up in transitive consistency.

2 MIXING CONSTRAINT GRAPHS

Definition: The MIXING PROBLEM consists of firstly adding the set of vertices W and an arc set S (with new constraints) to an existing constraint graph C with vertex set V and secondly of establishing local consistency of the new graph M. The new constraints describe relations between a new and an old interval or between two new intervals.

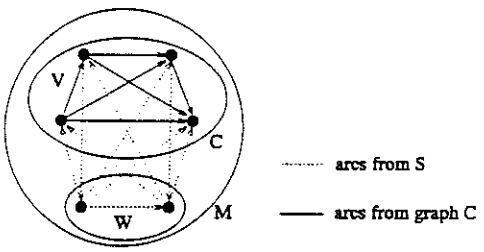


Figure 1: example for the MIXING PROBLEM

In this section the behaviour of both algorithms will be analyzed in various mixing scenarios. Suppose $n_1 = |V|$ and $n_2 = |W|$. Sequences of measurements with fixed n_1 and several values for $n_2 \leq n_1$ were made. C was propagated first by $3C$. The incremental algorithm was executed after every iterative interval update. Thus IC was invoked n_2 times. However $3C$ was executed once more again after the propagation of C to restrict the graph M. For $n_1 \ll n_2$ a lower complexity of IC was expected, because for every update of an interval with index m in the initialization phase only $\mathcal{O}(m)$ arcs are pushed onto the queue in contrast to altogether $\mathcal{O}((n_1 + n_2)^2)$ arcs in $3C$. However for $n_1 \approx n_2$ a better performance of $3C$ was expected.

We did extensive measurements with respect to the problem pointed out. The first sequence of measurements were based on consistent graphs M. For that purpose intervals were generated first and then the valid basic relation was added to every arc. Furthermore at least one up to at most 12 relations were added randomly to each arc, the number of additional relations was generated randomly, too. For both algorithms and all values of n_2. 1000 measurements were done.

For consistent scenarios, the following tables illustrate calls of the procedure *constraints*, the number of such calls directly from the initialized arcs and the number of elementary calculation steps.

$3C$						
n_2	1	3	6	9	12	15
elementary steps	2420689	2446105	3921333	6217175	8049125	10336797
constraints	5205	9900	22055	38042	60915	84341
initialization	3360	4896	7980	12144	17550	24360

IC						
n_2	1	3	6	9	12	15
elementary steps	1074599	1570635	3134194	5064793	7170779	9819225
constraints	2487	6592	17037	29719	49095	76607
initialization	840	2888	7000	12552	19760	28840

In a second sequence the graph C was created as before. The arcs of S got random values with at least two and at most 13 relations. After the propagation of C, cases of inconsistency could occur, implying that the algorithms often terminated by reporting contradictions.

In all consistent scenarios n_1 had been fixed 15. As expected IC was faster for small values of n_2. Not expected was that IC and $3C$ had fairly equal complexities for $n_2 = n_1$. The reason is, as we will prove later on, that IC has an $\mathcal{O}(n^3)$ complexity ($n = n_1 + n_2$). The tables provide other important conclusions, too. E.g. procedure *constraints* has permanently lower rates (the number of attempts performed to constrain arcs) than in $3C$. The reason why IC has a smaller *constraints*-rate than $3C$ may be twofold: First every arc which was popped from the queue, yields the propagation of $2(n-2)$ arcs. In contrast to this the incremental algorithm propagates only $2(m-2)$ arcs if the m-th arc is added. Second IC takes advantage of the fact, that the propagated graph (with $m-1$ vertices) is already constrained. In the average IC needs more steps for the propagation of an arc, and therefore produces more restrictions per triangle.

Furthermore it is interesting to know, how often arcs, which are pushed onto the queue in the initialization phase, are pushed onto the queue again in the average.

Calculation for IC: Consider phase three of IC. Let m be the index of the currently added vertex. Then, in the initialization phase, $m-1$ elements are pushed onto the queue. For all these elements *constraints* is called $2(m-2)$ times. Therefore by the adding of the m-th vertex $2(m-2)(m-1)$ propagations were executed in consequence of the propagation phase. If the constraint graph is built up from the first up to the last vertex n, then the rate is in all $\sum_{m=1}^{n}(m-1)(2m-4)$. In the mixing case n_2 vertices are added, (n_1 vertices are already existing), thus $m \in \{1, .., n_2\}$ is supposed. Therefore, the rate of *constraints*-calls is $((m+n_1)-1)(2(m+n_1)-4) = 2m^2+4mn_1+2n_1^2-6m-6n_1+4$. Altogether the number of *constraints*-calls is $\frac{2}{3}n_2^3-2n_2^2+\frac{4}{3}n_2+2n_2^2n_1+2n_1^2n_2-4n_1n_2$ ($\in \mathcal{O}((n_1+n_2)n_2^2+n_2n_1^2)$).

Calculation for $3C$: During the initialization phase, up to $\binom{n}{2} = \frac{n(n-1)}{2}$ elements are pushed onto the queue. For every arc on the queue we have $2(n-2)$ calls of *constraints*, together $n(n-1)(n-2) = n^3 - 3n^2 + 2n$ calls. In the mixing case is $n = n_1 + n_2$. Therefore the rate of calls is $\boxed{n_1^3} + 3n_1^2n_2 + 3n_1n_2^2 + n_2^3 - 3(n_1^2 + 2n_1n_2 + n_2^2) + 2n_1 + 2n_2$. Remarkably the formula of $3C$ contains n_1^3, in contrast to IC. Therefore $3C$ is handicapped if n_2 has small values. It is evident, too, that the fraction of compositions and the initialized compositions has lower values in the case of IC (except if n_2 has small values). It follows that IC repeats the propagation of arcs fewer times than $3C$. Since both algorithms result in identical networks, IC constrains the triangles stronger than $3C$ in the average. These observations will be important in the sequel (section 5).

3 COMPLEXITY AND CORRECTNESS OF IC

3.1 Complexity of IC

As already mentioned, Loganantharaj et.all. [4] supposed a runtime of $\mathcal{O}(n^3)$ (best case) and $\mathcal{O}(n^4)$ (worst case). Our measurements concerning the MIXING-PROBLEM and the following considerations show, that the worst case is $\mathcal{O}(n^3)$. In order to analyze the complexity, note that it is sufficient to consider the third phase, as in the first phase the same elements are pushed onto the queue or a subset; whereas the second phase has a complexity $\mathcal{O}(n^2)$ in any case.

Theorem 3.1 *The third phase of IC has a complexity of* $\mathcal{O}(n^3)$.

Proof: For all popped elements (not the initialized elements) at least one relation was deleted. If such an element causes no restriction, then no further arc is pushed onto the queue. This way, no more than $13\binom{n}{2} \in \mathcal{O}(n^2)$ elements can be pushed or popped. The lemma in the next subsection proves, that only $\binom{n}{2}$ initialized elements are possible. Adding the m-th vertex causes $\mathcal{O}(m) \equiv \mathcal{O}(n)$ steps in the for-loop (m goes up to n). Therefore the total complexity of the algorithm for the propagation of a constraint graph with n vertices is $\mathcal{O}(nn^2) = \mathcal{O}(n^3)$.

3.2 Correctness of IC

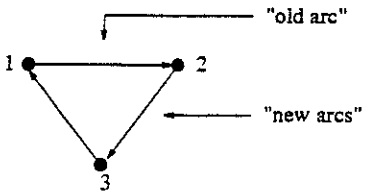
Lemma 3.1 *Consider the scenario, in which a constraint graph is constructed iteratively from the first up to the last vertex n and suppose that after each update of the m-th vertex the third phase of IC is executed. Then, all $\binom{n}{2}$ arcs are pushed onto the queue exactly once in the initialization phase.*

Proof: Let m be the vertex currently updated. Then, $m-1$ different arcs are pushed onto the queue in the initialization. More precisely, these arcs were not contained in the propagated subgraph. Referring to the whole graph with n vertices there are consequently the following number of different arcs:

$$\sum_{m=1}^{n}(m-1) = (\sum_{m=1}^{n} m) - n = \frac{n(n+1)}{2} - n = \frac{n(n-1)}{2} = \binom{n}{2}$$

Theorem 3.2 *IC detects any case of transitive inconsistency or results in local consistency otherwise.*

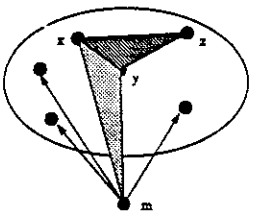
Proof: It is sufficient to consider phase three. According to lemma 3.1, each arc of the final graph is pushed exactly once onto the queue during the various initialization steps. As in $3C$, all transitive consequences are done. This is shown by the following induction; let m be the currently added vertex.
Basis of induction: $m = 3$



Here, the arcs (1,3) and (2,3) are pushed onto the queue. The calculations are:

1. (2,3) is propagated by means of $\boxed{(1,2)}$ and (1.3).
2. (1,2) is propagated by means of (2,3) and (1.3).
3. (1,3) is propagated by means of $\boxed{(1,2)}$ and (2.3).
4. (1,2) is propagated by means of (1,3) and (2,3).

The first two steps are due to the processing of element $(1,3)$, the last two steps are due to element (2,3). Considering the first and the third step, it is evident that arc $(1,2)$ propagates the "new" arcs. So, it is unnecessary to push the "old" arc onto the queue in the initialization. All possible restrictions are performed.
Induction assumption:
The constraint graph with $m-1$ vertices is locally and transitively consistent.
Induction step ($m-1 \rightarrow m$):



In the initialization phase $m-1$ new arcs are pushed onto the queue. With respect to a triangle (x,y,m) the same arguments hold as for $(1,2,3)$ in the basis of induction. Here again it is redundant to lay old arcs onto the queue, because the old subgraph with $m-1$ vertices is locally consistent due to induction. Therefore triangles like (x,y,z) need not to be propagated. After the initialization the third phase of IC works the same way as $3C$. Since all relevant arcs are propagated in the initialization phase, the constraint graph with m vertices is locally consistent and any case of transitive inconsistency is reported. Therefore, the third phase (and the whole IC algorithm) is correct.

Data Indeterminacy : A Relational Approach

Robert A. Morris
Institute for Human and Machine Cognition
University of West Florida
Pensacola, FL 32514
(904) 474-2091
rmorris@ai.uwf.edu
http://www.coginst.uwf.edu
Topic Areas: Learning, Constraint Reasoning

Abstract

Complex problem solving often requires multiple representations or models of a domain. A control mechanism is then required to switch among models. An expert at problem solving will tend to select the simplest model which contains the solution to the problem instance. Observations provide the primary input for a model-switching control mechanisms. This paper examines contexts in which observations do not carry enough information to allow a control mechanism to find a single model to switch to. In such cases, an intelligent control mechanism must devise tests in order to discriminate among competing models. These tests will be based on finding differences among the competing models. Two examples of the usefulness of this component of the control mechanism are provided: in the diagnostic realm, and in failure-driven learning.

1. Introduction

Researchers in many disciplines have debated the idea that experience does not emerge fully "constructed", but rather is constructed through a robust interpretation by the cognitive processes of the agent. This leads to the dual problems of confirmation and refutation, i.e., "the difficulty of determining the relevance of available facts to some hypothesis or problem at hand". A related problem is that of determining which of a number of competing hypotheses or theories the evidence confirms, and of ranking, or comparing theories on the basis of their degree of confirmation.

The objective of this paper is to provide a mechanism for a rigorous treatment of these issues. The framework for this discussion is the investigation of adaptive problem solving agents capable of modifying their view about the world based on observations. Adaptive problem solving is useful in diagnosing complex systems, planning or scheduling in a changing or uncertain world, and other contexts.

This paper introduces a relational framework in which to investigate computational issues in model management. To illustrate the problem of model switching, two examples are provided, one in the diagnostic realm, the other in learning about a domain. The main focus of this paper is on contexts in which observations do not carry enough information to allow a model manager to identify a unique new model which explains the new observation. In such cases, we say that the data "underdetermines" the correct model, or is indeterminate. In such cases, it is sometimes possible to discriminate among competing models by making further observations. Which observations to make depends on finding observable differences between the competing models. For relational models based on discrete domains, this amounts to performing simple operations on relations. Even models based on continuous domains can apply this technique, however, by first applying a representational transformation into a discrete domain. Hence, the proposed technique can claim a general

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/30

applicability.

In the remainder of this paper, we first introduce the relational approach to representing knowledge. Then, two contexts in which model transformations are required are introduced (section 3). The first context occurs in diagnosis, and is referred to in the literature as discrepancy-directed refinement [Weld]. This is the case where an observation in the world is not explained by the current model. The second context occurs when a planning agent also executes its plans. In this case, the agent's model of the world generates an plan which cannot be executed. In section 4, data indeterminacy is introduced into both examples, and a technique for discriminating among the competing models is outlined. The paper closes with connections to related work.

2. Preliminaries

We utilize the constraint network representation as presented in [Dechter and Pearl]. This representation views theories in terms of the logical models they induce. This allows for an abstraction from considerations that are not essential in representing differences among models, in particular, differences in the language used to express the knowledge. A constraint network is based on a representation consisting of a set of variables X and associated domains of discrete values, one for each element in X. In the examples that follow, each domain is bi-valued. It is assumed that an ordering has been imposed on the members of X; thus X_i is the ith element of the X.

An n-ary relation $r(X_1, ..., X_n)$ is a subset of the set of tuples forming the Cartesian product $d = D_1 \times ... \times D_n$ of the associated domains. Such relations interpret constraints on assignments to n-tuples of variables from X. We abbreviate the presentation of $r(X_1, ..., X_n)$ as $r_{1,...,n}$. Arbitrary tuples will sometimes be indexed by the scheme of the relation of which the tuple is a member; e.g., $(0,1)_{12}$ is in the relation r_{12}, the relation between x_1 and x_2.

A constraint network N is a set of relations. Each relation in the set is defined on a subset of variables in X. We let scheme(N) refer to a subset of the power set of variables from X that defines one of the relations in N. The term rel(N) refers to the unique relation which consists of all the solutions for N, i.e.:

$$rel(N) = \{ x = (x_1, ..., x_n) : \forall S_i \in S, \prod_{S_i}(x) \in r(S_i) \}$$

The symbol $\prod_A$ stands for the projection of a relation onto a set of variables A. A solution of a constraint network N is any tuple $(v_1,...,v_n)$ in rel(N).

Example 1. Let $N= \{r_{12}, r_{23}\}$ be a constraint network, where $r_{12}= \{(0,1),(1,0)\}$, $r_{23}= \{(0,1),(1,0)\}$, and $scheme(N)= \{\{v_1,v_2\},\{v_2,v_3\}\}$. N is thus a binary constraint network. $rel(N) = \{(0,1,0),(1,0,1)\}$.

3. Model Management in Relational Models

This section introduces two kinds of contexts for model switching. The first occurs in diagnosis and is referred to as discrepancy-driven refinement. In this context, the role of the model manager is to eliminate one or more simplifying assumptions that are inconsistent with current observations. The second example occurs in the process of executing plans, when a plan implies a false statement about the world. In this case, the model manager must find a means of revising the knowledge in a way that no longer generates the falsehood. This example can be referred to as "learning from failure".

3.1 Discrepancy-driven Refinement

Refining a model is the dual of simplifying it. Simplification involves the addition of simplifying assumptions; refinement means deleting them. Refinement is generally performed as the result of observation(s) that render the simplifying assumptions false [Weld].

Formally, refinement can be viewed as a procedure REFINE(OBS,M,S), where OBS is a set of observations, M is a model, and S is a set of assumptions underlying M. REFINE returns a pair <M',S">, where M' is a new model based on the set S' (a subset of S) of simplifying assumptions. M' and M are based on the same representation.

Example 2. Let $M= \{r_{12}, r_{23}\}$ be a constraint network, where $r_{12}= \{(0,1),(1,0)\}$ and $r_{23}= \{(0,1),(1,0)\}$. M is based on the set $S = \{v_2=1-v_1, v_3=1-v_2\}$ of simplifying assumptions. Let $OBS = \{(0,1,1)_{123}\}$. Notice that $rel(M) = \{(0,1,0),(1,0,1)\}$. OBS is a discrepancy in the sense that it is not predicted by the model. The proper refinement of M would be to delete $v_3=1-v_2$ from S, resulting in the model M' with $r_{12}= \{(0,1),(1,0)\}$ and $r_{23}= \{(0,1),(1,0),(0,0),(1,1)\}$. Note that $rel(M')= \{(0,1,0),(1,0,1),(0,1,1),(1,0,0)\}$. Hence, OBS is no longer a discrepancy.

To carry out a discrepancy-directed refinement, a model manager must, in general, generate refinements of the simpler model until one is found in which the current set of observations is no longer a discrepancy. In this example, it was obvious which of the simplifying assumptions was the source of the discrepancy, since the projection of

OBS to the variables v_2,v_3 was not in r_{23}. Hence, the simplifying assumption involving those variables had to be the source of the discrepancy. Finally, notice that there is a preference for "minimal" refinements, ones that do not make unnecessary deletions of simplifying assumptions. In the previous example, deleting all the assumptions from S would result in a model in which OBS in no longer a discrepancy, but the result is more refined (i.e., more complex) than is required for the observation.

3.2 Learning From Failure

Imagine a robot navigating through a complex of buildings, offices and corridors for the purpose of delivering mail. The robot begins with an internal map of the complex, but this map is only an approximation of the actual world. For example, it fails to identify the width of the corridors, and some are too narrow for the robot to navigate through. The robot needs to "learn from failure" the paths it can follow to deliver mail.

It is natural to characterize this and other examples of learning from failure by saying that the agent's model of the world admits more solutions to a problem than there really are. To formalize the notion of failure within the network framework, it is necessary to distinguish the generation of a solution from its execution. Intuitively, to learn from failure, a reasoner applying a constraint network N generates solutions which fail in their execution. We can use the same representation for generated and executed solutions, viz., a set of tuples, but an execution tuple will be called a **trace**. A trace of a successful execution is always a solution of N. The trace of a failed solution of assignments to a set of n variables can be of any cardinality $1 \leq i \leq n$.

Thus, learning from failure can be viewed as a procedure LFF(M,f), where M is a model and f is a trace of a failure execution. Thus, f is in rel(M). The result of applying LFF is a model M' such that f is not in rel(M'). Again, it is clear that to be effective as a model management function, LFF should apply minimal changes to M. In this case, minimality will ensure that no tuples are deleted from rel(M) that are part of solutions to the problem.

Example 3. Let N = $\{r_{12}, r_{23}\}$, where r_{12} = { (0,0), (0,1)} and r_{23} = { (0,0), (1,1), (1,0) }. Let f = $(0,1,1)_{123}$. rel(N)$_{123}$ = { (0,0,0), (0,1,1), (0,1,0)}.

This example represents the case where N generated $(0,1,1)_{123}$ as a complete solution, but failed after the last assignment was completed. The result was a trace on the relation r_{123}. The solution to this instance of the LFF problem is found by deleting (0,1,1) from rel(N)$_{123}$. This implies that (1,1) is not part of any solution involving x_2 and x_3, which means that this tuple should be deleted from r_{23}.

4. Indeterminacy of Data

It is almost a truism to say that data is often incomplete or partial. The effective handling of such data by reasoning systems is the focus of much, even most, of Artificial Intelligence. The purpose of this section is to characterize the effective handling of partial data in systems with adaptive methods for modeling the world. Briefly, partial data is undesirable in the context of model management because it may result in more than one solution to the model switching problem. Formally, partial or incomplete data means that REFINE and LFF are best characterized as relations, not functions. It will turn out that to perform effective model management in the presence of partial data, a reasoner can attempt to find observable differences among competing modifications to the current model. This process is described with respect to our example problem-solving contexts.

4.1 Indeterminacy in Discrepancy-driven Refinement

First, consider the following variation of the discrepancy-directed refinement problem.

Example 4. Let M= $\{r_{12}, r_{23}\}$ be a constraint network, where r_{12}= {(0,1),(1,0)} and r_{23}= {(0,1),(1,0)}. M is based on the set S = $\{v_2=1-v_1, v_3=1-v_2\}$ of simplifying assumptions. Let OBS = $\{(0,1)_{13}\}$.

This example differs from Example 1 only in the fact that OBS does not contain values from all the variables in scheme(M). Hence, it is partial. However, it is now the case that deleting either of the simplifying assumptions from S will explain the discrepancy. We say that the data underdetermines the proper selection for refinement.

In this example, the problem of indeterminacy can be solved by sampling the value of the missing variable v_2. This is because the candidate models for refinement (i.e., the two models that result from single deletions from S) differ with respect to v_2 when the observed variables have their corresponding values.

A more general characterization of the problem of model refinement in the presence of incomplete data is the following:

Model Testing for Refinement. Let OBS be a set of observations, and M and M' be competing models for refinement based on this set. Let V be the set of observable variables in scheme(M). OBS is thus a subset of V. Let $\sigma_{OBS}(M)$ refer to the set of tuples in M which are consistent with the observations OBS (i.e., with values instantiated to those in OBS). To discriminate between M and M', given OBS calculate

$$D = \sigma_{OBS}(M) - \sigma_{OBS}(M')$$

i.e., the set of tuples in M not in M' for values of variables in OBS set to the currently observed values. D is the set of observable differences between the two models for the current observations. To test M, observe the values of observable variables in the set V - OBS (i.e., values of observable variables not yet observed). If the observations match those in $\sigma_{OBS}(M)$, then M is confirmed; else if they match the values in $\sigma_{OBS}(M')$, then M' is confirmed.

In the previous example, if M is the refinement consisting of removing the assumption that v_2=1-v_1, and M' the refinement consisting of removing v_3=1-v_2, then $D = \sigma_{OBS}(M) = \{(0,0,1)_{123}\}$. The discriminating test between M and M' is the observation of the value of v_2. If this value is 0, then M is confirmed; if it is 1, then M' is confirmed.

An assumption being made here is that there exists observable differences between competing refinements. This is a reasonable assumption, since typically simplifying assumptions, and hence refinements, are based on observable variables. If, in the previous example, v_2 is not an observable variable, then there is no observable difference between the competing refinements. In such cases, refinement decisions are not deducible from any set of observations, and hence need to be retractable.

4.2 Indeterminacy in Learning from Failure

In Example 3, a reasoner with knowledge stored as a binary constraint network reasoned from failure to a deletion of tuple in some relation in its network. The next example illustrates the problem of indeterminacy of data for learning from failure by demonstrating that such deletions are not always possible.

Example 5. Let $N = \{r_{12}, r_{23}, r_{34}\}$, where

$$r_{12} = \begin{matrix} 1 & 1 \\ 0 & 1 \end{matrix},\ r_{23} = \begin{matrix} 1 & 0 \\ 1 & 1 \end{matrix},\ \text{and } r_{34} = \begin{matrix} 1 & 0 \\ 1 & 1 \\ 0 & 0 \\ 0 & 1 \end{matrix}.$$

Note that $rel(N)_{1234} = \begin{matrix} 1&1&0&1 \\ 1&1&1&0 \\ 1&1&1&1 \\ 0&1&0&0 \\ 0&1&0&1 \\ 0&1&1&1 \\ 0&1&1&0 \\ 1&1&0&0 \end{matrix}$. Let

$f_{1234} = 1\ 1\ 0\ 0$.

We noted that the preferred modifications to a network based on failure are those that are "minimal". This means in this example that changes to N should result in only (1,1,0,0) being deleted from rel(N). We will call such deletions **sound**. Unfortunately, the trace does not contain enough information to allow the constraint reasoner to infer the proper tuple in N to delete. For example, if the tuple (0,0) is deleted from r_{34}, this would also remove (0,1,0,0) from rel(N). But this might be a solution to the problem; hence the deletion removes more tuples than can be properly inferred from the trace.

Again, to infer the proper tuple to delete, more traces need to be generated. Notice that a set of traces collectively permits a sound deletion of some tuple t of a relation of N if and only if all ways of extending t into a solution resulted in failure (i.e., is in *f*).

Example 6. Suppose N contains a tuple $(00)_{23}$ and that $f_{1234} = \begin{matrix} 1&0&0&1 \\ 1&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \end{matrix}$. Every combination of tuples involving $(00)_{23}$ has been tried, resulting in failure. Deleting this tuple from N would therefore be sound; this tuple is not part of any solution.

Thus, one technique for inferring sound deletions would be to attempt to generate all solutions involving a tuple in some relation. If all the solutions are failures, then the original tuple can be deleted. This approach might be impractical, due to the number of traces that need to be generated, or because the intentional generation of failures is not available to the agent. An alternative solution is to store the information provided by the traces as tuples in a new relation. This has the advantage of not requiring further tests to be performed on the space of solutions. It has the potential disadvantage of slowing down the search process in future problem instances. New relations translate into more tests to be performed while generating a solution. Finding effective ways of processing

data acquired from failure is beyond the scope of this paper (see [Morris and Ford] for more discussion of this issue).

This section has briefly investigated the problem of selecting among models in the presence of partial or incomplete data. We noted that often a set of observations can be inferred from the knowledge obtained in order to select among competing models. Examples from both the diagnostic domain and the planning domain were provided.

5. Related Work

From the perspective of the learning literature, the method described here is related to a body of work which incorporates both inductive learning and learning to improving the efficiency of search. The approach proposed here is perhaps most similar to the set of methods underlying Explanation-based Learning (EBL). Researchers in EBL, for example Minton [Minton], and others [Gupta], [Chien] have emphasized the importance of failure-based learning.

From the perspective of the CSP literature, there similarity between the problems attacked here and the research directed at learning while searching for a solution to improve the search process, e.g. [Dechter], [Sadeh et al]. This work can be described informally as the process of making implicit constraints explicit when a dead end occurs in the search process. The main difference between this work and the work presented here is that in the former, failures (deadends) indicate that the search procedure, in conjunction with the consistency-enforcing scheme and other heuristics, are inadequate to deal with the problem instance at hand; in our approach, failure is rather an indication that the constraint network which represents an agent's knowledge about a problem inaccurately describes the problem.

Also of relevance to this work is the body of literature (e.g., [Kautz, et.al]) investigating the relationships between sets of formulas and their models, in particular, of converting a set of models into an efficient representation such as sets of Horn clauses. Also of relevance is recent work on learning to reason [Khardon and Roth]. Finally, the method of finding differences between competing refinements is very close to the testing method proposed by Struss [Struss] for selecting among competing diagnoses.

6. Conclusion

This paper has offered a characterization of data indeterminacy in the context of model switching. In solving complex problems, or learning in a complex and changing world, it behooves a reasoning mechanism to adapt to perceived changes in the world. These changes may be temporary, based on the aspects of a particular problem instance, or permanent, reflecting an inadequate model of the world.

7. References

[Chen] Chen, S. A., Extending Explanation-based Learning: Failure-driven Schema Refinement. *Proceedings Third IEEE Conference on Artificial Intelligence Applications,* Orlando, FL ,(1987), 106-111.

[Dechter] Dechter, R., Enhancement Schemes for Constraint Processing: Backjumping, Learning and Cutset Decomposition. *Artificial Intelligence* **41** (1989/1990) 273-312.

[Dechter and Pearl] Dechter, R. and Pearl, J. Structure Identification in Relational Data. *Journal of Artificial Intelligence,* **58,** (1992), 237-270.

[Gupta] Gupta, A., Explanation-based Failure Recovery. *Proceedings of AAAI-87,* Seattle, WA, (1987), 606-610.

[Kautz et. al] Kautz, H., Kearns, M., and Selman, B., Horn Approximations of Empirical Data. *Journal of Artificial Intelligence,* **74,** (1995), 129-145.

[Khardon and Roth] Khardon, R., and Roth, D., Learning to Reason. *Proceedings of AAAI-94,* Seattle, WA., (1994), 682-687.

[Minton] Minton, S., Carbonel, J.G, Knoblock, C.A., Kuokka, D. R., Etzioni, O., and Gil, Y., Explanation-based Learning: A Problem Solving Perspective. *Journal of Artificial Intelligence,* **40,** (1989), 63-118.

[Morris and Ford] Morris, R. and Ford, K., Learning Based on Failure. Unpublished.

[Sadeh et al] Sadeh, N., Sycara, K., Xiong, Y., Backtracking Techniques for the Job Shop Scheduling Constraint Satisfaction Problem. *Artificial Intelligence* **76,** (1995), 455-480.

[Struss] Struss, P., Model Abstraction for Testing of Physical Systems. Manuscript.

[Weld] Weld. D.S.,``Reasoning About Model Accuracy".*Artificial Intelligence,* **32,**(1992),255-300.

In Search of a Perfect Boltzmann Machine for CSPs*

Rolf Weißschnur and **Joachim Hertzberg**
German National Research Center for Computer Science (GMD)
AI Research Division, Schloss Birlinghoven, D-53754 Sankt Augustin, Germany

Hans Werner Guesgen†
Computer Science Department, University of Auckland
Auckland, New Zealand

Abstract

Earlier, Guesgen and Hertzberg have given a theoretical description of how to implement constraint relaxation in terms of combinatorial optimization using the concept of Boltzmann Machines (BMs). This paper sketches some lessons that an implementation of this idea has taught us about how to tailor the translation from constraint networks to BMs such that the resulting implementation be efficient.

1 BACKGROUND

Constraint satisfaction is now a standard AI problem solving technique. A constraint satisfaction problem (CSP) can informally be defined as follows: Given a set of variables over some arbitrary domains and a set of constraints, each constraint ranging over a subset of the variables, find an assignment of values to the variables such that the constraints are satisfied.

Solving a CSP using the usual constraint satisfaction methods assumes that it is solvable in the first place. However, experience tells us that many real-world problems, when formulated as CSPs, have no solution. Nevertheless, humans are often able to handle them; a natural way for doing so is trying to find an approximation (or almost solution) for the problem, which best satisfies our needs. If it is not possible to satisfy the entire constraint set, we relax the over-constrained problem by switching off or weakening some constraints such that the relaxed problem is close to the original problem, yet solvable.

There are several approaches to constraint relaxation, ranging from theory as in [6] to practical applications as in [3, 5]. All these approaches have in common that they attack a given CSP by finding a solution of a relaxed CSP that differs only minimally from the original one. The difference is expressed in terms of a metric.

There are various ways to define a metric on a given CSP, i.e., to state how different a relaxed CSP is from the original CSP and how far away the approximate solution is from the ideal one. In [9], for example, we have used the concept of penalties. Values not included in the original constraint relations are marked by natural numbers greater than 0. More recently, fuzzy set theory has been used to capture the idea of constraint relaxation [4, 7, 10, 12].

Since the aim is to find a relaxed CSP whose distance to the original CSP is minimal, constraint relaxation can be seen as an optimization problem. Recently, it has been proposed to formulate this optimization problem in terms of *Boltzmann Machines* (BMs) [1]. While BMs are a little exotic an optimization tool, they offer a number of nice properties that makes considering them plausible:

- The behavior of BMs is governed by a simulated annealing algorithm. Given that using constraint relaxation presupposes that there is no sharp concept of solution of a CSP, the statistical flavor of simulated annealing leads to a nice *anytime* [2] solution behavior: given short run time bounds, reasonable solutions can be expected; given no or large time bounds, one is guaranteed to get a globally optimal solution.
- BMs offer the potential of being implemented directly on massive parallel hardware. Given that CSPs, by their very nature, have a flavor of distributedness, massive parallel implementation is an issue to consider, even if todays standard hardware is of, or close to, von Neumann type.

This paper sketches a number of ideas we found useful when actually implementing constraint relaxation in terms of BMs. It is organized as follows: Section 2 recapitulates the essentials of translating relaxation networks into BMs, assuming that the reader

*Revised version of a paper presented at the CP-95 Workshop on Over-Constrained Systems.

†Supported by the University of Auckland Research Fund under grant A18/XXXXX/62090/F3414045.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/35 ©1996 FLAIRS

is already familiar with the basic BM notions. The key section of the paper is Section 3, describing our findings how to best tailor a BM corresponding to the original constraint problem. Section 4 concludes.

2 BASIC ISSUES

A constraint represents an arbitrary relation on variables over arbitrary domains; in this paper we assume that this relation is finite, hence consists of finitely many relation elements specifying the combinations of values that the variables are allowed to take.

To introduce our demo constraint network for this paper, consider you want to draw a graphics consisting of two boxes B_1, B_2 connected by an arrow. The boxes shall be equally sized and horizontally aligned. You have six places for positioning each of the boxes, say, $a, \ldots, f$. Figure 1 sketches the grid of positions and the intended mini-graphics. To translate this into a constraint problem, we need the following relations and—corresponding to them—constraints:

- *Position*(B_1) enumerates all positions (initially) possible for B_1; it is initialized with $\{a, \ldots, f\}$.
- *Position*(B_2) analogously to *Position*(B_1).
- *Arrow*(B_1, B_2) enumerates all pairs of positions that can be connected with an arrow spanning one position, possibly crossing levels; it is initialized with $\{(a, c), (c, a), (a, f), \ldots\}$.
- *Same-Level*(B_1, B_2) would normally enumerate all pairs of positions on the same level that are different, i.e., $\{(a, b), (b, a), \ldots, (f, e)\}$.

Note that all information is expressed in terms of relations here, while the information about possible positions of the boxes (relations *Position*(B_i)) could have been represented by domains of the respective variables. For technical convenience, we assume that all information is expressed in terms of relations; a constraint formulation that guarantees this, while being as expressive as the usual relation-and-variable formulation, can be found in [8].

This problem has obvious solutions, for example, put B_1, B_2 on positions a, c, respectively. However, to make it more interesting and to satisfy possible other constraints on the boxes that might forbid these positions, let us relax the *Same-Level*(B_1, B_2) constraint: accepting some *penalty* of, say, 2 points, pairs of positions crossing levels would also be acceptable, i.e., the pairs $(a, f), (f, a), (d, c), (c, d)$; the "really" same-level pairs obtain zero penalty. To represent this in a constraint network, we increase the arity of the *Same-Level*(B_1, B_2) constraint by one, now representing position pairs plus a penalty value, and we introduce a new relation, *SL-Penalty*(P), on an additional variable P, providing the two possible penalty values of 0 and 2. Figure 2 shows the respective constraint network.

Here then is the basic idea of mapping constraint networks to BMs, the details of which are described in Section 3. In principle, the idea is the same as in [8], but we are aiming here at a mapping yielding more efficient BMs.

Relation elements are mapped to BM units; e.g., we obtain 8 units from translating the *Arrow*(B_1, B_2) constraint, labeled with $(a, c), (d, f), \ldots$, respectively. Naturally, BM configurations with a high consensus are intended to correspond to solutions of the corresponding constraint network. Consequently, we have to take care that no two units are *on* that represent different elements of the same relation, and that only units are *on* that represent fitting elements of relations of different constraints. E.g., if a unit is *on* that represents *Position*(B_1) being a, then only a unit should also be *on* that represents a relation element of the *Arrow*(B_1, B_2) constraint with a in its first argument. So there are "good" links in a BM connecting units that should be *on* together, and "bad" links connecting other pairs of units. Moreover, we have to represent different qualities of solutions in the BM; this will be done by translating the penalty values from the constraint networks into BM biases.

Finally, as the point here is to find a "useful" translation of a constraint network into a BM, we have to state a measure telling which translation is to be prefered to another. Components of such a measure might be the time for transforming a constraint network into the corresponding BM, the average or expected run time of the BM, the average or expected ratio of finding an optimal solution, or the average quality of the solutions found after some fixed run time. These components are clearly dependent. E.g., BM run time can be bought for transformation time: Just find an optimal solution for the problem by exhaustive search, represent it in a one-unit-BM, and solve this one quickly by switching *on* the one unit—to give an extreme example. Consequently, any judgement of the practical value of transformations has to respect the trade-off between the different criteria.

In the following section, we consider in more detail how to find a smart translation, addressing in turn the different components of the to-be-developed BM, which are units, connections, and weights.

3 TUNING

We now discuss in more detail our findings in search for an effective transformation schema. The presentation addresses in turn the different components of the to-be-developed BM, which are units, connections, and weights. Performance measurements for our test implementation on the demo problem are summarized and discussed in 3.4.

3.1 The Unit Set

Constraint networks can straightforwardly be transformed into BMs by generating one unit for every relation element. However, this turns out to be impracticable: the configuration space of a BM corresponding to a constraint network with n relation elements

Figure 1: The demo problem: the grid for positioning the boxes, and the desired appearance of the graphics.

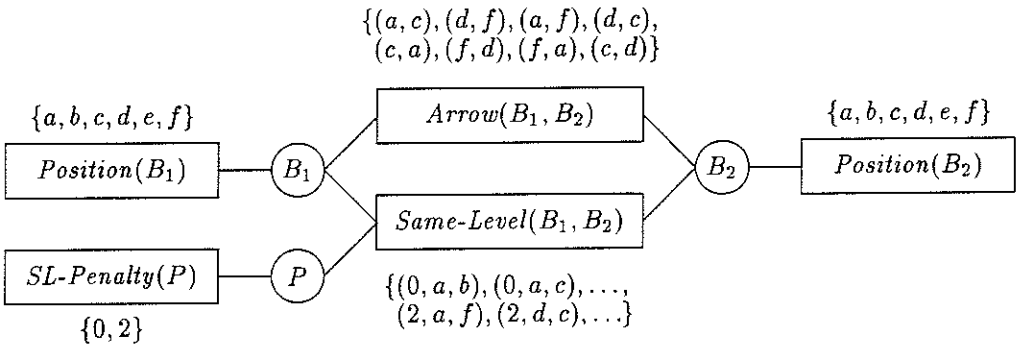


Figure 2: Constraint network representing the graphics demo problem. Boxes represent constraints; circles respresent variables. Each constraint is annotated with its initial relation elements.

consists of 2^n different configurations, n being 52 for the mini-network in Figure 2. So a practical transformation should save units whenever possible. However, there is the trade-off just mentioned between the effort spent in minimizing the unit set and the effort used for calculating the units: structurally "efficient" BMs take time to be calculated.

Luckily, there is a cheap way to find a potentially large reduction. Consider the constraint network in Figure 2. It includes constraints representing only domains that get used in turn to assemble the "proper" relations—the $Position(B_1)$ constraint representing the 6 possible positions is an example. These constraints state essentially no restriction not included in the "higher" constraints, e.g., the *Arrow* constraint, given that their relation elements involve only the values sanctioned by the "lower" ones. So we can skip the respective units in the corresponding BM—the units representing $Position(B_1)$, in our example—and generate units only for those relation elements that don't stem from a domain binding constraint.

3.2 The Connection Pattern

To differentiate between arbitrary BM configurations and configurations that correspond to solutions of the constraint network, the BM's consensus function must be such that solution configurations get high values, and all other configurations get low ones. To model this difference in the BM, we introduce the following two types of connections between units.

First, positive connections connect locally consistent units. Two units are locally consistent if they represent relation elements sharing variables and agreeing on all assignments of shared variables. E.g., the units representing $Arrow = (a,c)$ and $Same\text{-}Level = (0,a,c)$ are locally consistent, sharing the assignment to *Arrow* and agreeing that it be (a,c). For example, assuming that $Position(B_1)$ and $Position(B_2)$ were not deleted from the BM according to the unit-saving approach just described, they do not share variables; hence, the units representing them are not connected by positive connections. Positive connections get a positive strength to give the respective units a tendency to be *on* in accepted BM configurations.

Second, negative connections connect locally inconsistent units, i.e., units that represent different assignments to the same variable. This *may* occur for conflicting elements of different relations, e.g., $Arrow = (a,c)$ and $Same\text{-}Level = (0,a,b)$. It *does* occur for units representing different elements of the same relation, like $Arrow = (a,c)$ and $Arrow = (a,f)$. Giving negative connections a negative strength yields a tendency of accepted configurations to have at least one of the respective units *off*.

While classifying connections into positive and negative allows to tell configurations corresponding to solutions from those that do not, it does *not* allow to qualify BM configurations as corresponding to good or bad solutions. Following the lines of our previous work, this is done by associating a bias connection with every unit, i.e., a unary connection from a unit to itself, allowing to associate a value with this one unit being *on*. How to set these is described below.

3.3 The Weights

Having described the structure of a BM corresponding to a constraint network, we must now be more

specific about the weights. The ideal weights would be such that they allow a BM, firstly, to converge to a configuration with a high value that is sure to correspond to a solution of the constraint network and, secondly, to do so quickly. A large part of the experiments described in [13] dealt with finding a reasonable compromise between these inconsistent desires.

To start with something that does *not* work, a suggestive idea for helping the BM stumble over solutions is to increase the quality differences between configurations that represent solutions and those that do not. In our experiments, this had no good effect: The BM did not find solutions more quickly, but they got significantly worse. An explanation for the deterioration is that the quality difference between different solutions gets lower if the average value of an solution increases. Consequently, to find not only solutions, but good ones, the share of the bias weights within the value of a solution must be as large as possible.

Our original schema for setting the bias values [8, 9] was to associate a nonzero bias with every value of a variable that could be relaxed, i.e., that allowed to formulate solutions of different qualities. Moreover, it was assumed that BMs corresponding to a CSP would have to have a particular property, namely, be *feasible*, meaning that all configurations with a consensus above a certain known threshold correspond to solutions of the CSP. To guarantee that, bias values of relaxable constraints could contribute only very little to the overall consensus of a configuration in order to guarantee that they cannot make a nonsolution a seeming solution.

Now, given the different connection pattern with a higher number of negative connections that tend to favor solution configurations automatically, there is more liberality. We stick to giving a 0 bias to all units corresponding to nonrelaxable constraints. However, we have experimented with slightly nonfeasible bias functions accounting for a larger percentage of the overall consensus. (Exact definitions would require some definition overhead—please consult [13].) The idea is that the generation of a good—rather than just any—solution becomes more likely.

We have experienced no problems with the theoretical lack of feasibility. And we suspect that this is no serious practical problem anyway: A BM is never guaranteed to find a solution in finite time, even if one exists. So, whether or not a BM practically converges to a configuration that corresponds to a solution does not depend on the feasibility of the consensus function. Feasibility just guarantees the existence of a quick check whether a BM solution represents a CSP solution—no more, no less.

3.4 Results on the Demo Problem

Given all these theoretical considerations, what is their gain in terms of run time and solution quality? To provide an impression of that, we finally discuss a number of transformation variants of the graphics demo CSP. Numerical values are summarized in Table 1, whose structure we have to explain first.

Its six columns correspond to different problem transformation variants. The first variant ("Basic procedure") shows the respective measures for the straightforward transformation as described in [8, Sec. 8.2]; it is included here as a point of reference for the other variants. The other transformations are cumulative from left to right: The transformation in the n-th column *includes* the technique in the $(n-1)$-th column, adding other features. The most dramatic saving comes from reducing the number of units; it is done last here to keep the BMs at more significant sizes for the other transformation variants.

The overall run times of the different BM variants are shown in the middle row ("Total time"). It consists of the time for the respective transformation plus the run time of the BM. As the transformation may be done off-line and just once, i.e., it may be considered less interesting, we give the individual values for transformation and run times in rows No. 1 and 2. Given that BMs are stochastic devices, the value for the run time has to be an average value; we have averaged over as many runs as necessary for the respective BMs to terminate 50 times in a solution configuration (and possibly in nonsolution configurations in other cases). Measurements were done on a Sun 4 workstation, using the PARABOL BM simulator [11].

The hit rate (4th row) describes the ratio of BM runs terminating in a solution configuration (50 runs in our case) compared to the overall number of runs.

For the runs terminating in a solution configuration, we computed the value shown in the last row ("Variance") as follows: Let q be the average consensus value of all BM configurations that correspond to a solution, and q^* the average consensus value of all 50 runs terminating in a solution configuration. The variance is the percentage of q^*-q in q, i.e., $\frac{q^*-q}{q}\cdot 100$. This value could, in principle, be negative, but it is positive in all columns of Table 1, signalling that all BM variants tend to find solution configurations (if any) that are better than average.

We will now discuss the results. Starting with the connections, let us first remark that the straightforward translation of the graphics demo CSP in Figure 2 *does* already contain all positive connections, so this would bring no enhancement. Introducing negative connections—second column—slows down the transformation process considerably: There is just more work to be done. On the other hand, it cuts the run time for the BM by about 4, and almost doubles the hit rate—practically a clear advantage.

The next three variants tune the weights. Increasing the weight of the negative connections (third column) has the desired and anticipated effect of tuning the solution quality: The hit rate does not change much, but the solution quality turns far better (variance 11.54 vs. 0.44). The value for the transformation time deserves additional mentioning: In the pre-

Measure	Problem Transformation Variant					
	Basic procedure	Introduce negative connections	Increase weight of negative connections	Decrease weight of positive connections	Maximize bias weights	Reduce number of units
Transformation (sec)	0.20000	0.66666	0.19666	0.18666	0.19666	0.08333
Runtime (sec) K	1.68819	0.39234	0.41226	0.42749	0.38942	0.17246
Total time (sec)	1.88819	1.05894	0.60892	0.61414	0.58608	0.25579
Hit rate (%)	47.77	92.86	89.71	89.82	90.48	91.90
Variance (%)	1.70	0.44	11.54	15.55	15.44	30.22

Table 1: Measurements for different transformation variants of the graphics demo CSP (Figure 2).

vious variant, the strength of the newly-introduced negative connections had to be set according to the BM structure in order to guarantee feasibility of the solutions, as discussed above. As we do not want to guarantee this now, this cumbersome computation can now be saved, setting the weight to some standard value that is local for each relevant constraint. This accounts for the dramatic saving in transformation time wrt. the previous variant; but note that this saving is not intrinsic in the idea of increasing the weight of negative connections. As suspected, run time does not change significantly.

The next two variants continue along this line, favoring to find a good rather than just any solution. This is shown by another significant increase in the variance value in the fourth column (15.55 vs. 11.54, all other values staying nearly constant). Further maximizing the bias weights (fifth column) does not yield much in this situation.

The final blow comes from reducing the number of units (last column), as described in Section 3.1. As expected, all times drop significantly, due to the reduction in size of the corresponding BM. There is again a significant increase in the variance value (30.22 vs. 15.44), but this has not occurred in all test domains that we have used, and there is no reason why this effect should result in general. On the other hand, other domains showed a significant increase in hit rate after the reduction of the unit set. So, the definite result is: First, and theoretically obvious: This transformation reduces significantly the transformation and run times. Second, as a matter of empirical evidence: The overall solution behavior of the respective BMs gets better.

For a more elaborate description and discussion of the results, consult [13].

4 CONCLUSION

We have sketched some results about obtaining efficient Boltzmann Machines for solving constraint networks. Theoretical consideration suggests that a number of improvements are possible with respect to our original method of transforming CSPs into BMs [8, 9]. The essence of these improvements, which could be validated in a number of experiments, is reducing the number of units, introducing negative connections, and raising the relative contribution of bias values in the overall consensus.

References

[1] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines.* John Wiley, Cichester, 1989.

[2] T.L. Dean and M. Boddy. An analysis of time-dependent planning. In *Proc. AAAI-88*, pages 49–54, St. Paul, Minnesota, 1988.

[3] Y. Descotte and J.C. Latombe. Making compromises among antagonist constraints in a planner. *Artificial Intelligence*, 27:183–217, 1985.

[4] D. Dubois, H. Fargier, and H. Prade. Propagation and satisfaction of flexible constraints. Rapport IRIT/92-59-R, IRIT, Toulouse Cedex, France, 1992.

[5] B.N. Freeman-Benson, J. Maloney, and A. Borning. An incremental constraint solver. *Communications of the ACM*, 33:54–63, 1990.

[6] E.C. Freuder. Partial constraint satisfaction. In *Proc. IJCAI-89*, pages 278–283, Detroit, Michigan, 1989.

[7] H.W. Guesgen. A formal framework for weak constraint satisfaction based on fuzzy sets. In *Proc. ANZIIS-94*, pages 199–203, Brisbane, 1994.

[8] H.W. Guesgen and J. Hertzberg. *A Perspective of Constraint-Based Reasoning.* Lecture Notes in Artificial Intelligence 597. Springer, Berlin, 1992.

[9] H.W. Guesgen and J. Hertzberg. A constraint-based approach to spatiotemporal reasoning. *J. Appl. Intell.*, 3:71–90, 1993.

[10] A. Philpott. Fuzzy constraint satisfaction. Master's thesis, University of Auckland, Auckland, 1995.

[11] J. Prust and W. Vonolfen. Dokumentation PARABOL 1.0. Unpublished documentation, GMD, 1993.

[12] Z. Ruttkay. Fuzzy constraint satisfaction. In *Proc. FUZZ-IEEE'94*, Orlando, Florida, 1994.

[13] R. Weißschnur. Die Projektion von Constraint-Satisfaction-Problemen auf Boltzmann-Maschinen. Master's thesis, Universität Bonn, Institut für Informatik, May 1994.

DEVELOPING AND IMPLEMENTING PLANNING HEURISTICS IN PROLOG

Klaus P. Jantke and Daniel Matuschek
Hochschule für Technik, Wirtschaft und Kultur Leipzig (FH)
{jantke,matusche}@informatik.th-leipzig.de

Abstract

The present paper is based on an implemented planning system running in Quintus Prolog under SUN/OS on Sparc Stations. This one has been developed to compete another system previously implemented in Allegro Common Lisp. There have been three essentially different prototypical applications for generating technical therapy plans: a flood prevention system, a chemical fibre production installation, and a ballast tank system for off-shore platforms. This work on planning has been embedded in a comprehensive approach towards knowledge-based process supervision and control within the joint project WISCON which has been funded by the German Federal Ministry for Research and Technology under grant no. 413–4001–01 IW 204 B. The paper developes a collection of planning algorithms lucidly derived by formalizing algorithmic ideas and heuristics within the logic programming paradigm. This is based on the second author's student's project work.

1 INTRODUCTION

The present paper is intended to exhibit the usefulness of the logic programming paradigm for planning, at least for therapy plan generation in complex dynamic environments. In some more detail, we are going to illustrate how applying a logic programming approach may lead to disciplinary insights in the area of planning. As a return, one gets a couple of hints to higly desirable metaprogramming features. Although we are investigating logic programming for planning, the ideas extracted on the basis of our approach may be implemented using any other programming paradigm. In technical terms, the aim of this presentation is to investigate and illustrate the possibilities of deriving heuristics, algorithmic ideas, and approaches to machine intelligence from the use of the logic programming paradigm.

In particular, the endeavour undertaken leads to a hierarchy of planning algorithms and, even more interesting, it provides ideas for enriching the algorithms developed with the ability to learn during planning and plan execution. From a bird's-eye view, we are interested in bridging the gap between programming paradigms and algorithmic ideas for knowledge processing including learning. At a first glance, programming paradigms seem only to serve AI by providing methodologies, guidelines, and tools for implementation. A closer look exhibits interesting potentials for the deduction of algorithms and problem solving ideas.

There are further planning approaches using the logic programming paradigm like [Sau93], e.g. However, the applications domains are essentially different. This differences has a serious impact on knowledge processing methods as discussed in [AJ94b].

Although we started with a logic programming approach to planning, naturally, the implementation of our ideas described below does not essentially depend on logic programming.

2 A PEEP AT PLAN GENERATION

We clearly focus on planning throughout this paper. Nevertheless, the application of planning ideas and implementations inevitably needs an interaction with the surrounding knowledge processing environment. Our approach belongs to a comprehensive endeavour towards knowledge-based process supervision and control (cf. [MAM92] and [AJ94b], e.g.).

Planning is investigated in an area where classical STRIPS-like approaches (cf. [FN71]) usually fail. The application domain is therapy (i.e. repair) for complex dynamic processes. The pecularities of this domain

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/40

are discussed in [AJ94b]. Plans are intended to be run for process therapy. Thus, plans are programs. Because of the unavoidable vagueness and uncertainty of information about complex dynamic processes in the case of disturbance, therapy plan generation turns out to be inductive program synthesis (cf. [AJ94a]).

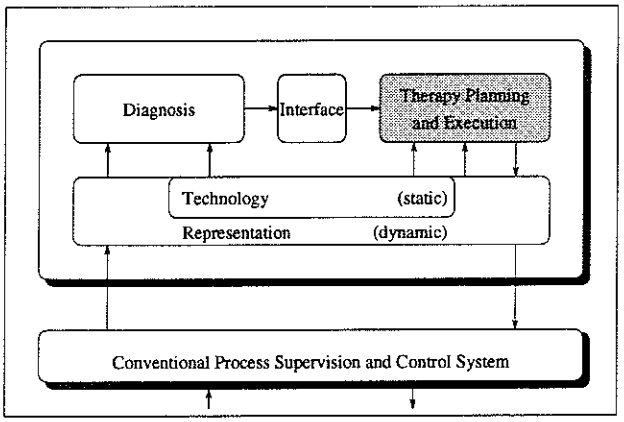


Figure 1: The Simplified Overall Knowledge Processing Architecture

A therapy control in engineering is some – possibly partial – ordering of control actions (a therapy plan) in order to influence the technical process in timely response. It's aim is to remove the causes of a detected disturbance, to perform a new control regime to minimize the losses of production, and to adapt the situation recognition which monitors alarm boundaries. These tasks should be done by both sending new setpoint values, control values, and alarm boundaries

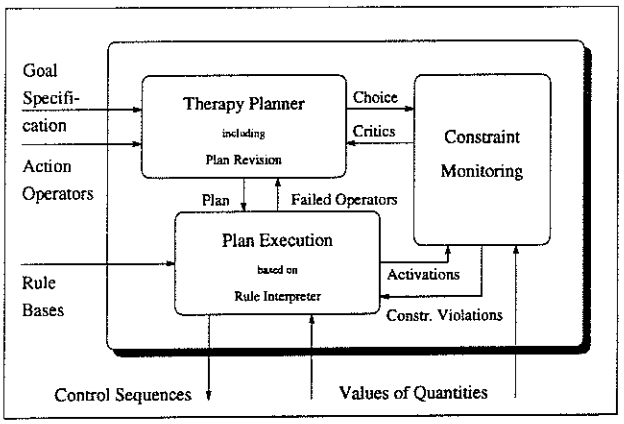


Figure 2: The Therapy Control System

to a conventional supervision and control system and informing the operator about some mechanical measures. Therefore, the corresponding control synthesis modules have to be embedded into comprehensive knowledge-based systems integrating process supervision, diagnosis, simulation, planning, and plan execution, among others.

The control synthesis which consists of a planner und an execution module is triggered by disturbance analysis. The underlying knowledge base is called a technology representation. It contains all information which is available about the technological equipment necessary for automated reasoning. There are certain hierarchies representing essentials of the underlying process. Knowledge about diagnosis and therapy (action scripts, in particular) are attached to nodes of the hierarchy describing technological objects. Inheritance is fundamental for searching this knowledge base. Further issues of the underlying knowledge representation are discussed in [Arn94b].

The planning approach itself is based on graph-theoretic fundamentals (cf. [AJ94b] and [AJ94c]). The unavoidable logical reasoning, in fact, implements the deduction operator of some modal temporal logic (cf. [Arn94a]). The key planning concept is a hierarchy of planning formalisms:

- *hierarchically structured families of graphs*, i.e. a finite collections of finite, directed, and acyclic graphs with certain regulations how to substitute one graph for a node of another one (in fact, some context-free graph grammar concept),
- *rooted families*, i.e. hierarchically structured families of graphs enriched and specified by a particular graph representing a goal specification,
- *hierarchically structured plans*, i.e. rooted families having a uniquely determined expansion (in a sense, defining a singleton language), and
- *plans*, i.e. flat graphs being the normal form of certain hierarchically structured plans, where every node has a clear operational semantics.

The outlined logic programming approach is implementing these concepts together with inference mechanisms realizing planning based on these formalizations. [MJA95] is a first publication of these ideas.

3 LOGIC PROGRAMMING FOR PLAN GENERATION

3.1 Knowledge Representation

In our target domains, we are faced to very different kinds of knowledge classified reasonably into static and dynamic knowledge. The static knowledge reflects the structure, organization, and implementation of the target process as well as essentials of the knowledge processing modules like action scripts for planning, e.g. Dynamic knowledge partially and vaguely represents the state of the supervised process.

To get process values, we use an interface to the technology representation, but action scripts and plans must be expressed using Prolog. As Prolog does not support global variables, action scripts will be stored in clauses. As a consequence, hypothesizing and possibly revising therapy plans requires a flexible and efficient changing of the clause base.

An action script is identified by some name. It mainly consists of a set of partially ordered actions subacts. It may contain some further information

```
complex(name,subacts,p_in,pref,constr)
```

where subacts is the list of all names of subactions, p_in represents the graph-theoretic knowledge of nodes without predecessors, pref is some numerically expressed preference value, and constr is a list of logical constraints.

During planning, every intermediate plan conceptually understood as a graph is represented by its nodes called actions. Actions will be stored as clauses, too. There are two different kinds of actions : compound actions, which describe goal specifications to be substituted lateron, and elementary actions. Both types of actions will be represented as

```
action(name,successors,substitutions,constraints),
```

where the list of substitutions is empty, if the action is elementary. The resulting therapy plan consists of a set of actions of the form action(_,_,[],_) indicating that nowhere in the plan further substitutions are possible.

3.2 Main Planning Algorithm

For briefty, we leave a discussion of our basic algorithm's structure to the reader:

```
1 planner :-
2   findall(Act,compound_action(Act),[]).
3 planner :-
4   compound_and_active(Act),
5   action(Act,_,Sub,_),
6   member(Gra,Sub),
7   subst(Act,Gra),
8   planner.
```

A few remarks on metaprogramming shall complete the top-level description.

- **Backtracking of Hypotheses** As therapy plans generated may be refused later, backtracking must be applicable to plans generated. This is implemented in the predicate **subst/2** which rectracts all inserted action clauses of Sub and asserts intermediately deleted actions Act), if the goal **planner** fails.
- **Metalevel Sorting** For flexibility in representing certain heuristics based on preference relations, metalevel sorting in dependence on some given partial ordering is desirable.
- **Self-Modification for Learning** For preferring successful alternatives to less successful ones during learning, there has to be identified parts of the logic program where incremental modifications may implement learning.

The simple approach above is mainly intended to provide a launching pad for developing and implementing interesting heuristics. Clarity is prefered to efficiency. It will be exploited below.

4 PLANNING ALGORITHMS

The intention of the main chapter is to introduce a number of fundamental algorithmic ideas for changing the overall behaviour of a planning system essentially. The declarative programming approach makes the changes easy to understand, easy to implement, and easy to analyze. If necessary, the prototypical programs can be rewritten under any other programming paradigm.

[JA95] is investigating a collection of similar algorithmic approaches without exploiting peculiarities of a particular programming paradigm.

4.1 Planning with Preferences

The initially introduced basic version of the planning algorithm did not use the preference values available. For shortness, we are listing only the changes to be introduced:

```
5     action(Act,_,Sub,_),
5(1)  predsort(higher_preference,Sub,Sorted_Sub),
6     member(Gra,Sorted_Sub),
```

This version differs in only one line ($5^{(1)}$), which sorts the list of possible substitutions by preference. Note that predsort/3 is a built-in predicate in SWI-Prolog, which can be easily simulated in other Prolog versions. However, it reflects a first step towards using metaprogramming concepts. member(Gra,Sorted_Sub) first selects the substitution with highest preference.

Learning is deemed important to make planners more effective and hypothetically generated plans more trustable. Our key idea is to change preference values within the course of planning. When one alternative among others during graph expansion has lead to a point where backtracking became necessary, this may be registered by decreasing its preference value. For implementing the details, a considerable number of bookkeeping strategies may be easily implemented.

4.2 Consistent Planning

Based on the underlying modal temporal logic (cf.[Arn94a]), checking the constraints of action scripts during planning is used to refute those actions which are provably inconsistent with respect to the given knowledge base

```
1    planner :-
2      findall(Act,compound_action(Act),[]).
3    planner :-
4      compound_and_active(Act),
5      action(Act,_,Sub,_),
6      member(Gra,Sorted_Sub),
6(1)   consistent_complex(Act,Gra),
7      subst(Act,Gra),
8      planner.
```

The test of consistency is performed in line $6^{(1)}$. consistent_complex(Act,Gra) means: If the node Act is substituted by Gra, the plan will turn out to be consistent.

However, this concept of consistency is more involved than the usual consistency concept in inductive inference (cf. [Wie92], e.g.). The knowledge base, i.e. the dynamic part of the underlying technology representation, is dynamically changing over time. In dependence on the mechanisms of updating the clause base, the notion of consistency refers to different stages of the process.

4.3 Planning Using Heuristics

Developing and implementing heuristics appropriate for particular problems under particular circumstances is one of the key issues of AI.

It is one of the basic intentions of the present paper to illustrate the suitability of the logic programming paradigm (1) to find appropriate positions where to invoke some heuristics and (2) to implement these heuristics easily. This applies to plan generation in several environments, at least.

As the search space of all plans is usually huge, it is recommended to use heuristics to increase performance. The logic programming approach is pointing directly to crucial steps of the planning algorithm. By using the predsort idea, one gets the following version:

```
3    planner :-
4(1)   findall(Actions,
           compound_and_active(Actions),All),
4(2)   predsort(better,All,Sorted_Actions),
4(3)   Sorted_Actions=[Act|_],
5      action(Act,_,Sub,_),
```

better/2 is any predicate provided such that its semantics for better(Act1,Act2) suggests to substitute Act1 before substituting Act2.

```
1    shorter([],[_|_]).
2    shorter([_|T1],[_|T2]) :-
3      shorter(T1,T2).
4    better(Act1,Act2) :-
5      action(Act1,_,Sub1,_),
6      action(Act2,_,Sub2,_),
7      shorter(Sub1,Sub2).
```

The few lines above implement the heuristics to expand graphs first at nodes where a minimal number of operations yield an irreducible form. In many applications, it is important to refute inconsistent alternatives as early as possible. Other heuristics may be directly plugged in via another predicate better.

5 APPLICATIONS

For illustration, we are depicting two small partial plans. They are part of therapy plans generated for a disturbance of the chemical fibre production installation. A student of us has done an enormous amount of work in acquiring and representing therapy knowledge (cf. the Master's Thesis [Zsc95]) underlying the planning experiment reported here.

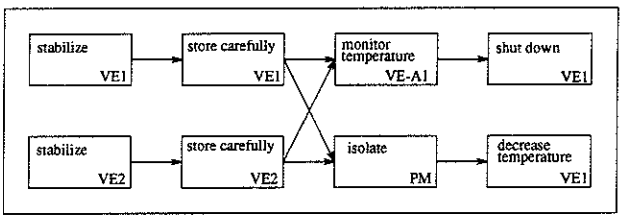


Figure 3: A Partial Plan of Highest Preference

These two plans are different in their right hand sides, where they consist of different expansions of one compound node. The first plan results from the algorithm described in chapter 4.1 above.

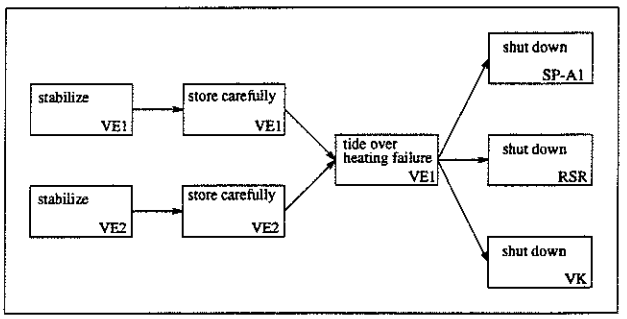


Figure 4: A Consistent Plan of Lower Preference Generated via Consistency Checking

The second plan results from the algorithm of chapter 4.2, as some constraint of the action "monitor temperature" has been disproven. In dependence on the

actually given data about the target process, the consistent planner may generate another plan.

In numerous application cases, consistent plans are refuted later because of other reasons. Thus bookkeeping to increase the preference value of actions which are more likely to be successful than others may qualify consistent planning algorithms.

6 ACKNOWLEDGEMENT

The comprehensive approach towards knowledge-based process supervision and control has been developed within the joint project WISCON funded by the German Federal Ministry for Research and Technology under grant no. 413–4001–01 IW 204 B. The conference participation of the first author has been supported by the German Research Fund (DFG) under grant 477/378/96. Our colleague Oksana Arnold (called Oki) did a pioneering work in therapy plan generation for complex dynamic systems. We are deeply indebted to Oki for all her support.

References

[AJ94a] Oksana Arnold and Klaus P. Jantke. Therapy plan generation as program synthesis. In Setsuo Arikawa and Klaus P. Jantke, editors, *Algorithmic Learning Theory, Proc. 4th International Workshop on Analogical and Inductive Inference (AII'94) and the 5th International Workshop on Algorithmic Learning Theory (ALT'94), October 10-15, 1994, Reinhardsbrunn Castle, Germany*, volume 872 of *LNAI*, pages 40–55. Springer-Verlag, 1994.

[AJ94b] Oksana Arnold and Klaus P. Jantke. Therapy plan generation in complex dynamic environments. ICSI Report TR–94–054, International Computer Science Institute, Berkeley, California, October 1994.

[AJ94c] Oksana Arnold and Klaus P. Jantke. Therapy plans as hierarchically structured graphs. In *Fifth International Workshop on Graph Grammars and their Application to Computer Science, Williamsburg, Virginia, USA*, November 1994.

[Arn94a] Oksana Arnold. A logic of constraints for dynamic process control. WISCON Report 09/94, HTWK Leipzig (FH), Fachbereich IMN, December 1994.

[Arn94b] Oksana Arnold. Towards structure and management of knowledge bases for controlling. In Eberhard Köhler, editor, *39. Internationales Wissenschaftliches Kolloquium der TU Ilmenau, Band 3*, pages 30–36. Techn. Universität Ilmenau, 1994.

[FN71] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to theorem proving in problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[JA95] Klaus P. Jantke and Oksana Arnold. Variants of plan generation for complex dynamic systems. In Xin Rao, editor, *Proc. 8th Australian Conference on Artificial Intelligence (AI'95), November 13-17, 1995, Canberra, Australia*, pages 531–538. World Scientific Publ. Co., 1995.

[MAM92] Volker May, Oksana Arnold, and Uwe Metzner. Wissensverarbeitung in dynamischen Prozeßumgebungen – Eine Anforderungsspezifikation. WISCON Report 01/92, HTWK Leipzig (FH), Fachbereich IMN, March 1992.

[MJA95] Daniel Matuschek, Klaus P. Jantke, and Oksana Arnold. Generierung von Therapieplänen mit Mitteln der logischen Programmierung. In Peter H. Schmitt, editor, *Logik in der Informatik, 3. Jahrestagung der GI-Fachgruppe 0.1.6*, pages 75–78. Universität Karlsruhe, Fakultät für Informatik, Bericht 23/95, Juni 1995.

[Sau93] Jürgen Sauer. *Wissensbasiertes Lösen von Ablaufplanungsproblemen durch explizite Heuristiken*, volume 37 of *DISKI, Dissertationen zur Künstlichen Intelligenz*. infix, 1993.

[Wie92] Rolf Wiehagen. From inductive inference to algorithmic learning theory. In Shuji Doshita, Koichi Furukawa, Klaus P. Jantke, and Toyaki Nishida, editors, *Proc. 3rd Workshop on Algorithmic Learning Theory, (ALT'92), October 20–22, 1992, Tokyo*, volume 743 of *Lecture Notes in Artificial Intelligence*, pages 13–24. Springer-Verlag, 1992.

[Zsc95] Christoph Zschiesche. Erstellung und Analyse von Wissensbasen der Prozeßsicherung und -optimierung zur Verbesserung und Erweiterung geeigneter Wissensrepräsentationen. Master's thesis, Leipzig University of Technology, August 1995.

CADDY: Adaptive Learning of Plans in a Probabilistic Domain

Jen-Lung Chiu
Microsoft Corporation
jenlc@microsoft.com

Abstract

The CADDY system uses adaptive planning to find efficient plans for a fixed goal in an imperfectly known and controlled environment. The system keeps a record of the costs of known plans in previous situations. In a new situation, CADDY estimates the cost of each known plan and chooses the most promising. It then tries domain-specific adaptations to improve the plan. The quality of these plans is estimated using simulation, and these results are recorded. The best plan is returned.

The test domain used is a simulated game of golf in a foggy day. Perception is limited by the maximal distance from which the ball can be seen, and by the precision with which the distance and direction of a stroke can be judged. Control is limited by the precision of the strokes. The "cost" of an execution is a weighted sum of the number of strokes, the time spent moving and looking, and a penalty for replacing a lost ball.

Despite the simplicity of the world and the goal, the comparative costs of different plans vary widely, depending on the control and cost parameters. Moreover, it is impractical to determine these costs analytically. This domain is therefore well-suited to heuristic adaptation and evaluation.

Finally, the paper presents our experimental results; compares CADDY to other adaptive and probabilistic planners; and discusses possible extensions and generalizations to other domains.

1 INTRODUCTION

The structure of a planner reflects the space of planning problems being addressed. AI planners, for the most part, address "classical" planning problems, characterized by most or all of the following features:

- The state space is discrete.
- The current state is always completely known.
- Actions have determinate effects.
- The correctness and cost of a plan are easily computed.
- There is a declarative description of the causal structure of the world.
- There is a large space of potential top-level goals.

Such a problem space is well served by classical planners [3, 6, 13], which "think about" the causal structure of actions to generate a provably correct plan consisting of a finite sequence of actions.

By contrast, many robotics applications involve an agent who works in a world of continuous parameters and who is much more limited in his control, in his perceptions, and in the range of tasks he must perform. The problem space in such applications may well exhibit quite different features.

- The state space is continuous.
- The current state is imperfectly known.
- The effect of an action is determined probabilistically.
- The correctness and expected cost of a plan are difficult to compute.
- The causal structure of the world is characterized, for the planner, by some set of parameters whose meaning is opaque.
- There is a unique top-level goal, or a small set of top-level goals.

The CADDY system [4] is an adaptive planner designed to search for efficient plans in problem spaces with the above characteristics. The system maintains a library of known plans with records of their cost in previously encountered situations. When a new situation is encountered, CADDY estimates the quality of each plan and chooses the most promising candidate. It then tries a variety of domain-specific adaptations which may improve the plan. The quality of these plans in the new situation is estimated using multiple simulations; these results are recorded in the library. The best plan for the new situation is returned.

The testbed domain we have used for CADDY is a simulated game of golf played on a foggy day. In this microworld, perception is limited by the maximal distance from which the ball can be seen, and by the precision with which the distance and direction of a stroke can be judged. Control is limited by the precision of the strokes. The "cost" of an execution of a plan is a weighted sum of the number of strokes, the time spent moving and looking, and a penalty for replacing a lost ball.

The paper will focus on the description and discussion of CADDY. Section 2 describes the golf world testbed. Section 3 gives an overview of CADDY and discusses the functionalities of each module. Section 4 compares CADDY with planners for probabilistic domains and other adaptive planners. Section 5 presents experimental results for the current prototype

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/45 ©1996 FLAIRS

system. Finally, section 6 discusses possible directions for system extension and how to generalize CADDY to other richer problem domains.

2 PROBLEM DOMAIN

The initial prototype system of CADDY operates in a testbed domain of a mediocre golfer playing golf on a foggy day. This domain was chosen to meet the following criteria:

1. Perception is partial; action is probabilistic; and the world is continuous.
2. The causal structure can be characterized by a collection of real-valued parameters. Many different types of plans are optimal under different settings of the parameters.
3. The world is designed to be simple, subject to objectives (1) and (2).

The golf course is an uniform plane; there are no roughs or traps. The golf hole, by convention, is a circle of radius ϵ around the origin, ϵ being a physical parameter. The physical state of the world at any moment consists of a triple of points: the current position of the ball, the current position of the golfer, and the position of the ball at the last stroke (used when a lost ball is replaced). Since the world is invariant under rotation about the origin, it has five real degrees of freedom.

There are five types of actions: hitting, moving, looking for the ball, scanning for the ball while moving on a path, and replacing a lost ball. The actions are characterized in terms of their preconditions, their effects on the physical world, their effect on the epistemic state of the agent, and their cost.

(hit X θ Z ψ). X and θ are input parameters while Z and ψ are output parameters.

CADDY tries to hit the golf ball distance X in direction θ. The golf ball actually goes to distance Y in direction ϕ, where Y follows a normal distribution centered at X with standard deviation $\beta_1 X$ and ϕ follows a normal distribution centered at θ with standard deviation α_1.

After hitting the ball, CADDY looks at the flying ball and makes an estimate of its final position. CADDY judges that the golf ball will land at a position with distance Z and direction ψ from CADDY's current position, where Z follows a normal distribution centered at Y with standard deviation $\beta_2 Y$ and ψ follows a normal distribution centered at ϕ with standard deviation α_2. All these distributions are independent.

The "hit" action can be executed if the golfer is at the same point as the golf ball.

The cost of (hit X θ Z ψ) is a constant *c_hit*, independent of the hitting distance.

(move X θ) where X and θ are input parameters.

CADDY moves distance X in direction θ. We assume that CADDY has perfect control in moving.

The cost of (move X θ) is *c_move* times X. It can always be executed.

(look Z ψ) where Z and ψ are output parameters.

CADDY looks for the golf ball. If the ball is within distance λ (the perception range of CADDY's sensors), Z and ψ will be distance and direction of the ball from CADDY; otherwise Z will be -1.

The cost of (look Z ψ) is a constant *c_look*. It can always be executed.

(scan $PATH$ Z ψ) where $PATH$ is input parameter while Z and ψ are output parameters.

$PATH$ is a list of (*distance*,*direction*) pairs. CADDY moves along $PATH$ scanning for the golf ball. If it finds the ball (that is, the ball is within distance λ), then it stops, and returns Z and ψ to be the distance and direction of the ball from CADDY's current position. Otherwise, it completes scanning the whole $PATH$ and returns $Z = -1$. We can view scan as a sequence of *look*ing and *move*ing a short distance along the pre-defined route $PATH$.

The cost of (scan $PATH$ Z ψ) is *c_look* plus *c_scan* times the distance CADDY actually moves through the $PATH$. *c_scan* should be greater than *c_move* but less than *c_move*+*c_look*/λ. The first inequality ensures that *scan* is more expensive than the sequence of *move*s, which form the $PATH$; while the second inequality ensures that *scan* is cheaper than repeatedly *move* a short distance λ and *look*. A scan can always be performed.

(replace)

CADDY gives up looking the golf ball, returns to the position of the last hit, and starts again with a new ball.

The cost of (replace) is *c_replace* plus *c_move* times the distance CADDY travels to return back to the position where CADDY hit the ball last time.

Thus, *look* is a pure sensing action (an action that provide information of the physical environment); *move* and *replace* are effect actions (actions that change the state of the physical environment); while *scan* and *hit* involve both sensing and physical changes. The planner can perfectly predict the effects of actions *move* and *replace*. It cannot perfectly predict the effects of *look* and *scan*, due to limited perception; nor the effect of *hit*, due to limited perception and imperfect physical control.

CADDY's knowledge of the world varies over time. It always knows the position of the golfer, and the position of the golf ball at the last hit (the position to which the ball would be replaced.) Its knowledge of the current position of the golf ball is in one of three conditions:

- Initially, and between any successful *look* or *scan* and the subsequent *hit*, CADDY knows the exact location of the ball.
- Immediately after a *hit*, CADDY knows a probability distribution for the position of the ball, which is a product of a Gaussian over distance from the hit point times a Gaussian over direction from the hit point. If the distance argument to the hit is X and the perceived distance is P, then the distribution of the true distance is a Gaussian centered at $(\beta_1^2 P + \beta_2^2 X)/(\beta_1^2 + \beta_2^2)$ with standard deviation $\beta_1\beta_2/\sqrt{(\beta_1^2 + \beta_2^2)}$

- The effect of an unsuccessful *look* or *scan* is to carve the region perceived — a circle or strip of radius λ — out of the probability distribution, and to raise the probability in the remaining region by renormalization. After a series of unsuccessful *looks* and *scans*, therefore, the probability distribution is the original Gaussian with a collection of holes poked out of it.

The characteristics of the golf world are thus determined by six physical parameters (α_1, β_1, α_2, β_2, λ, and ϵ) and five cost parameters (*c_hit*, *c_move*, *c_look*, *c_scan*, and *c_replace*). The current implementation of CADDY allows four of these — *c_replace*, α_1, β_1 and λ — to vary independently, while the values of the remaining five are either constant or functionally determined by these four.

It is evident that any reasonable plan for CADDY has the following general form: First, you hit the ball toward the hole. Then you search for it, using some combination of looks and scans, until either you see it or you give up. If you see it, you move straight to it; if you give up, you replace it at the last hit point. Then hit it again, and iterate.

The variations in plan therefore rest in the strategies for deciding how far to hit the ball; how to search for it; and when to give up and replace it. The formulation of these strategies depends on the various world parameters. For example, if the hole is large, hitting is accurate, and replacement is cheaper than searching, then the optimal strategy could be to keep replacing the ball at the starting point and trying until you hit the ball directly from the start to the hole. If searching is expensive as compared to hits, and control is poor then it is advisable for the golfer never to hit the ball further than he can see it. If searching is expensive and control is fairly good, then the golfer should hit it far enough that the uncertainty in position is comparable to the visibility distance. If the control in direction is good but control in distance is poor, so that the region of uncertainty is long and thin, then the golfer may wish to scan on a linear path. If the region of uncertainty is both long and wide, then the golfer may wish to do a spiral search, starting from the center.

3 SYSTEM OVERVIEW

CADDY is an adaptive planner, which works by finding a promising plan in its library, and modifying it to fit altered circumstances. The CADDY planning system is composed by the following five modules:

The Plan Library

The Plan Library (PL) consists of three tables of known plans: the Best Table (BT), which stores the best plan CADDY has actually found for particular circumstances; the Failure Table (FT), which stores the failed experiences CADDY has encountered; and the Simulation Table (ST), which records the simulation costs of each plan in PL under selected circumstances. BT and FT represent the history of CADDY's execution while ST is used as an assistance table which helps the comparison the similarity between plans and circumstances. The significance of BT, FT, and ST will be discussed later in this section.

A plan in CADDY is represented as a tree whose leaf nodes are primitive actions, and whose inner nodes are the control structures "sequence", "cond", and "while". Nodes in a plan are annotated with comments indicating the purpose of the corresponding step.

The Estimation Module

The Estimation Module (EM) estimates the cost of a known plan under the current circumstances using information stored in ST.

In the current implementation, EM uses linear interpolation on system parameters (physical parameters α_1, β_1, λ, and cost parameter *c_replace*) and the starting situation (distance between CADDY and the hole at the beginning) to calculate the estimated cost. This is a 5-stage interpolation. In each stage, EM searches through ST, chooses two nearest values of the current attribute, and applies linear interpolation to get the estimated cost under current interpolation stage. After passing down then passing up 5 stages, EM will get the estimated cost of a specific plan under chosen circumstance. Information stored in ST is used as basis points during interpolations.

The Plan Selection Module

The objective of the Plan Selection Module (PSM) is to find the best plan in PL for the current circumstances. PSM first looks through BT to find if CADDY has met similar circumstances before. If a match is found, PSM retrieves the plan from PL. Otherwise PSM uses EM to estimate the cost of each plan in PL and passes the plan with the least estimated cost to PAM. The use of BT prevents CADDY from executing unnecessary interpolations over and over, thus helping to improve the system performance.

The Plan Adaptation Module

The Plan Adaptation Module (PAM) modifies the candidate plan output by PSM to fit the current circumstances using domain-specific heuristics. PAM begins by collecting the set of heuristics that apparently apply to the selected plan in the current circumstances. PAM then checks in FT whether this set of heuristic rules has previously been found to fail in similar circumstances. If not, PAM is safe to apply these collected heuristics to the selected plan. Otherwise PAM tries one heuristic at a time, checks whether it gives an improvement, and applies all those that give improvements. The use of FT to prevent CADDY from attempting an adaptation that has already failed is an application of failure-driven learning.

The Plan Execution Module (PEM)

PEM performs multiple (typically hundreds) of simulations of the execution of the plan generated by PAM, in order to determine its expected value. PEM includes a user interface which can demonstrate the execution of a plan on a graphic board and give helpful information and an execution log for trace purpose.

PEM stores the simulation result in PL for future uses. If the plan generated by PAM outperforms the plan selected by PSM, PEM stores the generated plan in PL, records the simulation result in BT, and invokes simulations under selected circumstances and records these results in ST so that this newly

generated plan can be interpolated and selected by future runs.

But if the plan generated by PEM is outperformed by the plan selected by PSM, PEM discards the generated plan and records the failed experience in FT so that CADDY can avoid performing the same adaptation on the same selected plan under similar circumstances.

4 RELATED WORKS

Several forms of probabilistic reasoning techniques have recently been applied to planning. These include probabilistic temporal reasoning [5]; QPNs (Qualitative Probabilistic Networks) [16], a graphical representation for probabilistic relationship among variables; BURIDAN [11], an implemented partial-order planner that uses probabilities to represent uncertainty in the world state and in the effects of the operators; and POMDPs (Partially Observable Markov Decision Processes) [2], an extension of MDPs (Markov Decision Processes) [9]. Some testbeds (like the Tileworld, the Truckworld, or Phoenix discussed in [8]) also use probabilistic models to govern the occurrence of events.

Of these, BURIDAN and POMDPs are closest to CADDY. [4] includes detailed comparison between CADDY and the above probabilistic planning models and systems.

Many forms of adaptive planning have been studied [1, 10, 12, 14, 15]. Of these systems, the closest to CADDY is the CHEF system [7]. CHEF is a case-based planner that builds new plans out of its memory of old ones in the domain of Chinese Szechwan cooking. We will consider here how the functionalities of the major modules of CHEF are achieved in CADDY.

Problem Anticipation

The purpose of the Problem Anticipator is to detect features in the current input that have participated in past planning problems. The Problem Anticipator avoids CHEF from making the same fault it has made before. In CADDY, PAM uses information in FT to prevent CADDY from constructing failed plans. The difference is that the Problem Anticipator is executed at the beginning and these features are added as a goal to avoid these problems (failure prevention); while in CADDY, FT serves as to inform PAM that the adapted plan is not a good one (failure detection).

Plan Retrieval

The Plan Retriever searches for a plan from a memory of plans indexed by the goals the plan satisfies and the problems the plan solves. In CADDY, PSM chooses plan either by consulting information in BT or by choosing the plan with the least cost with the assistance of EM.

Plan Modification

The Plan Modifier alters the plan selected to satisfy the goal that are not yet achieved. In CADDY, PAM uses domain-dependent heuristic rules to adapt the plan selected by PSM.

Plan Repair and Credit Assignment

CHEF fixes the faulty plan when a plan fails to achieve its goal, and CHEF also records the causal explanation of why the failure has occurred so that the Problem Anticipator can identify the features in the input that lead to the same problem in future planning executions. CADDY does not try to repair the faulty plan but CADDY records the failure experience in FT to prevent from making the same plan given the same circumstance.

Since CHEF does not deal with uncertainty, it can correctly predict whether the constructed plan will succeed. In CADDY, a plan can run well one time and poorly the next. CADDY uses simulation to estimate the effectiveness of the generated plan, but it cannot predict the outcome of an actual execution. This is why we do not repair the generated plan; a plan can be poorly executed although it is projected as a good one.

Plan Storage

PEM records needed information in BT, FT, or ST. The generated plan is also stored in PL if it outperforms the original selected plan.

5 SIMULATION RESULTS

Currently CADDY along with the golf world testbed are implemented on a SUN workstation, with a simple graphic interface. All program modules are written in C. PL initially contains 18 different plans, BT and FT are empty, and ST contains the simulated cost results of the 18 plans under different parameter assignments.

The graphic interface enables users to change parameter settings as well as to initialize the position of CADDY and the golf ball at the starting situation. While the graphic interface will simulate and display the effect of each primitive action performed and the execution of the selected plan, another text window will also display the execution log for tracing purpose. Demonstrated below are several typical simulation results we got from the current system.

With ($c\_replace$, λ, α_1, β_1) = (120.00, 35.00, 3.00, 0.03) and initial position (200.0, 200.0), PSM chooses plan **a3** with estimated cost 714.16, and PAM applies heuristic rules **H1** and **R3** to plan **a3**, generating the new plan p0. After 256 simulations, the average costs of the two plans are 752.66, 670.84; with 3.96, 3.28 *hit* operations and 0.00, 0.00 *replace* operations. Actual simulation results show that after adapting the selected plan, CADDY gets a better plan with lower average cost and fewer *hit* operations. The newly generated plan **p0** is stored in PL; the execution experience (**p0**, $c\_replace$ = 120.00, λ = 35.00, α_1 = 3.00, β_1 = 0.03, $dist$ = 282.00, $cost$ = 670.84) is logged in BT; and PEM evokes simulations of the plan **p0** under selected circumstances and records the result in ST.

With ($c\_replace$, λ, α_1, β_1) = (54.00, 26.00, 28.00, 0.24) and initial position (200.0, 200.0), PSM chooses plan **e3** with estimated cost 1526.65, and PAM applies heuristic rules **H2** and **R1** to plan **e3**. After 256 simulations, the average costs of both plans are 2074.67, 1934.58; with 8.26, 9.02 *hit* operations and 0.04, 1.48 *replace* operations. Though with higher number of *hit* and *replace* operations, the adapted plan shows a lower average cost because it now spends less on *scan*ing those bad

hits. Again, simulation results show that applying heuristics makes CADDY get a better plan with lower average cost. The newly generated plan **p1** is stored in PL; the execution experience is logged in BT; and PEM evokes simulations of the plan **p1** under selected circumstances and records the result in ST.

With (*c_replace*, λ, α_1, β_1) = (100.00, 56.00, 20.00, 0.20) and initial position (200.0, 200.0), PSM chooses plan **e3** with estimated cost 1259.75, and PAM applies heuristic rules **H2** and **R1** to plan **e3**. After 256 simulations, the average costs of both plans are 1359.77, 1420.30; with 6.69, 7.07 *hit* operations and 0.00, 0.45 *replace* operations. This time, the adapted plan is worse than the original plan with higher cost. The newly generated plan is then discarded by PEM, the adaptation experience (**e3**, λ = 56.00, α_1 = 20.00, β_1 = 0.20, *c_replace* = 100.00, *dist* = 282.00, {**H2**, **R1**}) is logged in FT, and the execution experience is logged in BT.

With (*c_replace*, λ, α_1, β_1) = (54.00, 26.00, 28.00, 0.24) and initial position (200.0, 200.0), PSM chooses plan **p1** with estimated cost 1934.58 since CADDY remembers the similar situation has been inputed before (**p1** was generated from **e3** by PAM with the same input starting situation). PAM does not apply any heuristic repair rules to the selected plan, and PEM shows similar simulation result.

With (*c_replace*, λ, α_1, β_1) = (100.00, 56.00, 20.00, 0.20) and initial position (200.0, 200.0), PSM chooses plan **e3** with estimated cost 1259.75, and PAM intends to apply heuristic rules **H2** and **R1** to plan **e3** but finds the failed experience in FT, so PAM decides to apply only rule **H2** only. After 256 simulations, the average costs of both plans are 1359.77, 1398.16; with 6.69, 6.78 *hit* operations and 0.00, 0.00 *replace* operations. This time, the adapted plan is still worse than the original plan with higher cost. The newly generated plan is then discarded by PEM, the adaptation experience is logged in FT, and the execution experience is updated in BT. Though the adapted plan is still outperformed by the original selected plan, the simulated cost this time (1398.16) is slightly better than the simulated cost last time (1420.30).

6 CONCLUSION

To the best of our knowledge, CADDY is the first attempt to apply re-planning techniques to planning in a probabilistic domain. CADDY's success in its testbed domain indicates that this is a promising direction for further study and development.

The next step in this research is to extend CADDY to more realistic domains. Two issues seem particularly important in this extension:

- Scalability. The current implementation of CADDY works in a domain characterized by four variable real parameters. Extending it to work in a domain of hundreds of parameters will require more powerful techniques in many of the modules.
- Adaptation heuristics. The library of adaptation heuristics in CADDY was developed *ad hoc*. It would be very valuable to automate or at least to characterize these heuristics in terms of the properties of the domains.

References

[1] Richard Alterman. Adaptive Planning. *Cognitive Science*, 12:393–421, 1988.

[2] Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting Optimally in Partially Observable Stochastic Domains. In *AAAI–94*, pages 1023–1028, 1994.

[3] David Chapman. Planning for Conjunctive Goals. *Artificial Intelligence*, 32(3):333–377, 1987.

[4] Jen-Lung Chiu. *Planning in Imperfect World Using Previous Experiences*. PhD thesis, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, May 1995.

[5] Thomas L. Dean and Keiji Kanazawa. Probabilistic Temporal Reasoning. In *AAAI–88*, pages 524–528, 1988.

[6] Richard E. Fikes and Nils J. Nilsson. STRIPS : A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3):189–208, 1971.

[7] Kristian J. Hammond. CHEF : A Model of Case–based Planning. In *AAAI–86*, pages 267–271, 1986.

[8] Steve Hanks, Martha Pollack, and Paul Cohen. Benchmarks, Testbeds, Controlled Experimentation, and the Design of Agent Architectures. Technical Report 93–06–05, Department of Computer Science and Engineering, University of Washington, June 1993.

[9] Ronald A. Howard. *Dynamic Programming and Markov Processes*. The MIT Publisher, Cambridge, Massachusetts, U.S.A., 1960.

[10] Subbarao Kambhampati and James Hendler. A Validation–Structure–Based Theory of Plan Modification and Reuse. *Artificial Intelligence*, 55:193–258, 1992.

[11] Nicholas Kushmerick, Steve Hanks, and Daniel S. Weld. An Algorithm for Probabilistic Least–Commitment Planning. In *AAAI–94*, 1994.

[12] R. Oehlmann, D. Sleeman, and P. Edwards. Learning Plan Transformations from Self–Questions : A Memory–Based Approach. In *AAAI–93*, pages 520–525, 1993.

[13] Earl D. Sacerdoti. *A Structure for Plans and Behaviour*. Elsevier–North, Holland, 1977.

[14] Reid G. Simmons. *Combining Associational and Causal Reasoning to Solve Interpretation and Planning Problems*. PhD thesis, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1988.

[15] Gerald Jay Sussman. *A Computer Model of Skill Acquisition*. American Elseview Publishing Company, Inc., New York, New York, U.S.A., 1975.

[16] Michael P. Wellman. Fundamental Concepts of Qualitative Probabilistic Networks. *Artificial Intelligence*, 44(3):257–303, 1990.

PLANRIK: A HIERARCHICAL NONLINEAR PLANNER BASED ON OBJECT STATES

Enrique Díaz-Infante
Instituto Tecnológico Autónomo de México
nrik@colossus.rhon.itam.mx

Carlos Zozaya-Gorostiza
Instituto Tecnológico Autónomo de México
zozaya@lamport.rhon.itam.mx

Abstract

The article describes the design and behavior of PLANRIK, a domain independent hierarchical nonlinear planner using a STRIPS like representation of *operators* and *states* [3] that solves for interactive conjunctive goals in a simple, different and more efficient manner than other planners such as TWEAK [1] or ABTWEAK [8,9].

PLANRIK efficiently solves for conjunctive goals by obtaining *partial plans* for each *object* of the problem domain (i.e., a block in the blocks world, a ring in the towers of Hanoi, etc.), and by merging these partial plans into a *global plan* that will become the solution to the problem. Each *partial plan* is generated by ignoring all other objects, and by using a *network of required object states* resulting from previous planning activities within the solution process.

Some differences in the behavior of PLANRIK with respect to other planning systems are the following: a) *operators* are reordered as a consequence of replacing a partial plan with another for the same object, and not by using *critics, tick-lists* or *white-knights* such as those found in NOAH [5], INTERPLAN [7] or TWEAK, respectively; b) the generation of a partial plan for an *object* is guided solely by a *network of required object states* for that particular object and not by a complete analysis of the global plan; and c) *abstraction* is handled by ignoring all other objects (as well as their corresponding logical formulas) when obtaining a partial plan for a particular object, and not by associating a *criticality index* for these formulas like other hierarchical planners do (e.g., ABSTRIPS [4] or ABTWEAK).

BACKGROUND

PLANRIK is an evolution of an algorithm which was embedded in a knowledge-based system architecture for process planning called PLANEX [10]. In this architecture, each of the operators used to obtain the process plan required for constructing a building or manufacturing a product (e.g., obtain activities, estimate activity duration, determine activity precedences, obtain activity costs, etc.) was implemented as a function, and knowledge about their preconditions and effects was represented using a STRIPS like description of the operator. Whenever the user changed the design of the desired product or the information of the process plan, the system applied the planning algorithm to decide which operators had to be applied to update the plan or to achieve what the user wanted (e.g., determine how much is the first floor of the building going to cost).

While the algorithm incorporated in PLANEX was tested successfully in the solution of classical blocks world planning problems like the "Sussman anomaly" [6] and the "double cross conflict" of four blocks described by Corkill [2], it had the limitation of not being able to include in the plan more than one instance of the same operator. Therefore, the algorithm was not applicable to problems that require using an operator more than once (e.g., the Hanoi-3 problem). This limitation, however, was not important for the behavior of PLANEX, since process planning operators were instantiated only once on each problem solving cycle.

PROBLEM REPRESENTATION

The problem consists of finding a sequence of *operators* transforming an initial world model into a desired *world model*. The world model, also called a *world state*, is represented by a set of *formulas* of first order logic, also called predicates or *literals*; the *operators* change the world model by adding or deleting some of these formulas, and their applicability is represented by a set of such predicates.

PLANRIK uses a representation of *operators* and *world states* based on *object states*. Objects are things whose properties change when operators are applied; thus, in the blocks world, each "block" is considered to be an object, while in the towers of Hanoi, an object corresponds to a "ring" (or disk). The world model is specified by describing the states of the objects it contains using a set of *formulas* of first order logic.

In PLANRIK, operators are expressed in such manner that they may be associated with a single object. An operator that has two or more arguments which

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/50 ©1996 FLAIRS

represent objects requires to be substituted by operators having only one argument. For example, the operator of the blocks world domain PUT-ON (x,y) is expressed as multiple operators, one for each possible value of argument x (e.g., PUT-A-ON, PUT-B-ON and so forth), since both x and y are objects; in contrast, the operator of the towers of Hanoi MOVE-SMALL (p1,p2) would not need to be decomposed since p1 and p2 represent *pegs* but not *rings* (i.e., the possible instances of p1 and p2 are not objects of the domain). As we will see below, being able to associate each operator with an object will turn out to be very helpful when obtaining *object plans.*

The system deals with the *frame problem* by taking as valid the STRIPS assumption that the truth value of formulas does not change unless subsequent operators deny or assert them.

PLANRIK ARCHITECTURE

PLANRIK has a three-layered architecture, as shown in Figure 1:

- The top layer, called the *strategy layer,* consists of a tree of *object* labels that describes the order in which object plans have to be achieved. The system uses the information stored in this layer for knowing what object plan has to be generated next, both when the previous plan was successfully added to the *global plan* or when this task was unsuccessful and therefore *backtracking* is necessary.

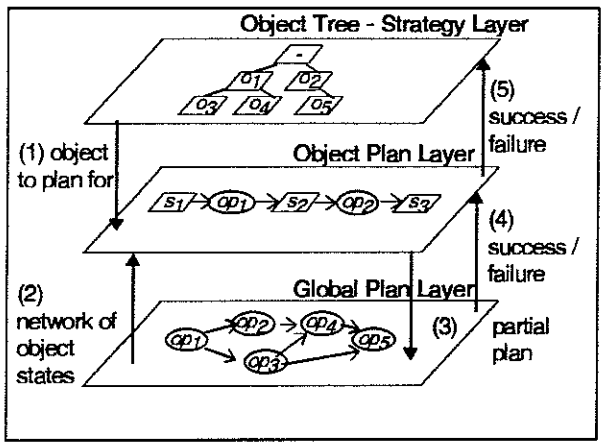


Figure 1: PLANRIK Architecture

- The middle layer, called the *object plan layer,* generates one or more linear plans for the object indicated by the top layer. Each of these linear plans specify a list of operators that, when applied, will change the object from its initial to its final state. These plans, however, may be constrained to include a set of object states in a particular sequence, according to the information contained in a *network of required object states.* This network is obtained from an analysis of the previous global plan before proceeding to generate the object plans.
- The bottom layer, called the *global plan layer,* contains a network of operators, preconditions and effects that are *interpreted* to obtain a partial or complete solution of the problem.

PLANRIK BEHAVIOR

The behavior of PLANRIK can be described as follows:

Step 0. Creation of the strategy layer.

0.1 Objects are ordered based on the total number of preconditions belonging to the operators associated with each of them[1].

0.2 A tree of object labels is created using this ordering, such that: a) children nodes of a particular node are added to their parent node in descending order from left to right; b) the path of a particular node to the root of the tree must not contain more than one label of any node; c) the tree has to have as many levels as objects.

0.3 Initialize the *network of required object states* to an empty set.

Step 1. Strategy Layer: Choose an Object.

1.1 Using depth-first search, find a new node of the tree stored in the *strategy layer.* If all nodes of the *strategy layer* have been visited, stop and indicate "failure"; otherwise, go to step 1.2.

1.2 Create the network of required object states (NROS). This network is constructed by: a) adding to the initial and goal states of the current object, all the object states that are preconditions of the operators already included in the global plan, and b) by obtaining precedences among these object states using a *precedence interpretation table* as explained below.

Step 2. Object-Plan Layer: Generate an Object Plan.

2.1 Choose a sequence of object states from the NROS. The nodes of the NROS obtained in step 1.2 are topologically sorted and a linear sequence of object states is obtained. If all possible sequences have been analyzed, backtrack one level in the tree of the top layer and go to step 1.1.

2.2 Generate a linear plan for the actual object that satisfies the sequence obtained in step 2.1. If failure is obtained, go to step 2.1.

2.3 Complete the object plan with the description of its operators including all their preconditions and effects.

[1] This generic heuristic (applicable to any problem) for ordering the objects is intuitive, but could be changed for a particular problem domain if this were convenient. The resulting ordering will only affect the efficiency (but not the effectiveness) of the system .

Step 3. Merge the Object Plan with the Global Plan.

3.1 Merge the object plan obtained in step 2.3 with the current global plan of the bottom layer. If failure is obtained, go to step 2.2.

3.2 Interpret precedences for the operators of the *global plan*, by analyzing their relationships with their immediate neighbors. For this task, the table of precedences shown in Figure 2 is used.

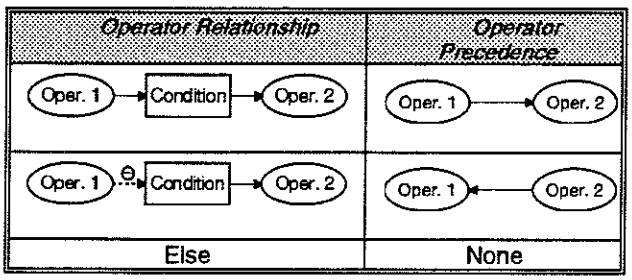


Figure 2: Precedence of execution between operators.

3.3 Identify whether the addition of the *object plan* creates loops in the *global plan.* If this is the case, the system returns to step 3.1 to find a new way of merging the *object plan* with the global plan.

3.4 If the current node in the strategy layer is a leaf, identify whether the *global plan* is complete by checking if all object states in the plan have the appropriate *cardinality*[2]; if this is the case, return and indicate "success", the *global plan* is correct and complete; otherwise, go to setp 3.1. If the current node in the strategy layer is not a leaf, go to step 1.1.

EXAMPLE

The behavior of the system will be illustrated by solving for the Hanoi-3 problem. This problem requires the system to include more than one instance of an operator in the global plan and has been used to describe the behavior of other planning systems such as ABTWEAK [8,9]. Figure 3 shows the operators used for the problem; the problem consists of moving the three rings (Big, Medium and Small) from peg 1 to peg 3 without putting a bigger ring on top of a smaller one.

[2] The *cardinality* of an object state is defined as the number of operators belonging to the plan that add it to the world model, minus the number of operators that delete it. In an appropriate plan solution, all the initial formulas that are not part of the final state have to have a cardinality of minus one; all the final formulas that are not part of the initial state have to have a cardinality of plus one, and all the rest have to have a cardinality of zero.

PLANRIK solves the problem as follows:

- Based on the ring order obtained by analyzing the table of Figure 3 (Big→ Medium→ Small), the object tree of Figure 4 is obtained. From the root node, the system chooses node B and asks the medium layer to create an object plan for this ring.
- At this point, the NROS for object B is empty since the global plan is also empty. The linear plan for object B consists of only one operator: MoveBig(1,3) (written as mB13). Note that this operator is not feasible at this point, since ring B has rings M and S above it in the initial situation, and therefore cannot be moved. The system, however, ignores this, since it is only concerned about finding a plan for object B (regardless of any other object or predicate).

Preconditions	*Operator*	*Effects*
(Big on pegx) ¯(Medium on pegx) ¯(Medium on pegy) ¯(Small on pegx) ¯(Small on pegy)	MoveBig (pegx,pegy)	(Big on pegy)
(Medium on pegx) ¯(Small on pegx) ¯(Small on pegy)	MoveMedium (pegx,pegy)	(Medium on pegy)
(Small on pegx)	MoveSmall (pegx,pegy)	(Small on pegy)

Figure 3: Operators for the Hanoi-3 problem.

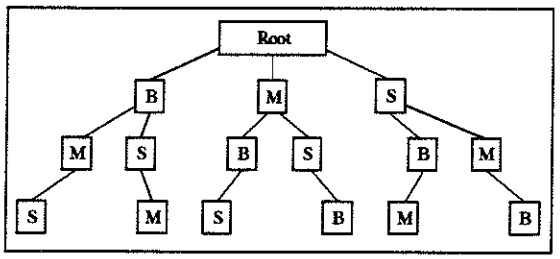


Figure 4: Tree of Objects for the Hanoi-3 Problem

- After completing this plan successfully, the top layer continues traversing the objects' tree in a depth first manner, choosing node M and asking the medium layer to create an object plan for this ring.
- At this point, the NROS for object M has only three nodes: the initial node M1 representing the initial position of the medium ring, the final node M3, and the intermediate node M2, obtained by looking at those preconditions in the global plan (see Figure 5) that refer to this ring. Therefore, the system searches for a linear object plan that takes ring M from peg 1 to peg 2 and then to peg 3. The resulting plan is mM12→ mM23.

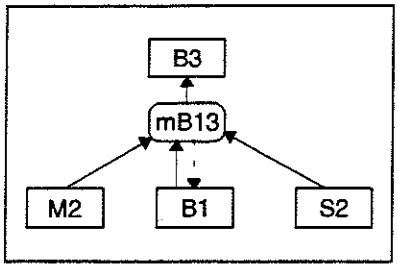


Figure 5: Global plan after obtaining the object plan for the Big ring.

- PLANRIK merges this plan with the current global plan, obtaining the new global plan shown in Figure 6. Since merging was successful, control returns to the top layer. The top layer continues traversing the objects' tree, choosing node S and asking the medium layer to create an object plan for this ring.

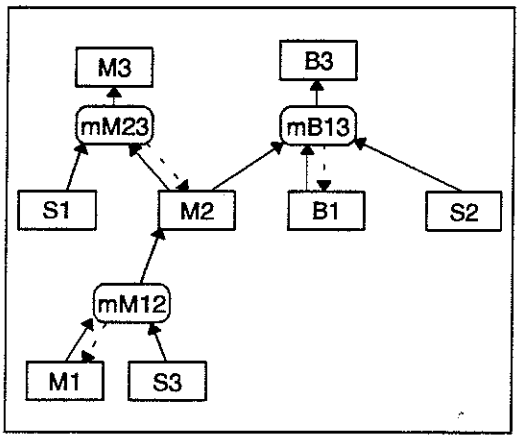


Figure 6: Global plan after merging the object plans for the Big and Medium rings.

- The NROS for object S is obtained from the global plan by analyzing the precedences between the operators directly connected to predicates referring to this ring. Since operator mM12 asserts the predicate M2, and this predicate is a precondition of operator mB13, a precedence from mM12 to mB13 is established. Similarly, precedence mM12→ mM23 is obtained. Finally, since operator mM23 negates predicate M2 which is a precondition of operator mB13, precedence mB13→ mM23 is interpreted. The resulting network of operators is shown in Figure 7.

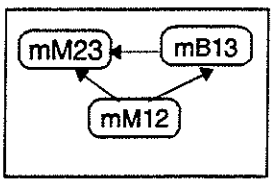


Figure 7: Operator precedences obtained by interpreting the Global plan of Figure 6

- Using this network of operators, the NROS for object S is obtained. Since S1, S2 and S3 are each a precondition of operators mM23, mB13 and mM12 respectively, the resulting NROS is similar to the network of operators, as shown in Figure 8. This NROS leads to only one possible sequence of object states that do not violate the precedence constraints: S3→ S2→ S1.

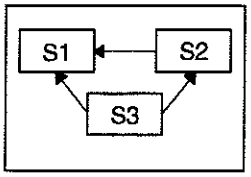


Figure 8: Network of required object states for the Small ring, obtained by interpreting the Global plan of Figure 6

- Now PLANRIK tries to obtain a linear plan for ring S that would change the position of ring S from its initial location (S1) to the desired one (S3), but that also includes all the intermediate states belonging to the sequence S3→ S2→ S1. The resulting plan is mS13→ mS32→ mS21→ mS13. Note that in this plan, operator mS13 is applied twice; also, that a more straightforward plan (i.e. mS13) is discarded since it does not satisfy the requirements expressed in the current NROS for this ring.
- The system merges this plan into the Global plan and obtains the final Global plan for the problem shown in Figure 9.

Finally, PLANRIK interprets the operator precedences from the Global plan by looking at the neighbors of each operator. This process yields to the nonlinear operator plan shown in Figure 10, which is the final solution of the Hanoi-3 problem.

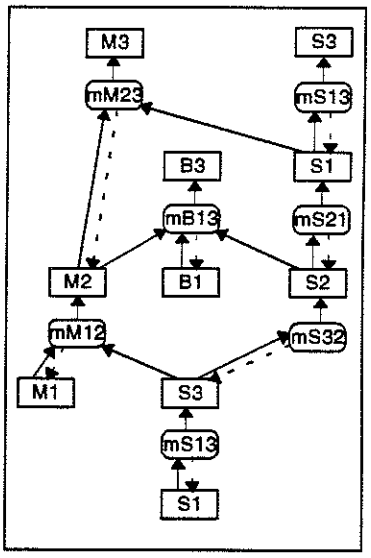


Figure 9: Global plan after merging the object plans for the three rings.

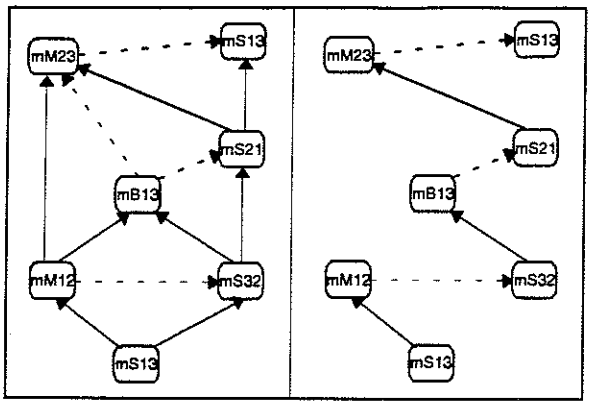


Figure 10: Interpretation of the Global plan of Figure 9 and final solution to the problem.

DISCUSSION

The paper discussed the architecture and behavior of PLANRIK, a planning system that solves problems with conjunctive goals by merging a set of linear *object plans* obtained separately to create a *global plan.* In obtaining these object plans, the system ignores information related to objects different from the one being considered. These object plans are not designed freely, however, since the current solution (i.e., the global plan) might impose some constraints about which *object states,* and in which order, have to be included in a particular *object plan.* This information is obtained from the *global plan* by locally interpreting precedences between *operators*, as well as between the *logical formulas* corresponding to a domain object.

PLANRIK has the following features: a) the system decomposes the problem into smaller subproblems, one for each object; b) abstraction is implemented not by preassigning a *criticality index* to the formulas describing the world, but by ignoring the information about other objects when solving for a particular one; c) *reordering of operators* occurs as a consequence of replacing a partial plan with another of the same object, and not by using *critics, tick-lists* or *white-knights* such as those found in NOAH [5], INTERPLAN [7] or TWEAK [1]; and d) the generation of a partial plan for an *object* is guided solely by a set of *requirements* representing intermediate object states that have to be accomplished in such partial plan. These features distinguish the manner how PLANRIK solves a particular problem from other planning systems using a similar representation of operators and states.

The paper described how PLANRIK solves the Hanoi-3 problem. In this example, the order specified in the first branch of the object tree happened to lead to the final solution without having to do backtracking. This, indeed, might not be the case in other problem domains. However, such a case only affects the efficiency but not the effectiveness of the system: if a different path would have been chosen at the beginning, the system would still have obtained the same solution to the problem by backtracking to the successful path.

References

[1] Chapman, D., *"Planning for Conjunctive Goals,"* Artificial Intelligence, Vol. 32, pp. 333-377, 1987.

[2] Corkill, D., *"Hierarchical Planning in a Distributed Environment,"* Proceedings of the Sixth International Joint Conference of Artificial Intelligence, Tokyo, pp. 168-175, August, 1979.

[3] Fikes, R.E., and Nilsson, N.J., *"STRIPS: A new approach to the Application of Theorem Proving to Problem Solving,"* Artificial Intelligence, Vol. 2, No. 4, pp. 189-208, 1971.

[4] Sacerdoti, E.D., *"Planning in a Hierarchy of Abstraction Spaces",* Proceedings of the Third International Joint Conference on Artificial Intelligence, Stanford, CA, pp. 412-422, 1973.

[5] Sacerdoti, E.D., *"A Structure for Plans and Behavior,"* Elsevier-Holland, New York, 1977.

[6] Sussman, G.A., *"A computational model of skill acquisition",* Technical report number AI-TR-287, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1973.

[7] Tate, A., *"INTERPLAN: A Plan Generation System Which Can Deal With Interactions Between Goals",* Technical Report MIP-R-109, Machine Intelligence Research Unit, University of Edinburgh, 1974.

[8] Woods, S.G., *"An Implementation and Evaluation of a Hierarchical Nonlinear Planner,"* unpublished Master's Thesis, Department of Mathematics, University of Waterloo, Ontario, Canada, 1991.

[9] Yang, Q. and Tenenberg, J., *"ABTWEAK: Abstracting a nonlinear, least commitment planner,"* Proceedings of the Eighth Conference of the American Association for Artificial Intelligence, 1990.

[10] Zozaya-Gorostiza, C.A., Hendrickson, C.T. and Rehak, D.R., *"Knowledge-Based Process Planning for Construction and Manufacturing",* Academic Press, San Diego, CA, 1989.

A MARKER PROPAGATION TEXT UNDERSTANDING AND INFERENCE SYSTEM

Dan I. Moldovan
Southern Methodist University
Dept. of Computer Science & Engineering
Dallas, TX 75275
moldovan@seas.smu.edu

Sanda M. Harabagiu
University of Southern California
Dept. of Electrical Engineering-Systems
Los Angeles, CA 90089-2562
harabagi@usc.edu

Abstract

This paper introduces a system intended for deep language understanding and inference. The system uses a large knowledge base structured around WordNet. Semantic paths between text concepts are established in the knowledge base with the help of the marker propagation method. These paths bring forward inferences and implicatures that otherwise are difficult to extract.

1 SYSTEM OVERVIEW

Introduction

This paper describes a natural language understanding system that is under development at the Southern Methodist University. The project, known as the SNAP project, started at the University of Southern California in late 1980's. Our group participated in the Message Understanding Conferences (MUC-4 and -5) where the task was to extract information from thousands of real messages and fill in some predetermined templates. In this paper we describe the *understanding* and *inference* components of our system.

A distinct feature of our system and an important innovation is the application of parallel processing to natural language. Another significant aspect is that we take a complete system approach that integrates many levels of language processing and aim at understanding unrestricted text. To facilitate the processing of unrestricted text, we use a knowledge base constructed on top of WordNet 1.5, the lexical database for English language developed at Princeton[Mil95].

Text understanding and inference

The understanding of natural language processing involves complex tasks such as lexical and semantic disambiguation, pragmatic reasoning for coherence and implicature recognition, anaphora and co-reference resolution. Each of these problems are AI-complete. That is, a system can reliably process text correctly only when much of real-world knowledge is encoded, enabling inferences humans are capable of.

Given a very large lexical and semantic knowledge base, like WordNet, which encompasses the vast majority of English words, we aim at integrating common-sense reasoning mechanisms with semantic disambiguators and frame constructors.

One of the first tasks is to determine the acceptable lexical and syntactic structures for each sentence. Prepositional attachments are solved immediately after the parse is produced. Our system uses a body of heuristics operating on WordNet, that are synthesized from experiments conducted over the Treebank corpora [Har96]. Further on, transformational grammars account for the recognition of semantic case functions. Semantic disambiguation is performed as inference which relates concepts with minimal semantic distance in WordNet [MiT91].

At this point, the system is ready to provide the explicit inferences implied by the text. Nevertheless, they still have to be enforced by the implicit inferences, supporting coherence evaluation and context determination. Almost all inferences, from disambiguation and prepositional attachments to discourse implicatures rely heavily on WordNet information.

In this paper we describe a solution for these problems that uses an encoded common-sense knowledge base which is publicly available. The inferences drawn on such a knowledge base show that human-like text understanding capabilities may be achievable. The evaluation of MUC-3, MUC-4 and MUC-5 results [Sud93] demonstrate that higher scores of recall and precision are not achievable without implementing reasoning mechanisms that rely on human common-sense knowledge. We present here our efforts to maximize our system's efficiency by dealing with world knowledge.

The basic idea

Our approach to NLP is to use a very large linguistic knowledge base and to consider language understanding as a search process for finding from the knowledge base the interpretations of the information conveyed by the text. The processing agents are markers, or data structures, that may be visualized as moving through the semantic network knowledge base according to user defined propagation rules.

The main advantages of this approach are: (1) it offers the possibility of exploiting the inherent parallelism in natural language, and (2) the same knowledge base and propagation mechanism are used at all levels of processing i.e., syntactic, semantic, pragmatic and discourse.

The approach we use is abductive interpretation [HSA92] integrating constraint satisfaction and search in a very large knowledge base. Constraints are implemented as links of the network and as marker processing functions. Markers carrying linguistic information check constraints, and interact with some other markers inside the nodes, and produce new markers that satisfy the constraints. The source of parallelism in this approach is simultaneous marker processing and propagation through the network that implements the linguistic knowledge base.

In general, each stage in our system performs its processing using marker-propagations, as a result of this processing certain markers pick up more information and become fatter. The output of each stage is the set of these fattened markers which arrive on a well defined set of nodes in the knowledge base. Each succeeding stage, in the course of its processing, selectively determines which information to use from the previous stages and how it shall be used.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/55

2 MARKER-PROPAGATIONS

Marker-propagation networks
The marker-propagation computational paradigm is especially suitable for applications where control flow is incompletely specified. NLU falls into this broad category. The marker propagation model can be briefly described as a network consisting of *nodes* and *links*, and a set of *markers* moving through the network according to some propagation rules. The marker propagation scheme designed by us is by far more complex than the ones previously proposed by Charniak [Cha86], Hendler [Hen88] and Norvig [Nor87].

One of the differences is that marker propagations are dependent on the processing of their *propagation rules*, which interact with the predicate values of the concepts they reach in the knowledge base, as well as with their constraint arguments. This reduces the number of nodes marked and the number of inferences drawn from the semantic paths devised by marker trajectories.

The nodes can independently execute a set of functions, store data, and communicate with other nodes. The links are directional, and may have associated functions that accommodate marker arguments whenever they are transversed. This is the second novelty of our system, and it supports our intuitions about world processes and phenomena. For example, an *ISA* link connecting the verb concept `hit#1v` with `propel` enables the propagation of the *object* argument, by having the function *same_object* attached to it. Therefore, the predicate *hit(ball)* implies *propel(ball)* whenever a marker is propagated from `hit#1v` to `propel`, and the marker carries along the object role filler `ball`. Similarly, when a function *change*($func_1$, $func_2$) is attached to a link, the semantic case function of the marker is altered from $func_1$ to $func_2$, although its value is maintained. For example, the *ISA* link between `hit#1v` and `move#1v` has a function *change(object, experiencer)*. This implements our intuition that a ball, the object of a `hit#1v` action is transfered as an experiencer of the `move#1v` process.

Marker propagation and control
A central question is how to control the reasoning process determined by marker propagations to avoid generation of already known facts or irrelevant facts. Since many of our inference rules, implemented as propagation rules, are linked to abduction, the role of the control strategy is increased. Given the large size of the linguistic knowledge base, computation efficiency becomes one of the dominating criteria. Due to the programmability of the propagation rules, the system generates a manageable number of paths. We have noticed that the shorter paths are the most useful, while some of the longer paths bring irrelevant inferences. A solution to filter out irrelevant paths, and even to rank the good paths, is to propagate syntactic and case constraints along with the markers. Whenever such restrictions are not satisfied, the respective propagation ceases. Any colliding marker trajectory indicates a valid semantic path in the knowledge base.

3 SYSTEM ARCHITECTURE

Block diagram
Figure 1 gives a functional representation of our system. The knowledge sources consisting of on-line dictionaries and WordNet, are shown at the left, the components of the linguistic knowledge base are shown in the center, and the components of the run-time system are shown at the right. A large part of this system was developed as an information extraction system for the Message Understanding Conference [Sud93]. The current system aims at high level inferences, provided by the last module of the system.

The preprocessor locates sentence boundaries and inserts sentence boundary markers for the syntactic parser. Phrase tagging at this point is meant to reduce the parser's work load. Our syntactic

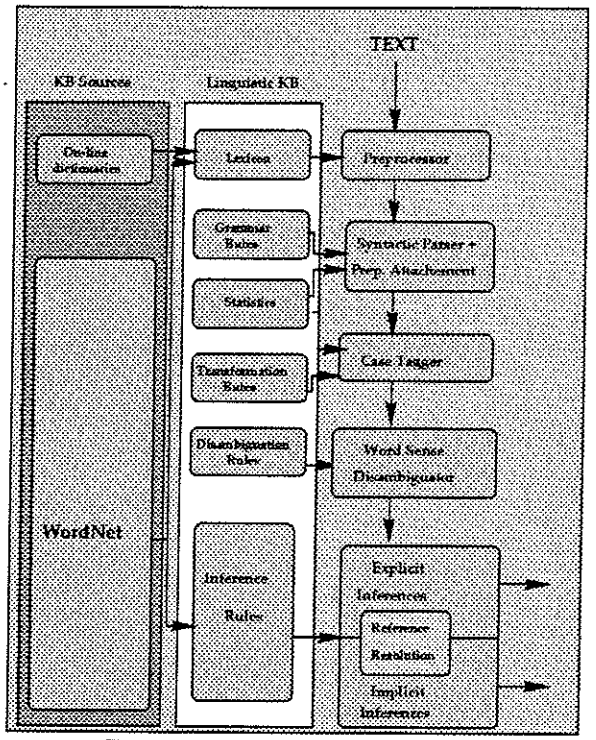


Figure 1. System organization

parser, a parallel version of a LR(0) shift-reduce parser [Hen95] recognizes the syntactic structure of an input sentence and produces markers for noun groups, verb groups and prepositional phrases. Many of the lexical ambiguities are resolved in this module. The output consists of larger markers that carry the accepted syntactical structure of a sentence. When there are syntactical ambiguities, multiple competing candidates are produced and passed to the semantic interpretation level. The parser performes especially well on very long sentences, which are typical in wired-news.

The semantic processing module takes as input a full syntactic parse of a clause and outputs a disambiguated case frame representation of that clause. The mapping is produced by integrating transformational grammar productions with semantic constraints devised from WordNet.

At this point, explicit inferences can be finalized, by producing the proper connections between relative or embedded clauses and primary clauses and by partially solving reference resolution. However, we reinforce explicit inferences by the implicit connections between the textual concepts that can be found in the commonsense knowledge base. These connections account for the global and local coherence of the text and are capable of providing the context in which the text in interpreted. They also finalize reference resolutions.

Text processing
An illustration of a typical text processed by the system is:

```
<p>
<s1> The 5-foot-8, 145-pounder's injuries included
tendinitis in his shoulder and elbow. </s>
<s2> It started when he was riding high, and has
persisted. </s>
<s3> "I guess I played too much too soon. </s>
<s4> I was a skinny kid who hit the ball very hard. </s>
<s5> Maybe too hard, it landed almost always near
the backline" he says. </s>
</p>
```

This is a paragraph from a Treebank Wall Street Journal article reporting tennis-related events at the U.S. Open. The text provides a fine example for need of non-trivial reference resolution as well as the necessity for meaningful implicatures related to the side-effects of tennis professional practice.

Next we present the knowledge base and the inference engine used in our system, after which we discuss how explicit and implicit inferences are obtained. We exemplify on the above text some reasoning mechanisms which produce explanations for reference resolution, context recovery and implicature understanding.

4 STRUCTURE OF A VERY LARGE LINGUISTIC KNOWLEDGE BASE

WordNet and extensions

Realistic natural language processing requires extensive knowledge. While researchers agree that large, scalable knowledge bases are needed [GPW95], [Jac92], [Len95], there is little agreement on how to structure such knowledge bases.

We decided to build our knowledge base on top of WordNet1.5, the lexical database developed at Princeton. WordNet was built as a semantic dictionary, in which words are searched based on conceptual affinity with other words. It covers the vast majority of nouns, verbs, adjectives and adverbs from the English language. The words in WordNet are organized in synonym sets, called *synsets*. Each synset represents a concept. There is a rich set of relation links between words and other words, between words and synsets, and between synsets. A large linguistic knowledge base may be developed starting from WordNet.

All the words with the same meaning are connected to a concept node representing that meaning. A word may belong to several synsets that correspond to the various meanings of that word.

The noun and verb concepts are structured into hierarchies by using ordering relations such as *isa*, *is_part*, or *is_member*. The adjectives and adverbs don't have hierarchies, instead are linked into clusters. Verb concepts are connected by semantic relations reflecting common-sense causality: the *entail* and *cause_to* relations. In WordNet, typical features of nouns are represented by *attribute* relations, connecting noun concepts to adjective concepts. *Pertain* relations connect words to the nouns describing typical situations in which their use is accountable.

Another characteristic of WordNet is that almost *90%* of the represented concepts have textual glossaries attached to them. Glosses contain representative information about the concepts, defining typical microcontexts in which the concepts operate. The glosses make explicit the functional relations to other concepts from WordNet, enhancing the degree of connectivity by an order of magnitude.

Our extension to WordNet is the transformation of each concept's gloss into a defining feature directed acyclic graph with nodes and links. This enables our system to obtain semantic paths between pairs of concepts by integrating various types of semantic information readily available in WordNet.

Table 1 shows the number of words and concepts for each part of speech. In its current version WordNet 1.5 has 168,217 words organized in 91,595 synsets, thus a total of 259,812 nodes.

Semantic relations

The knowledge base relations, that link various nodes, are the building blocks to our solution for text inference. Altogether there are 345,264 links in the knowledge base, thus on average there are approximately 1.5 links per node. There are 17 relation types.

The relations that are asymmetric and transitive are more useful for inference than others since often they support deductions. For example, going up an isa hierarchy of nouns or verbs, takes us from more specific to more general concepts and inferences make sense. The same for some of the verb relations such as entail or causation. For example, if a man is snoring, than the same man is sleeping, and further he is temporarily unconscious. However, inferences sometimes need to go against these transitive relations, and conclusions may be false. For example, if one is sleeping, we don't know for sure that one is snoring, but it is plausible. In these cases we perform abduction, which is providing the best explanation for the logical connection implied by the text.

Part of speech	words	concepts
noun	107,484	60,560
verb	25,768	11,364
adjective	28,762	16,428
adverb	6,203	3,243
Total	168,217	91,595

Table 1. Knowledge base nodes

Our extension of WordNet introduces 18 new relations to the 17 original ones. Each defining feature (represented in Figure 2 as white blocks) is accessible from the concepts along a *DF_isa* relation. Inside the defining feature graph, the concepts are linked through *DFC*(defining feature continued) relations, differentiated by the semantic case functions they represent in the gloss. We devised *DFC_agnt* relations for agents connected to corresponding actions in the glosses, *DFC_obj* between actions and their objects or *DFC_instr* to link actions to their instruments. Attributes connections within glossary phrases are represented as *DFC_attr* relations. Many verbs have the goal of their actions explicitly stated in the glossary. the relations *DFC_purpose* links actions to their goals in the defining features.

Logical connections between textual concepts

We developed a parallel marker propagation algorithm [HM95] that establishes the logical connections between pairs of textual concepts. Filtered by a variety of linguistic constraints [HMY96], the semantic paths bring forward the implicit inferences implied by the text. The textual context is also recovered along this process. Context is produced as a cohesion of the defining features of the concepts found along the paths, providing relevant pragmatic information.

5 HIGH LEVEL INFERENCES

Explicit inferences

The system detects explicit inferences from the text by the recognition of prepositional attachments and semantic case relations. In sentence <s1>, the noun phrase `'' the 5-foot-8, 145-pounder's injuries''` is interpreted as `''injuries of the 5-foot-8, 145-pounder''`. The semantics of this prepositional attachment reveals that `injuries`, a nominalization of the verb `to injure`, are performed upon a `5-foot-8, 145-pounder`, therefore `5-foot-8` and `145-pounder` are coreferring objects of the verb `to injure`. At this point, we do not know whether it is a human or any other living being. The first sense from WordNet of the verb `to include` admits multiple objects, but we take the including object (in this case, `injuries`) to be the agent which reflects the state `to include`, whereas `tendinits` is the object which is `included`.

The prepositional attachment `tendinitis in his shoulder and elbow` reveals that the `shoulder` and the `elbow` are the body parts where tendinitis is observed. Figure 2 illustrates the fact that in WordNet, `shoulder`, `elbow` and `tissue` are represented as `body parts`.

In sentence <s2>, the pronoun `it` is the agent of the action `start#4v`, as well as for the state `persist`, when the ellipsis of

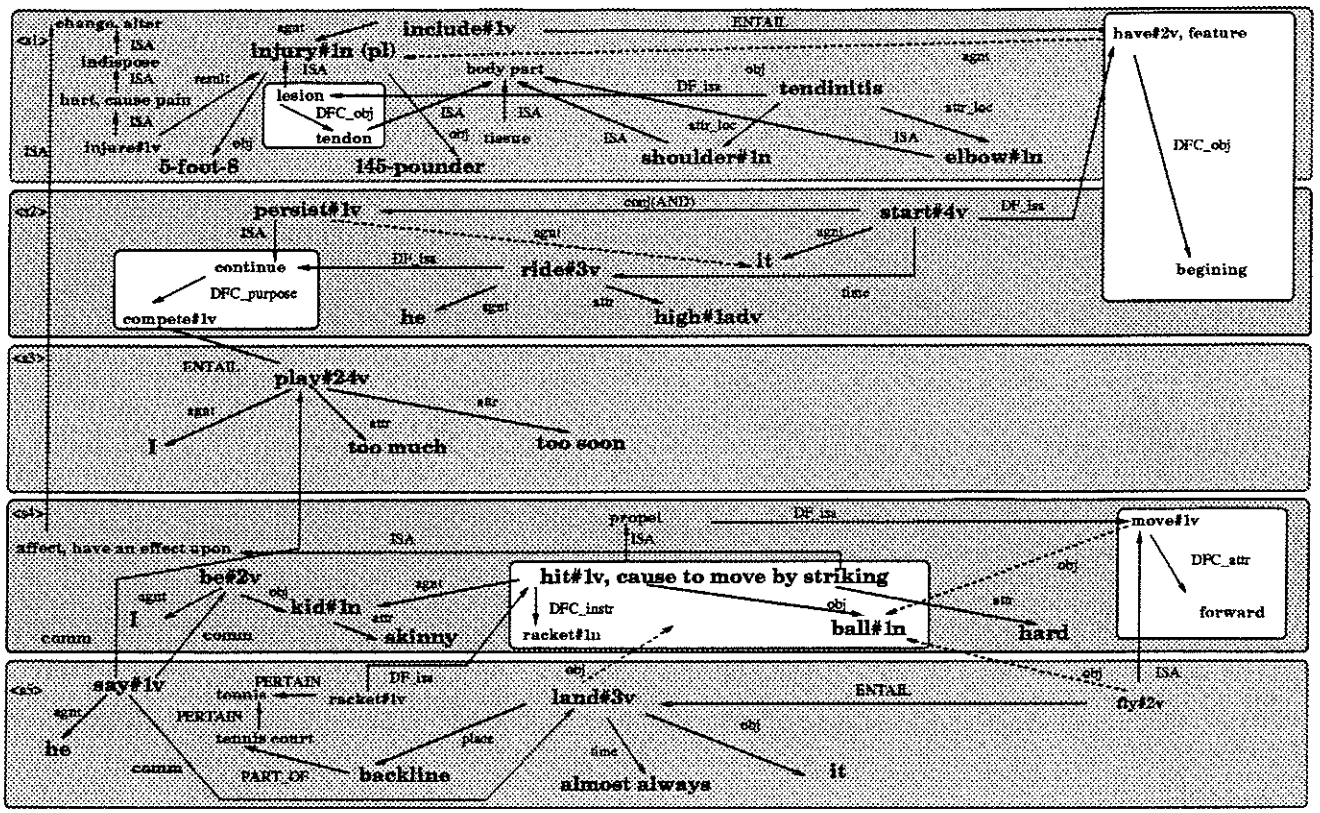


Figure 2. Semantic paths in a real world text

the conjunctive clause is solved. The `start#4v` action is marked by the time span when the agent `he` was `riding high`.

In sentence <s3>, the agent of `play#24` is the pronoun `I`, and the attributes of the action are marked by the scaled adverbs `too much` and `too soon`.

The pronoun `I` in <s4> is identified to the `kid#1n`, with attribute `skinny`. The cohesion of the relative clause takes place by transferring `kid#1n` as an agent for the action `hit#1v, cause to move by striking`, whose object is `the ball#1n`.

The last sentence <s5> ends the communication initiated by the referent of the pronoun `I`, who uttered the last three sentences. Therefore, the *communication* relations of the verb `say#1v` point to the main verbs from <s3>, <s4> and <s5>. The object of action `land#3v` is the pronoun `it` and the location is the `backline`, while the time is marked by the adverbial phrase `almost always`.

Implicit Inferences

Now we show how our system is able to provide an explanation for the text coherence and the relative implicatures. By providing such an explanation, one can infer a number of assertions unstated in a text.

We have developed a marker propagation algorithm that finds coherence paths between a pair of clauses. The algorithm places markers on some key concepts, and the markers propagate through the knowledge base according to predefined inference rules. Many paths are explored in parallel.

Several paths are relevant for the understanding of this text. We shall illustrate only six of them , that account for global and local coherence. They are:

```
Path 1    include#1v(injuries)->ENTAIL-> [COLLISION]
have#2v,feature(->DFC_obj->beginning)<-DF<-start#4v

Path 2    play#24v->ENTAIL->compete#1v [COLLISION]
compete#1(<-DFC_purpose)<-DF<-ride#3v

Path 3    hit#1v->ISA-V->propel->DEF_FEAT [COLLISION]
move#1v<-ISA-V<-fly#2v<-r_ENTAIL<-land#3v

Path 4    backline#1n->PART-OF->tennis court->PERTAIN->
[COLLISION]tennis#1n<-PERTAIN<-racket#1v<-r_DF<-hit#1v

Path 5    persist#1v->ISA-V->continue [COLLISION]
continue(->DF_purpose->compete)<-DF<-ride#3v

Path 6    tendinitis#1n->DF>lesion->DFC_obj->tendon->
->ISA-n->body part [COLLISION]<-ISA-N<-shoulder,elbow
```

The semantic connections provided by `Path 1` relate the verb `include#1v` (whose agent are the `injuries`) with the verb concept `have#2, feature` through an entailment relation. The verb `start#4v` from <s2> has defining feature `have#2v` with defining feature object (`DFC_obj`) `beginning`. Therefore, uttering in <s1> about the inclusion of injuries, entails an implication about when those injuries started (communicated in <s2>).

In `Path 2`, the entailment between `play#24` from <s3> and `compete#1v`, an element of the defining features of `persist#1v` (of <s2>) explains the persistent nature of the injuries as determined by playing (too much and too soon).

`Path 3` conveys further implicatures, detailing the specifics of the game that caused the injuries. The `kid` was `hitting` the `ball` very hard. `Hitting` *isa* `propelling`, with defining feature `move#1v` (and attribute `forward`). The collision of `Path 3` at

move#1 brings information about the ball that flew through the air before landing (land#3 is entailed by fly#2, a form of move#1v).

Path 4 reveals the context of the declarations. The backline, the location where the ball used to land, is part of a tennis court, a concept pertaining to the tennis game. Hitting a ball with a racket in tennis defines the features of the racket action.

Temporal reasoning related to the understanding of the text is motivated by Path 5. The injury process (with the agent co-referred by the agent of persist) is concurrent with the competing activities pursued by the speaker. This explains the local coherence of <s2>.

The local coherence of <s1> is accommodated by Path 6, which identifies the location attributes of tendinitis, as described by its defining feature, the tendon in the space of body parts, which is shared by shoulder and elbow too.

Reference Resolution

Reference resolution is the byproduct of combined implicit and explicit inferences. This text displays several types of reference: (a) pronominal anaphorae: it from <s2> and <s5>, I from <s4> and <s4> and he from <s2> and<s5> and (b) definite reference for the backline from <s5> and the ball from <s4>.

Case propagations produced by the path-finding algorithm account for the resolution of these types of reference. From Path 1, the agent of start is propagated as an agent of have#2v, where it is unified with the agent of include, i.e. the injuries. Case propagations prevail over syntactic restrictions, because although there is a preference for a singular referent (to provide agreement with the number of pronoun it), the plural form of the referent is acceptable, and the resolution counts for a prosentential reference, where it refers to the whole class of injuries which constitutes the topic of <s1>.

Path 2 unifies the pronoun he from <s2> as an agent of ride with the agent of play#24 from <s3>, which is the speaker, represented by the pronoun I in <s3> and <s4> and by he in <s5>. The agent of ride#3v is explicitly referred in the text as a skinny kid, which hit the ball as part of his tennis game (inferred by the context generating Path 4). Because the activities of the referent of I, he and skinny kid are concurrent with the injuries suffered by the 5-foot-8, 145-pounder, the system concludes that he and I refer to a person (capable of communication, as deduced from <s3>, <s4> and <s5>), that is a tennis player (from Path 3 and Path 4) with the combined attributes skinny kid, 5-foot-8 and 145-pounder.

The backline of <s5> is identified as a part of a tennis court, the spatial context in which the tennis game takes place, as revealed by Path 5. Similarly, the ball from <s4> is recognized as a typical artifact used in the tennis game.

6 Conclusions

There is a growing trend to use electronic dictionaries to construct large linguistic knowledge bases. In this paper we argued that WordNet provides a platform on which powerful, generally applicable knowledge bases can be built. Although WordNet does not provide information regarding the conditions for actions and events that may be necessary to fully explain a text, it nevertheless contains almost all English words mapped into concepts and a large number of links between concepts. We found this to be an ideal setting on which to apply marker propagations to draw inferences. More information about inference rules that may be constructed by chaining relations can be found in [HMY96]. The inference method proposed here is still very simplistic because it cannot rank the semantic paths it finds and it does not take into account enough linguistic constraints. These are not inherent limitations and can be resolved.

References

[Cha86] E. Charniak. A neat theory of marker passing. In *Proceedings of the Fifth National Conference on Artificial Intelligence AAAI-86*, pages 584–588, 1986.

[Cha95] E. Charniak. Natural Language Learning. *ACM Computing Surveys*, vol 27: No3, pages 317–319, September 1995.

[COL88] P.R. Cohen and C.L. Loiselle. Beyond ISA: Structures for Plausible Inference in Semantic Networks. In *Proceedings of the Seventh National Conference on Artificial Intelligence AAAI-88*, pages 415–420, 1988.

[GPW95] L. Guthrie, J. Pustejowsky, Y. Wilks and B. Slator.. The Role of Lexicons in Natural Language Processing *Communication of the ACM*, vol 39: No1, January, 1996.

[Har96] S.M. Harabagiu. An Application of WordNet to Prepositional Attachment. To apear in the *Proceedings of ACL-96*, 1996.

[HM95] S.M. Harabagiu and D.I. Moldovan. A Marker-Propagation Algorithm for Text Coherence. In *Working Notes of the Workshop on Parallel Processing for Artificial Intelligence, IJCAI-95*, pages 76–86, 1995.

[HMY96] S.M. Harabagiu, D.I. Moldovan and T.Yukawa. Testing Gricean constraints on a WordNet-based Coherence Evaluaiton System. In *Working Notes of the AAAI-96 Spring Symposium on Computational Approaches to Interpreting and Generating Conversational Implicature*, pages 31–38, 1996.

[Hen88] J.A. Hendler. Intergating Marker–Passing and Problem Solving : A Spreading Activation Approach to Improve Choice in Planning. Lawrence Erlbaum Associates, 1988.

[Hen95] K.J. Hendrickson. A New Parallel LR Parsing Algorithm. In *Proceedings of the 1995 ACM Symposium on Applied Computing*, pages 115–120, 1995.

[HSA92] J.R. Hobbs, M. Stickel, D. Applet and P. Martin. Interpretation as Abduction. *Artificial Intelligence*, vol 63:pages 69–142, 1993.

[Jac92] P.S. Jacobs. Text Power and Intelligent Systems. In Lawrence Erlbaum Associates, *Text–Based Intelligent Systems*, pages 1–8. ,editor Jacobs, P.S., Hillsdale, N.J., 1992.

[Len95] D.B. Lenat. CYC: A Large-Scale Investment in Knowledge Infrastructure. *Communication of the ACM*, vol 38: No11, pages 32–38, November, 1995.

[Mil95] G.A. Miller. WordNet: A Lexical Database. *Communication of the ACM*, vol 38: No11, pages 39–41, November, 1995.

[MiT91] G.A. Miller and D.A. Teibel. A Proposal for Lexical Disambiguation. In *Proceedings of the ARPA Speech and Language Workshop*, pages 395–399, February,1991.

[Nor87] P. Norvig. Inference in text understanding. In *National Conference on Artificial Intelligence AAAI-87*, pages 561–565, 1987.

[Sud93] B. Sundheim. Proceedings. *Fifth Message Understanding Conference (MUC-5)* August, 1993.

Trading off Completeness for Efficiency — The PARSETALK Performance Grammar Approach to Real-World Text Parsing

Peter Neuhaus Udo Hahn

Freiburg University
Computational Linguistics Lab
Europaplatz 1, D-79085 Freiburg, Germany
{neuhaus,hahn}@coling.uni-freiburg.de

Abstract

We argue for a performance-based design of natural language grammars and their associated parsers in order to meet the constraints posed by real-world natural language understanding. This approach incorporates declarative and procedural knowledge about language and language use within an object-oriented specification framework. We discuss several message passing protocols for real-world text parsing and provide reasons for sacrificing completeness of the parse in favor of efficiency.

1 INTRODUCTION

Over the past decades the design of natural language grammars and their parsers was almost entirely based on *competence* considerations (Chomsky, 1965). These hailed pure declarativism (Shieber, 1986) and banned procedural aspects of natural language use out of the domain of language theory proper. The major premises of that approach were to consider sentences as the primary linguistic object of investigation, to focus on syntactic descriptions, and to rely upon perfectly well-formed utterances for which complete grammar specifications of arbitrary depth and sophistication were available. In fact, promising efficiency results can be achieved for parsers under corresponding optimal laboratory conditions. Considering, however, the requirements of natural language *understanding*, i.e., the integration of syntax, semantics, and pragmatics, and taking *ill-formed* input or *incomplete* knowledge into consideration, the processing costs either tend to increase at excessive rates or linguistic processing even fails completely.

As a consequence, the challenge to meet the specific requirements imposed by real-world texts has led many researchers in the NLP community to re-engineer competence grammars and their parsers and to provide various add-ons in terms of constraints (Uszkoreit, 1991), heuristics (Huyck & Lytinen, 1993), statistics-based weights (Charniak, 1993), etc. In contradistinction to these approaches, our principal goal has been to incorporate performance conditions already in the design of natural language grammars, yielding so-called *performance grammars*. Thus, not only declarative knowledge (as is common for competence grammars), but also *procedural* knowledge (about control and parsing strategies, resource limitations, etc.) has to be taken into consideration at the *grammar specification* level proper. This is achieved by providing self-contained description primitives for the expression of procedural knowledge. We have taken considerable care to transparently separate declarative (structure-oriented) from procedural (process-oriented) knowledge pieces. Hence, we have chosen a formally homogeneous, highly modularized object-oriented grammar specification framework, namely the actor model of computation which is based on concurrently active objects that communicate by asynchronous message passing (Agha, 1990).

The PARSETALK system whose design is based on these performance considerations (Hahn et al., 1994) forms part of a text knowledge acquisition system whose input comes from two domains, viz. test reports from the information technology field and medical findings reports. The analysis of texts (instead of isolated sentences) requires, first of all, the consideration of textual phenomena by a dedicated *text grammar* (local ones such as anaphora (Strube & Hahn, 1995), or even global coherence relations (Hahn, 1992)). Second, text understanding is based on the execution of inferences by which text propositions are integrated into the text knowledge base with reference to a canonical representation of the *background knowledge* of the underlying domain. This way, grammatical (language-specific) and conceptual (domain-specific) knowledge are closely coupled. Third, text understanding in humans occurs immediately and at least within specific processing cycles in parallel (Thibadeau et al., 1982). These processing strategies we find in human language processing are taken as hints how the complexity of natural language understanding can reasonably be overcome by machines.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/60 ©1996 FLAIRS

Thus, text parsing devices should operate *incrementally* and *concurrently*. In addition, the consideration of *real-world* texts (instead of researcher-generated material covered by complete specifications) forces us to supply mechanisms which allow for the *robust* processing of extra- and ungrammatical input. We take an approach where — in the light of abundant specification gaps at the grammar and domain representation level — the degree of underspecification of the knowledge sources or the impact of grammar violations directly corresponds to a lessening of the precision and depth of text knowledge representations, thus aiming at a sophisticated *fail-soft* model of *partial* text parsing.

2 THE PARSETALK GRAMMAR

The PARSETALK performance grammar contains fully *lexicalized* grammar specifications in terms of configurational constraints on word classes and morphological features as well as on word order and conceptual compatibility. Grammatical conditions of these types are combined in terms of *valency* constraints (at the phrasal and clausal level) as well as *textuality* constraints (at the text level of consideration), which concrete dependency structures and local as well as global coherence relations must satisfy. The compatibility of grammatical features (encapsulated by methods we refer to as SYNTAXCHECK) is computed by a unification mechanism, while the evaluation of semantic and conceptual constraints (we here refer to as CONCEPTCHECK) relies upon terminological reasoning in terms of subsumption. Thus, while the dependency relations represent the linguistic structures of the input, the conceptual relations yield the targeted representation of the text content (for an illustration, cf. Fig. 7).

In order to structure the underlying lexicon, *inheritance* mechanisms are used. Lexical specifications are thus organized along the grammar hierarchy at various abstraction levels, e.g., with respect to generalizations on word classes. Lexicalization of this form already yields a fine decomposition of declarative grammar knowledge. It lacks, however, an appropriate description equivalent at the procedural level. We therefore provide lexicalized communication primitives to allow for heterogeneous and local forms of interaction among lexical items.

Semantic and domain knowledge specifications are based on a common hybrid terminological, classification-based knowledge representation language (for a survey, cf. Woods & Schmolze (1992)). Ambiguities which result in interpretation variants are managed by a context mechanism of the underlying KB system (there is no distinction at the representational level between semantic and conceptual interpretations of texts).

Robustness at the grammar level is achieved by several means. Dependency grammars describe binary, functional relations between words rather than contiguous constituent structures. Thus, ill-formed input often does still have an (incomplete) analysis in our grammar. Furthermore, it is possible to specify words at different levels of syntactic or semantic granularity. The main burden of robustness, however, is assigned to a dedicated message passing protocol discussed in the next section.

3 THE PARSETALK PARSER

At the class level of object-oriented grammar specifications, we represent lexical items as *word actors*, which, when instantiated, are acquainted with other actors representing the head or modifiers in the current utterance.

A specialized actor type, the *phrase actor*, groups word actors which are connected by dependency relations and encapsulates administrative information about that phrase. Accordingly, a message does not have to be sent directly to a specific word actor, but will be sent to the mediating phrase actor which forwards it to an appropriate word actor. Furthermore, the phrase actor holds the communication channel to the corresponding interpretation context in the domain knowledge base system.

A *container actor* encapsulates several phrase actors that constitute alternative analyses for the *same* part of the input text (i.e., structural ambiguities). Container actors play a central role in controlling the parsing process, because they keep information about the *textually* related (*preceding*) container actors holding the left context and the *chronologically* related (*previous*) container actors holding a part of the head-oriented parse history.

Basic Parsing Protocol. We use a graphical description language to sketch the message passing protocol for establishing dependency relations as depicted in Fig. 1 (the phrase actor's active head is visualized by $\oplus$). A searchHeadFor message (and *vice versa* a searchModifierFor message if searchHeadFor fails to succeed) is sent to the textually preceding container actor (precedence relations are depicted by bold dashed lines), which simultaneously directs this message to its encapsulated phrase actors. At the level of a single phrase actor, the distribution of the searchHeadFor message occurs for all word actors at the "right rim" of the dependency tree (depicted by $\odot$). A word actor that receives a searchHeadFor message from another word actor tests whether a dependency relation can be established by checking its corresponding valency constraints (applying SYNTAXCHECK and CONCEPTCHECK). In case of successful evaluation, a headFound message is returned, the sender and the receiver are copied (to enable alternative attachments in the concurrent system, i.e., no destructive operations are carried out), and a dependency relation, indicated by a dotted line, is established between those copies which join into a phrasal relationship. For illustration purposes, consider the anal-

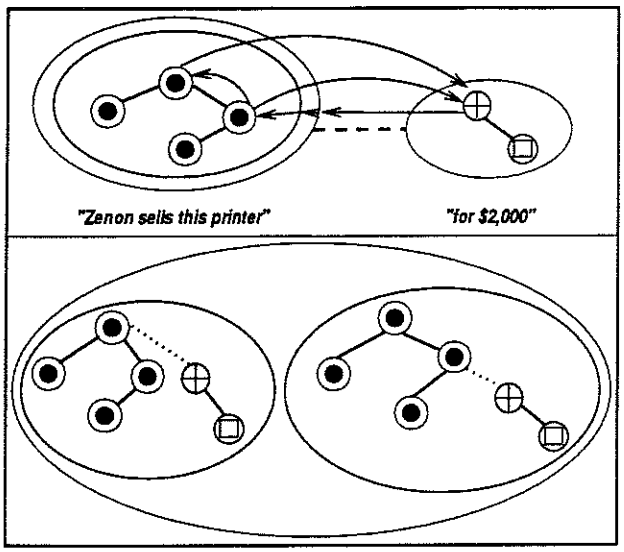


Figure 1: Basic Mode (incl. Structural Ambiguities)

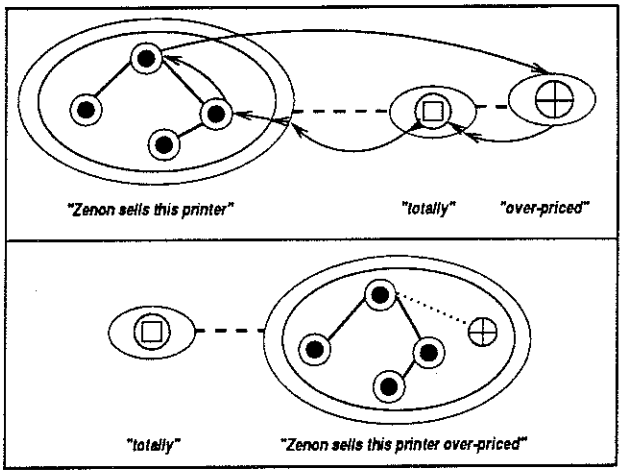


Figure 2: Skipping Mode

ysis of a phrase like *"Zenon sells this printer"* covering the content of the phrase actor which textually precedes the phrase actor holding the dependency structure for *"for $2,000"*. The latter actor requests its attachment as a modifier to some head. The resultant new container actor (encapsulating the dependency analysis for *"Zenon sells this printer for $2,000"* in two phrase actors) is, at the same time, the historical successor to the phrase actor covering the analysis for *"Zenon sells this printer"*.

The structural ambiguity inherent in the example is easily accounted for by this scheme. The criterion for a structural ambiguity to emerge is the reception of at least two positive replies to a single searchHeadFor (or searchModifierFor) message by the initiator (cf. Fig. 1). The basic protocol already provides for the concurrent copying and feature updates required. In the example from Fig. 1, two alternative readings are parsed, one phrase actor holding the attachment to the verb (*"sells"*), the other holding that to the noun (*"printer"*). The crucial point about these ambiguous syntactic structures is that they have conceptually different representations in the domain knowledge base. In the case of Fig. 1 verb attachment leads to the instantiation of the PRICE slot of the corresponding SELL action, while the noun attachment leads to the corresponding instantiation of the PRICE slot of PRINTER.

Robustness: Skipping Protocol. The principled lack of exhaustive linguistic and conceptual specifications requires mechanisms for a fail-soft parsing behavior. An example for this is shown in Fig. 2 which illustrates a typical "skipping" scenario. The currently active container addresses a searchHeadFor (or searchModifierFor) message to its textually immediately preceding container actor. If *both* types of messages fail, the immediately preceding container of the active container forwards these messages — in the canonical order — to its immediately preceding container actor. If any of these two message types succeeds after that mediation, a corresponding (discontinuous) dependency structure is built up. Furthermore, the skipped container is moved to the left of the newly built container actor. Note that this behavior results in the reordering of the lexical items analyzed so far in terms of textual precedence such that skipped containers are continuously moved to the left. As an example, consider the phrase *"Zenon sells this printer"* and let us further assume *"totally"* to be a grammatically unknown item which is followed by the occurrence of *"over-priced"* as the active container. Skipping yields a structural analysis for *"Zenon sells this printer over-priced"*, while *"totally"* is simply discarded from further consideration. This mode requires an extension of the basic protocol in that searchHeadFor and searchModifierFor messages are forwarded across non-contiguous parts of the analysis when these messages do not yield a positive result for the requesting actor relative to the *immediately* adjacent container actor.

Backtracking Protocol. The backtracking mode characterizes a state of the structural analysis where none of the aforementioned protocols terminate successfully in *any textually* preceding container, i.e., several repeated skippings have occurred, until a linguistically plausible barrier is encountered. In this case, backtracking takes place and messages are now directed to *historically* previous containers, i.e., to containers holding the parse history. This is realized in terms of a protocol extension by which searchHeadFor (or searchModifierFor) messages, first, are reissued to the *textually* immediately preceding container actor which then forwards these messages to its *historically* previous container actor. This actor contains the head-centered results of the analysis of the left context prior to the structural extension held by the historical succes-

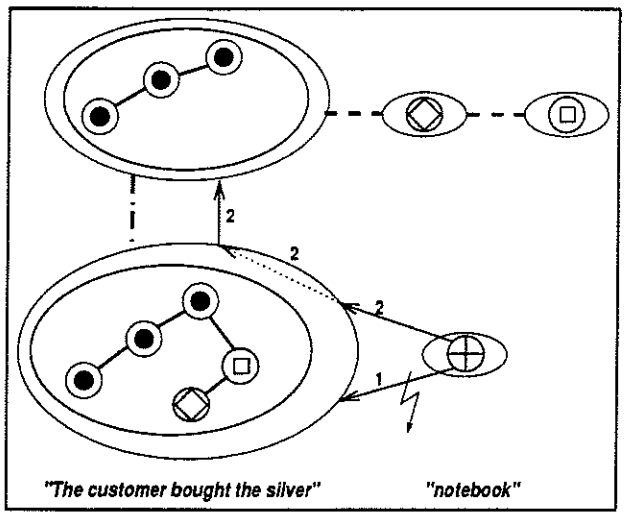


Figure 3: Backtracking Mode

sor.[1] Relation attachments for heads or modifiers are now checked between the historically preceding container and the active container as depicted in Fig. 3.

If the valency constraints are met, a composite phrase actor is formed (cf. Fig. 4) with a necessarily discontinuous analysis. A slightly modified protocol allows the skipped items to try a reanalysis such that the reSearchHeadFor (or reSearchModifierFor) messages they send to the new container actor are directly forwarded to those word actors in the composite phrase actor where the discontinuity occurs. As an example, consider the fragment *"the customer bought the silver"* (with *"silver"* holding the noun reading, cf. Fig. 3). This yields a perfect analysis the result of which, however, cannot be further augmented when the word actor *"notebook"* asks next for a possible attachment.[2] Two intervening steps of reanalysis yield the final structural configuration depicted in Fig. 4.

Preference Encoding. In the model we have sketched so far, messages are directed from the active container actor to its textually preceding container actor and are sent *in parallel* to *all* the phrase actors encapsulated by that container actor. However, the management of ambiguities can be made more efficient by structuring the set of ambiguous analyses in that preference is given to a particular subset of these phrase actors. A predicate is supplied that separates a set of possible attachments into preferred and deferred ones. Then the former ones can be collected into the active container, while the latter ones are stored

[1] Any container which holds the modifying part of the structural analysis of the historical successor is deleted. Hence, the deletion of containers not contained in historically previous head-centered parses, obviously, may result in the parser becoming incomplete in spite of backtracking.

[2] As PARSETALK is an arc-eager parsing system, a possible dependency relation will always be established and favored over an alternative structural description that cannot be immediately attached. Hence, the adjective reading of *"silver"* will not be considered in the initial analysis.

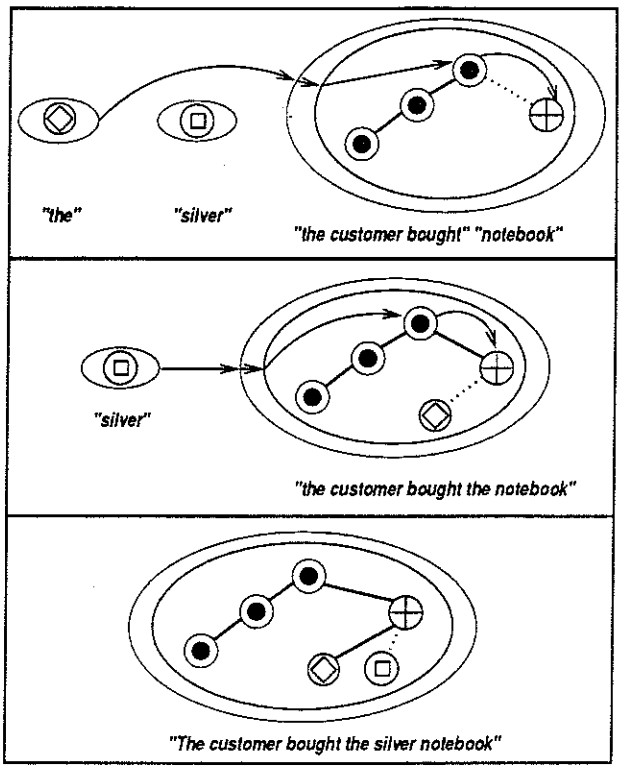


Figure 4: Backtracking Mode (cont.)

in a separate container. Thus, they are not pursued in the current analysis but are retained for backtracking.

Prediction Protocol. The depth-first approach of the parser brings about a decision problem whenever a phrase cannot be integrated into (one of) the left context analyses. The reason for this indeterminacy is twofold: Either, the left context and the current phrase are to be related by a word not yet read from the input and, thus, the analysis should proceed without an attachment. Or, depth-first analysis was misguided and a backtrack should be invoked to revise a former decision with respect to attachment information available by now.

We follow a prediction approach, where certain words predict the *word class* of their head and/or mandatory modifiers. In addition to the basic searchHeadFor and searchModifierFor protocols, predicted word actors (cf. Fig. 5) process a searchPredictionFor protocol by which a word actor is unified with its predicted "placeholder" if its class is equal or more specific. To attach a modifier to a predicted word we exploit the class hierarchy and add any contextually plausible valency of a *subclass* to the valency frame of the predicted word. Under these conditions, the analysis can proceed more informed.

Text Phenomena. A particularly interesting feature of the PARSETALK grammar is its capability to seamlessly integrate the sentence and text level of linguistic analysis. The protocol which accounts for local text coherence analysis

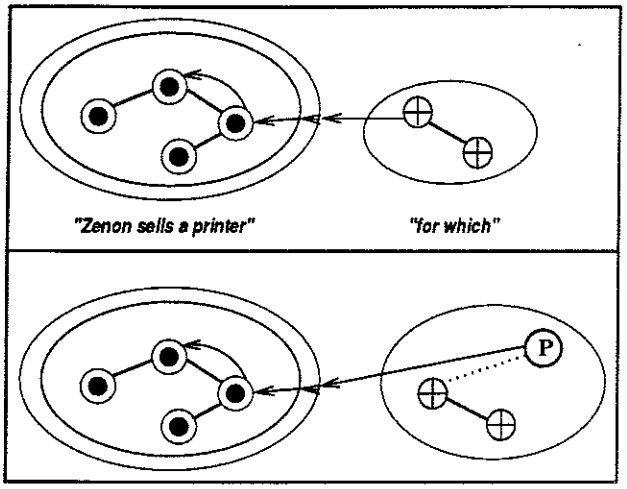


Figure 5: Prediction Mode

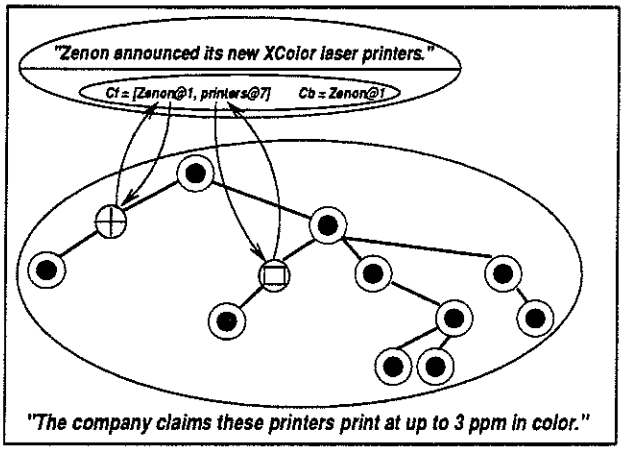


Figure 6: Anaphora Resolution Mode

makes use of a special actor, the *centering actor*, which keeps a backward-looking center (C_b) and a preferentially ordered list of forward-looking centers (C_f) of the previous utterance (we here adapt the well-known centering model by Grosz et al. (1995)). These lists are accessed to establish proper referential links between an anaphoric expression in the current utterance and the valid antecedent in preceding ones. Nominal anaphora (cf. the occurrences of *"the company"* and *"these printers"* in Fig. 6) trigger a special searchNomAntecedent message. When it reaches the C_f list, possible antecedents are accessed in the given preference order. If an antecedent and the anaphor fulfil certain grammatical and conceptual constraints (Strube & Hahn, 1995) an antecedentFound message is issued to the anaphor, and finally, the discourse referent of the antecedent replaces the one in the original anaphoric expression in order to establish local coherence. The effects of these changes at the level of text knowledge structures are depicted in Fig. 7, which contains the terminological representation structures for the sentences in Fig. 6.

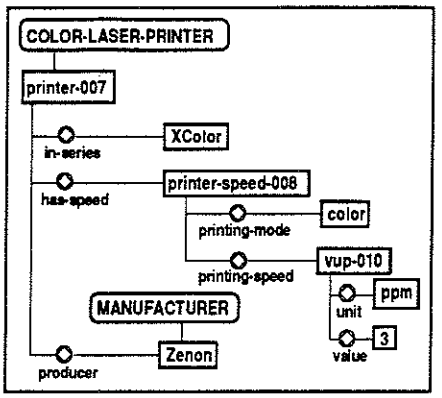


Figure 7: Sample Output of Text Parsing

4 COMPLETENESS *vs.* EFFICIENCY

The requirements of real-world text understanding pose severe complexity problems for parsers in such an operational environment. A depth-first parsing strategy usually will increase the parsing efficiency in the *average case*, i.e., for inputs that are in accordance with the parser's preferences. For the rest of the inputs backtracking is necessary to get the correct analyses. Thus, the *worst case* for a depth-first strategy applies to input which cannot be assigned any analysis at all (i.e., in cases of extra- or ungrammaticality). Such a failure situation leads to an exhaustive backtracking in a computationally complete parsing algorithm such that breadth-first and depth-first complexities will coincide.

With respect to real-world text inputs we have to face such worst-cases in many sentences. That is, by a complete depth-first strategy we merely trade space for time complexity. To maintain the gained efficiency of depth-first operation it is necessary to prevent the parser from exhaustive backtracking. In the PARSETALK parser this is achieved by two means. First, by restricting memoization of attachment candidates for backtracking (e.g., by retaining only the head portion of a newly built phrase, cf. footnote 1). Second, by restricting the accessibility of attachment candidates for backtracking (e.g., by bounding the forwarding of backtracking messages to linguistically plausible barriers such as punctuation actors). In effect, these restrictions render the parser computationally incomplete, since some inputs correctly covered by the grammar specification cannot be analyzed by the parser. Thus, the trade-off between robustness, efficiency, and completeness becomes obvious.

The efficiency gain that results from deciding against completeness is empirically demonstrated by a comparison of the PARSETALK system with a standard active chart parser[3]. Since the chart parser is implemented in

[3]The chart parser (Winograd, 1983) was adapted to parsing a dependency grammar. No packing or structure sharing techniques could be used, since the analyses are continuously interpreted in conceptual terms.

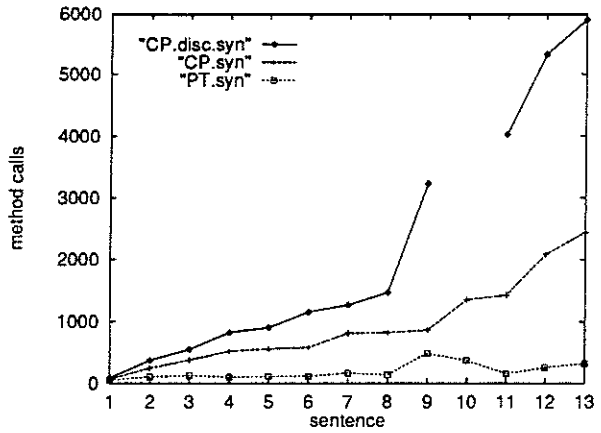


Figure 8: Calls to SYNTAXCHECK

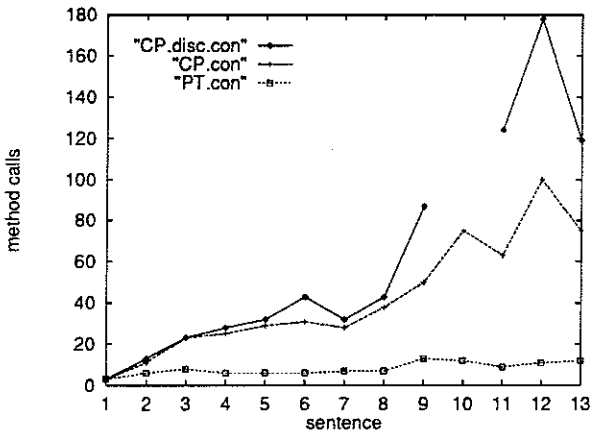


Figure 9: Calls to CONCEPTCHECK

Smalltalk, while the PARSETALK system is implemented in Actalk (Briot, 1989) — a language which simulates the concurrent execution of actors —, a direct comparison of run times is not meaningful (though even at that level of consideration the PARSETALK system outperforms the chart parser). We rather compare the number of method executions given exactly the same dependency grammar. The computationally most expensive methods we consider are SYNTAXCHECK and CONCEPTCHECK (cf. Section 2). Especially the latter consumes large computational resources, since for each interpretation variant a context has to be built in the KB system and conceptual consistency must be checked. The number of calls to these methods for a sample of 13 increasingly complex sentences from the information technology domain test library is given by the plots in Figs. 8 ("CP.syn" and "PT.syn", respectively) and 9 ("CP.con" and "PT.con", respectively). A reduction by a factor of four to five in the (unweighted) average case can be observed applying the PARSETALK strategy.

Furthermore, the PARSETALK parser is able to cope with discontinuities stemming from un- or extragrammatical input anyway. The performance of a revised version of the chart parser which also handles these cases is given as "CP.disc.syn/con" in the above graphics. The missing value at sentence 10 results from the chart parser crashing on some input because of space restrictions of the run time system (the experiments were conducted with a SPARCstation 10 with 64 MB of main memory). The average reduction in comparison with this version of the chart parser is about six to nine.

Though persuasive at first sight, the superior efficiency of the PARSETALK parser is due to the incomplete depth-first nature of our approach and, thus, it is gained at the risk of not finding a correct analysis at all. A comparison of the *effectiveness* of the PARSETALK system based on the conceptual representations extracted from input texts is currently under way.

References

Agha, G. (1990). The Structure and Semantics of Actor Languages. In J. W. de Bakker et al. (Eds.), *Foundations of Object-Oriented Languages*, pp. 1–59. Berlin: Springer.

Briot, J.-P. (1989). Actalk: A Testbed for Classifying and Designing Actor Languages in the Smalltalk-80 Environment. In *Proc. of ECOOP-89*, pp. 109–129.

Charniak, E. (1993). *Statistical Language Learning*. Cambridge, MA: MIT Press.

Chomsky, N. (1965). *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press.

Grosz, B. J., A. K. Joshi & S. Weinstein (1995). Centering: A Framework for Modeling the Local Coherence of Discourse. *Computational Linguistics*, 21(2):203–225.

Hahn, U. (1992). On Text Coherence Parsing. In *Proc. of COLING-92*, Vol. 1, pp. 25–31.

Hahn, U., S. Schacht & N. Bröker (1994). Concurrent, Object-oriented Natural Language Parsing: The PARSETALK Model. *International Journal of Human-Computer Studies*, 41(1–2):179–222.

Huyck, C. R. & S. L. Lytinen (1993). Efficient Heuristic Natural Language Parsing. In *Proc. of AAAI-93*, pp. 386–391.

Shieber, S. M. (1986). *An Introduction to Unification-Based Approaches to Grammar*. Stanford, CA: CSLI.

Strube, M. & U. Hahn (1995). PARSETALK about Sentence- and Text-level Anaphora. In *Proc. of EACL-95*, pp. 237–247.

Thibadeau, R., M. A. Just & P. A. Carpenter (1982). A Model of the Time Course and Content of Reading. *Cognitive Science*, 6:157–203.

Uszkoreit, H. (1991). Strategies for Adding Control Information to Declarative Grammars. In *Proc. ACL-91*, pp. 237–245.

Winograd, T. (1983). *Language as a Cognitive Process.*, Vol. 1, Syntax. Reading, MA: Addison-Wesley.

Woods, W. A. & J. G. Schmolze (1992). The KL-ONE Family. *Computers & Mathematics with Applications*, 23(2–5):133–177.

Spotting Technical Concepts in Natural Language Text

Tomek Strzalkowski and Ronald Brandow
GE Corporate Research and Development
Schenectady, NY 12301
{tomek,brandow}@thuban.crd.ge.com

Abstract

We describe a general method for spotting various kinds of concepts in text using a mixture of shallow processing techniques and standard lexical resources. The concepts of interest may include equipment ("gas turbine assembly"), tools ("adjustable wrench"), products ("canned soup", "Arm & Hammer baking soda"), industries ("aerospace"), companies ("AmCan Feather Co.", "local tv station"'), people, etc. The identification process consists of several smaller sub-processes that can be broadly divided into text preparation and concept extraction phases. In the text preparation phase, a shallow syntactic analysis is performed to identify non-overlapping phrasal groups that are likely to contain the concepts we want to find. For example, equipment names are expected to be noun phrases. Phrasal groups are identified with a PhrasePicker which uses a part of speech tagger to detect possible phrase boundaries. In addition, syntactic context is used to filter out some phrases, e.g.,products are likely to be objects of verbs such as 'produce' or 'sell'. In the extraction phase, a stochastic N-gram analysis is run over all phrases selected in the preparatory phase to detect strong collocations and to break weaker phrases: e.g.,``gas generator installation kit''. Finally, selected N-grams are semantically validated using an on-line universal semantic hierarchy, such as WordNet. After the initial vocabulary is extracted, the text is searched for all instances of the concepts. This in turn may lead to another extraction step with additional syntactic contexts considered.The method has been applied to identify technical concepts such as equipment, systems, tools and consumables in legacy technical manuals.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/66 ©1996 FLAIRS

1 FINDING EQUIPMENT NAMES IN TECHNICAL MANUALS

The task was to identify equipment names and other ``items of interest'' in legacy technical manuals provided in SGML marked electronic form.

Finding equipment names and technical terms in text is not as simple as determining which words or group of words have the highest frequency or largest Mutual Information score. Some low frequency (sometimes a frequency of 1) word groups contain very valuable information: most notably the complete name of equipment or technical terms which are later abbreviated or replaced by an acronym. In technical manuals, the most frequent N-grams usually represent the diminutive form of or an anaphoric reference to the full name.

1.1 Algorithm

Our approach was divided into 4 steps as follows:

1. Tokenization, sentence breaking and tagging
2. Noun phrase extraction
3. N-gram generation
4. Statistical validation
5. Semantic validation

1.2 Tokenization, Sentence Breaking and Tagging

Tokenization is done in a fairly straight forward manner: a stream of characters is divided into words, symbols and groups of single letter abbreviations separated by a period (e.g., U.S.).

The relatively formal nature of the corpus allowed the use of a simple sentence division algorithm which correctly identified 99.6% of sentences in the test corpus. A corpus with a more difficult sentence structure would probably necessitate the use of a

more sophisticated sentence boundary detection algorithm.

Each sentence is tagged utilizing a modified and re-trained version of Brill's tagger (Brill, 1992). Retraining was necessary because the technical manuals used certain common words in different ways than they were used in Wall Street Journal or Brown corpora on which the tagger was originally trained. Modifications were necessary because the original tagger 's lexicon lookup was case-sensitive causing all upper-case words to be initially marked as unknown and tagged during further processing. This caused tagging errors such as mistagging DETECTOR, which only appears as a lower case noun in the lexicon.

1.3 Noun Phrase Extraction

After tagging, the sentences are passed through a noun phrase extractor. As a noun phrase extractor, we used a pattern matcher and a set of rules to identify noun phrases. The patterns are written using part-of-speech information for words as obtained from the tagger, e.g., the rule ***adj noun+*** would capture, among others, the phrase ***high speed turbine.***

1.4 N-gram Analysis

The noun phrases extracted by the phrase picker are run through a N-gram generator, where the length of the N-gram was user selectable; for our experiments we generated N-grams up to six words in length.

For each phrase, the N-gram generator generates all possible N-grams up to the user selectable length subject to the following rules:

1. common words were not allowed in a N-gram[1]
2. all N-grams must end in a noun
3. a N-gram could not start with a number ending in a period

To allow measurement sizes such as **'3/8'** in **'3/8 inch wrench'** to be retained, the N-gram generation algorithm does not preclude numbers from being part of a N-gram. Unfortunately in the test corpus, all the section/title/paragraph headings were numbered. Without explicitly removing the numbers via the third rule, the head number was attached to the N-gram.

The N-gram generator generates multiple N-grams from a single string of tokens. For example, given a string of tokens $X_1X_2X_3$, the N-gram generator would, in the worst case, generate the following N-grams:

- X_1
- X_1X_2
- $X_1X_2X_3$
- X_2
- X_2X_3
- X_3

Next the N-grams are frequency pared by discarding any (N-1)-gram whose frequency is equal to that of the N-gram. That is, for each N-gram, compare its frequency to that of its first (N-1)-gram $X_1..X_{N-1}$ and its second (N-1)-gram $X_2..X_N$. If the frequencies of either (N-1)-gram match that of the N-gram, discard the (N-1)-gram. [2] This paring results in the longest significant N-grams being retained.

After frequency paring, each N-gram is decomposed into two sub-N-grams. The first sub-N-gram is the longest N-gram ending in X_n ($X_k..X_n$) whose frequency is greater than the original N-gram ($X_1...X_n$); the remaining words, $X_1...X_{k-1}$, comprise the second sub-N-gram. This decomposition results in a sub-N-gram, $X_k..X_n$, representing a frequent N-gram with the other sub-N-gram representing the sub-N-gram $X_k..X_n$ modifiers.

1.5 Statistical Validation of N-grams

For each N-gram, a N-gram Mutual Information and t-score value are calculated. Mutual information is used to determine whether a word grouping is accidental or significant. Because most mutual information (MI) calculations assume bi-grams, we modified the common MI formula. Our modified MI formula doesn't favor shorter or longer N-grams and

1. We used a modified version of the standard stoplist used in text retrieval systems, e.g., Frakes & Baeza-Yates, 1992.

2. Identical frequencies imply that a (N-1)-gram never appears outside of the N-gram and thus could be discarded because it does not add any additional information.

for N = 2, collapses into the familiar bi-gram Mutual Information formula. The modified N-gram mutual information formula is given in equation 1.

T-test are usually used to measure the dissimilarity of two bi-grams.; however, to accurately calculate t-Scores for larger N-grams we modified the traditional bi-gram t-test formula. Furthermore, we used the t-score not as a test for dissimilarity but as an objective measure of the strength of the word grouping. The modified t-test formula is given in equation 2.

$$\frac{\frac{F(X_1 \ldots X_n)}{N_{n-grams}}}{\frac{F(X_1 \ldots X_{k-1})}{N_{k-1-grams}} \times \frac{F(X_k \ldots X_n)}{N_{n-k+1-grams}}} \qquad Eq1$$

$$\frac{\frac{F(X_1 \ldots X_n)}{N_{n-grams}} - \frac{F(X_1 \ldots X_{k-1})}{N_{k-1-grams}} \times \frac{F(X_k \ldots X_n)}{N_{n-k+1-grams}}}{\sqrt{\frac{F(X_1 \ldots X_n)}{N_{n-grams}}}} \qquad Eq2$$

where:
$F(X_a \ldots X_b)$ is the frequency of $X_a \ldots X_b$, and
$N_{X\text{-}grams}$ is the number of N-grams of size X

1.6 Semantic Validation of Candidate N-grams

To further reduce the number of N-grams, the surviving N-grams were validated using an on-line universal semantic hierarchy, in our case WordNet. The last word of each N-gram was inspected to ascertain if it fell within a conceptual category signifying items of interest for the corpus. All N-grams which passed this validity test were retained with all failing N-grams being rejected.

2 EXAMPLE

To test our method we used an.8 Mbyte SGML marked file as input and were supplied with a list of 40 customer selected representative "items of interest" to be identified; this list is shown in Table 1. From the 40 representative items, we identified nine conceptual categories and 7 "keyword" categories. "Keyword" categories were necessary to include desired head nouns which did not have a sufficiently narrow WordNet concept hierarchy.

The file contained approximately 130,000 tokens and 9600 sentences. From the 9600 sentences, 27220 noun phrases were extracted by the noun phrase picker. These phrases were processed by the N-gram generator which generated 8554 unique N-grams occurring a total of 70227 times. Thirty-seven (92.5%) of the forty representative "items of interests" were generated.[3]

Table 2 shows the breakdown of the N-grams by N-gram length.

base/enclosure assembly	bulkhead mounted electronic enclosure	cable interface module	center body
clutch/brake assembly	control console	controllable reversible pitch propeller system	drive system
electronic control and monitoring system	electronic support module	engine control module	engine room
exhaust collector	free standing electronic enclosure	gas generator	gas turbine
gas turbine assembly	gas turbine electronic power control system	gas turbine equipment	gas turbine main fuel control
gas turbine module	inlet duct	interim integrated electronic control	lube storage and conditioning assembly
lube storage tank	main propulsion system	overspeed switch electronics	power lever angle actuator electronics
power train system	power turbine	propulsion gas turbine	propulsion gas turbine module
propulsion gas turbine unit	propulsion system	propulsor gearbox	reduction gear
removable panel	shafting and bearing	start/stop sequencer electronics	torque computer electronics

Table 1: Representative "Items of Interest"

N-gram Length	# Unique N-grams	N-gram Frequency	Average N-gram Frequency
1	1799	46014	25.5
2	3443	16278	4.7
3	1920	5376	2.8
4	828	1664	2.0
5	341	575	1.7
6	223	320	1.4

Table 2: Generated N-Gram Distribution

3. Two of the "items of interest" were not found due to embedded punctation and acronyms; the remaining "item of interest" was not generated due to a part of speech tagging error.

Because the majority (80%) of the N-grams generated occurred less than 4 times, a simple frequency threshold would not be sufficient to identify the items of interest therefore T-scores were calculated. Generated N-grams whose $\log_2$(t-Score) was less than1.65 were rejected.[4] This filter rejected all but 2195 of the generated N-grams. Thirty (75%) of the forty "items of interest" survived the t-test filter[5].

Lastly, semantic filtering using WordNet was performed. After semantic paring 855 N-grams remained and 30 (75%) of the 40 "items of interest" remained.

Of the remaining 825 N-grams, 643 (77.94%) were valid technical terms. Therefore for this test an overall precision of 78.71% was attained. However, an overall recall value was not calculated because we did not examine the test document to determine the number of technical terms which were missed.

3 RELATED RESEARCH

Justeson and Katz (Justeson, 1993) developed an algorithm to identify terms in technical articles. They used lexical part of speech filtering on a word basis with a grammar to identify possible technical terms.

Work in this area has also been done by Dagan and Church (Dagan, 1994). Like Justeson et al, Dagan and Church also utilized regular expression matching and term frequency to determine candidate terms; however, instead of using lexical part of speech filtering they utilized a part of speech tagger to obtain each word's part of speech.

Unlike Justeson et al we used lexical and contextual part of speech tagging on a sentence basis and a phrase picking algorithm to identify our technical terms. Furthermore we utilized Mutual Information and t-tests to determine our technical terms while Justeson and Katz primarily utilized frequency to determine whether to accept or reject a noun phrase and Dagan and Church who used frequency to order the display of candidate terms.

4. A $\log_2$(T-test score) >= 1.65 implies 95% confidence that the N-gram grouping is real. See Church(1993) for further information.

5. Interestingly, all but one of the failing "items of interest" had a Mutual Information score greater than 6.

Additionally our N-gram generation algorithm identifies noun phrases within the noun phrase identified by the phrase picking algorithm. This feature allows us to identify strong technical terms even if those terms do not appear individually within the text.

Lastly, our system does not rely upon a corpus of documents but utilizes the information present in a document to determine the items of interest.

4 CONCLUSIONS

One of the problems facing automatic understanding of textual information is automatic identification of items of interest. Summarization, Information retrieval, natural language generation, and extraction systems would benefit by the ability to automatically identify terms in text.

This method illustrates the feasibility of utilizing shallow methods in connection with semantic filtering to obtain reasonable N-grams. We attained a 75% recall on the customer selected "items of interest" subset with an overall precision of 78.71%.

We have implemented this method as part of a larger system that converts technical manuals for various types of equipment from plain text format to hierarchical object-oriented representation. The method is being continually improved as we process more text and gain more empirical data.

References

Brill, Eric. 1992. ``A simple rule-based part of speech tagger" In *Proceedings of 3rd Conference on Applied Natural Language Processing*, Trento, Italy.

Church, Kenneth Ward and Patrick Hanks. 1990. ``Word association norms, mutual information, and lexicography" *Computational Linguistics*, 16(1), MIT Press, pp 22-29.

Church, Gale, Hanks, Hindle "Using Statistics in Lexical Analysis" in *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon,* U. Zernik, ed., 199, pp 115-164

Dagan, I. and Church, K. "Termight: Identifying and Translating Technical Terminology", in *4th Conference on Applied Natural Language Processing,* October 13-15, 1994, Stuggart,

Germany, pp 34-40

Frakes, William, B. and Ricardo Baeza-Yates. (eds.). 1992 *Information Retrieval*, Prentice-Hall, Englewood Cliffs

Justeson, J. and Katz, S. "Technical Terminology: Some Linguistic Properties and an Algorithm for Identification in Text", *IBM Research Report # RC 18906,* 1993

Proceedings of the Fifth Message Understanding Conference (MUC-5)

Tipster Text Program (Phase 1)

On the Complexities of NLP systems

Wlodek Zadrozny
IBM Research, T. J. Watson Research Center
Yorktown Heights, NY 10598
wlodz@watson.ibm.com

Abstract

We give an account of the semantic differences between NLP systems, e.g. those used for information extraction and the older ones such as BORIS or ELIZA. We start by defining the concept of semantic complexity, and then compare NLP systems wrt. to different complexity measures. We argue that BORIS could not control the complexity of interaction, and that is why it did not scale up: its "iterated why-complexity" was orders of magnitude too large. In contrast, by limiting the extracted information to simple semantic types, current systems limit their semantic complexity.

1 Introduction

The problem

We give an account of the semantic differences between NLP systems, e.g. those used for information extraction and the older ones such as BORIS ([4]) that were capable of "in depth" conversation about one topic. This comparison is not based on a list of features, but on a method, introduced in [18], that potentially has a much larger domain of application. Namely, we define the concept of semantic complexity wrt. to a set of questions, and then compare NLP systems wrt. to those complexity measures. We argue that BORIS could not control the complexity of interaction, and that is why it did not scale up. The problem was inherent: its "iterated why-complexity" (defined below) was orders of magnitude larger than what is possible to handle even today. In contrast, by limiting the types of extracted information to simple semantic types, such as numbers, yes-no answers, and type-value pairs, current information extraction systems and dialog systems reduce their semantic complexity and make it manageable.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/71 ©1996 FLAIRS

Two dialogs

Dialog 1:

```
-- I want to set up an appointment
   with Martin on the 14th of March
   in the IBM cafeteria.
-- At what time?
-- At 5.
```

Dialog 2:

```
-- Why did Sarah lose her divorce case?
-- She cheated on Paul.
```

The first dialog is a task dialog from MINCAL [19], a calendar management program. The analysis described in this paper is based on our experience with MINCAL and related programs dealing with banking transactions, fast food, insurance and email. (Furthermore, there is rich literature on that topic e.g. [1], [16], [19]). The second kind of dialog comes from BORIS. (And there were a whole series of such reports in the 70s and early 80s by Schank and his students, cf. [7], [13]).

Of course one can argue (e.g. [17]) that none of the programs truly understands any English. But even if they fake understanding, the question remains in what sense is the domain of marital relations more complex than the domain of appointment scheduling (if it really is). (Notice that the syntax of the first dialog is more complex).

First comparison

If we compare BORIS ([4], [7]) with MINCAL we notice some clear parallels. First, they have an almost identical vocabulary size of about 350 words. Secondly, they have a similar number of background knowledge facts. Namely, BORIS uses around 50

major knowledge structures such as Scripts, TAUs, MOPs, Settings, Relationships etc.; on average, the size of each such structure would not exceed 10 Prolog clauses (and no more than 4 predicates with 2-3 variables each per clause) if it were implemented in Prolog. If we apply a similar metrics to MINCAL, we get about 200 facts expressing background knowledge about time, events and the calendar, plus about 100 grammatical constructions, many of them dealing with temporal expressions, others with agents, actions etc. Clearly then the two systems are of about the same size. Finally, the main algorithms do not differ much in their complexities (as measured by size and what they do). So the question remains: why is the domain of scheduling meetings "easier" than the domain of discussing divorce experiences? How could we measure the open-ended character of the latter?

2 Meaning automata

In order to compare NLP systems, esp. if we want to compare systems performing different tasks we need a set of tools. One of the tools for measuring complexity widely used in theoretical computer science is *Kolmogorov complexity*. However, the concept of K.-complexity must be modified to work for our task. This will be done in this and the next section.

Kolmogorov complexity of a string x is defined as the size of the shortest string y from which a certain universal Turing machine produces x. Intuitively, y measures the amount of information necessary to describe x, i.e. the information content of x. (cf. [8] for details and a very good survey of Kolmogorov complexity and related concepts). For our purposes any of the related definitions of complexity will work. For example, the Minimum Description Length of Rissanen ([8] and [10]), or the size of a grammar (as in [12]), or the number of states of an automaton.

Obviously, K.-complexity does not by itself tell us anything about natural language and semantics, so we will modify it, so that we could define and measure the semantic complexity of natural language. To do that, we assume that the meaning of a sentence is encoded in the questions it can answer (examples below), a view also held by Bloomfield [2]. So let Q be a set of questions. We will define *Q-complexity* of a set of sentences S as the size of the simplest model scheme $M = M_S$, such that for any sentence s in S its semantics is given by *M(s)*, and *M(s)* correctly answers all questions about s contained in Q.

The words "model scheme" can stand for either "Turing machine", or "Prolog program", or "description", or a related notion. In this paper we think of M as a Turing machine that computes the semantics of the sentences in S, and measure its size by the product *(Number of States)* × *(Number of Symbols)*. Thus defined *State* × *Symbol*-complexity is a good approximation of K-complexity ([8]).

To define Q-complexity, we first define the concept of *meaning automaton* (M-automaton): Let, as above, Q be a set of questions, and S set of sentences (e.g. accepted by a grammar). Formally, we treat each question as a (partial) function from sentences to a set of answers A:

$$q : S \rightarrow A$$

Intuitively, each question examines a sentence for a piece of relevant information. Under this assumption the semantics of a sentence (i.e. a formal string) is not given by its truth conditions or denotation but by a set of answers:

$$||s|| = \{< q, q(s) >: q \in Q\}$$

Now, given a set of sentences S and a set of questions Q, their *meaning automaton* is a function

$$M : S \times Q \rightarrow A$$

which satisfies the constraint

$$M(s, q) = q(s)$$

i.e. a function which gives a correct answer to every question. We call it a meaning automaton because for any sentence s

$$||s|| = \{< q, M(s, q) >: q \in Q\}$$

As before, the *Q-complexity* of the set S is the size of the smallest such M.

Note that the idea of a meaning automaton as a question answer map allows us to bypass all subtle semantics questions without doing violence to them.

Note also that in practice we will deal not with *the* simplest model scheme, but with the simplest we are able to construct. Furthermore, to take care of the possible non-computability of the function computing Q-complexity of a set of sentences, we can put some restriction on the Turing machine, e.g. requiring it to be finite state or a stack automaton. Finally, we will use the size of the Turing machine description — *T-rule-complexity* — as another, related, measure (see Section 3.2).

3 Semantic complexity classes

3.1 Choosing Q for Q-complexity

We can measure the semantic complexity of a set of sentences by the size of the smallest model that

answers all relevant questions about those sentences (in practice, of the simplest one we are able to construct). But how are we going to decide what relevant questions can be asked. Before we attempt to solve this problem, we can examine the types of questions. A simple classification of questions given by [11] (pp.191-2) is based on the type of answer they expect: (1) those that expect affirmation or rejection — yes-no questions; (2) those that expect a reply supplying an item of information — Wh questions; and (3) those that expect as the reply one of two or more options presented in the question — alternative questions.

3.2 yes/no, "what-is" complexities

Let us consider a simple case of two sets of 24 sentences based on similar constructions. One specifying the content of a calendar: *The meeting is at X*, with X ranging over hours. The other one describing whereabout of a person: *John is at Y*, with *at Y* chosen from (cf. [14]): *at breakfast, at lunch, at dinner, at school, at work, at rest, at ease, at liberty, at peace,* etc.

We can measure the *yes-no*-complexity of both T and S. Let M_T be the mapping from $T \times Q_T \rightarrow \{yes, no\}$, where

$$Q_T = \{q_X : q_X = \text{"Is the meeting at X?"}\}$$

and $M_T(s_Y, q_X) = yes$, if $X = Y$, and *no* otherwise. (s_Y = *"The meeting is at Y"*, and we identify the time expressions with numbers for the sake of simplicity). Clearly, under this mapping all the questions can be correctly answered (remember that question q_{13} returns *yes* for s = *"The meeting is at 1 pm"*, and *no* otherwise).

M_S is a similar mapping: we choose arbitrary 24 tokens, and map the sentences of S into them in a 1-1 fashion. As before, for each s in S, $M_S(s, q)$ is well defined, and each question of the type *Is John at breakfast/.../at age?* can be truthfully answered.

If we measure the semantic complexity by the number of pairs in the $M_{S/T}$ functions, the yes-no complexities of both sets are the same and equal to 24^2. If we measure it by the product *States* × *Symbols* (of their respective Turing machines), because the two problems are isomorphic, their yes-no complexities will again be identical. For example, we can build a two state, 4-tape Turing machine. It would scan symbols on two input tapes, and print *no* on the output tape if the two input symbols are not equal. The third input tape would contain five 1's and be used as a counter (the binary string *twxyz* represents the number $1t + 2w + 4x + 8y + 8z + 1$). The machine moves always to the right, scanning the symbols. If it terminates with *accept* and the empty output tape, it means *yes*; if it terminates with *accept* and the *no* on the output tape, it means *no*. this machine has the *State* × *Symbol* complexity of $6 = 2 \times 3$. Since it can be described as a 6×5 table, we can assign it the T-rule complexity of 30.

state	*input1*	*input2*	*counter*	*output*	*next*
1	1	1	1	b	1
1	0	0	1	b	1
1	1	0	1	no	1
1	0	1	1	no	1
1	b	b	b	b	acc
(acc)					

We arrive at a surprising conclusion that a set of idiomatic expressions with complicated meanings and a trivial construction about time can have the same semantic complexity. (From the perspective of answering yes/no questions).

"what is?"-complexity

Let U be a finite set of tokens. Consider the following semantic machine M_U: For any token u in U, if the input is "what is u" the output is a definition def_u of u, and the next state is *acc* (accept). The output may be one or more tokens, but is written in one move; the input consists only of one token, namely u, and the question is implicit. Then, the size of M_U is the measure of "what is"-complexity of U. M_U can be described as a one state, two tape, Turing machine consisting of the following rules

$(1, u, def_u, acc)$ for $u \in U$

Both the T-rule– and the *State*×*Symbol*– complexity of M_U are a simple function of the size of U.

Thus, for T the size of U is 12+4=16, as we can ask about every number, the meeting, the word "is", and the tokens "am" and "pm". (We assume "the meeting" to be a single word). For S we get 24+2=26, as we can ask about every X in "at X", about "is", and about "John".

Iterated "what is?"-complexity

What would happen if one would like to play the game of asking "what is" questions with a machine. How complex such a machine would have to be? Obviously, in reality "what is" questions produce multi-token answers. Furthermore, the results of [5] and [3] (and the common experience of NL researchers) suggest a roughly 10:1 ratio of the number of words to the number of facts necessary to understand sentences with them. Based on this estimate, we can assume that answers to "what is" questions are 10

tokens long.

If we now examine the two sets of sentences, we can see that for the set T we need about 20 facts for two rounds of questions (because we can use the same answer "number" for tokens 1..12). However for S we would need about 250 facts for two rounds of questions (since the words of the PPs are in different semantic classes). These new numbers are closer to our intuitive understanding of the semantic complexity of the two sets. (We leave the computation of the *State* × *Symbol*- complexity to the reader).

3.3 Q-Complexity of ELIZA

We can compute the semantic complexity of ELIZA [15] as Q-complexity. Namely, we can identify Q with the set of key-words for which ELIZA has rules for transforming input sentences (including a rule for what to do if an input sentence contains no key-word). ELIZA had not more than 2 key list structures for each of the about 50 keywords, and its control mechanism had 18 states.

If we assume that states and keywords corresponds to rules of a Turing machine, we can estimate ELIZA's complexity at 118 rules. Since the machine could be represented as a 4-column table, its T-rule complexity would be roughly $4 \times 118 = 472$. (Its *State* × *Symbol* complexity can be estimated in a similar way).

However, ELIZA maintained the distinction between program and data. Thus we can distinguish between the relative complexities of ELIZA database for the domain X, given ELIZA program (here about 4×100 for T-rules), and the complexity of the program itself (about 4×18). The moral is that all discussions of complexity must assume some level of abstraction. (Notice the parallels with the Lindenbaum algebras in logic).

3.4 Complexity of extraction

Similar measures can be applied to information extraction (e.g. [9]). Huffman, in a recent paper [6], uses about 60 extraction patterns generated from 150 examples to create an NLP system attuned to corporate management changes. Thus the complexity of his system measured by the number of patterns is 60.

This is the most relevant complexity measure for the system. But at a different level of abstraction, its complexity would be larger, since it would have to include the complexity(-ies) of the grammar that is needed to break the text into noun and verb groups; or even the complexity of determining the sentence boundaries.

In addition to talking about Q-complexities at different abstraction levels, we can also say that the number 60 reflects the complexity of the task at the level of 90% precision (as achieved by the program). This is natural because of the possible translation between Minimum Description Length and K-complexity ([8]).

3.5 Semantical complexity of BORIS

We now can explain the difference in the apparent semantic complexities of BORIS and MINCAL. First, as we noted in Section 1, their vocabulary sizes and the sizes of their respective knowledge bases are almost identical. Thus, their "what-is"-complexities are roughly the same.

But our theory can give an explanation of why the sentence *The meeting is at 5* seems simpler than *Sarah cheated on Paul.* Namely, for the last sentence we assume not only the ability to derive and discuss the immediate consequences of that fact such as "broken obligation " or "is Paul aware of it?", but also such related topics as "Sarah's emotional life" , "diseases", "antibiotics", and "death of grandmother". In other words, the real complexity of discussing a narrative is at least the complexity of "iterated-what-is" combined with "iterated-why" (and might as well include alternative questions). By the arguments of the preceding section this would require really extensive background knowledge, and the Q-complexity (measured by the number of facts) would range between 10^5 and 10^7; assuming each round of dialog requires an iteration of "why" and "what is", with at least 5 rounds of dialog, and an average of 10 facts per token in the database of background knowledge. In contrast, the Q-complexity of MINCAL is less than 10^4; there are no restrictions on the number of exchanges; but there is an implicit assumption that all dialogs are restricted to the tasks of scheduling, canceling and moving meetings, and there is no expectation of discussing the content of the meetings with the machine (the meetings could have topics though). Here, the set Q consists of questions about parameters of calendar events: *action_name?*, *event_time?*, ... (In general, in the context of a set of commands, we identify Q with the set of queries about the required and optional parameters of actions).

4 Conclusions

We have introduced a theoretical framework in which different NLP problems and programs can be compared. It is based on the idea that the semantics of a sentence is given by a *meaning automaton* — a device

that extracts answers to a set of questions Q. Thus the same sentence can be assigned different meanings by different automata. Therefore sentences do not have one, task independent, meanings. Since all NLU tasks can be viewed as the problems of extracting data from sentences, the idea can be universally applied. In particular it fits well both with information extraction programs and with dialog systems. (And a similar idea would also apply to NL generation).

The complexity of NL system is correlated with the types of constructions the system can process, e.g. "why" vs. "what is" questions. This processing requires background knowledge. Dictionaries and manuals can be used as knowledge bases for "what" questions. There are currently no KB with causal knowledge, plans etc. This makes the management of "why"-complexity a more difficult task. (However, a system can recognize "why" questions, but process them in a 'yes-no' mode, e.g. by saying "I cannot answer any 'why' questions at this time".)

The difficulty of a problem can be measured by describing its class of questions Q, and estimating the size of an automaton that computes answers to Q for a given class of sentences. The complexity of the smallest such automaton is the *Q-complexity* of the problem. NLP programs — the existing ones and those being designed — can be analyzed by comparing their complexity with the Q-complexity of the task they are supposed to accomplish. As we have noticed in our analysis, these automata can be described at different levels of abstraction.

References

[1] E. Bilange and J-Y. Magadur. A robust approach for handling oral dialogues. *Proc. Coling'92*, pages 799–805, 1992.

[2] L. Bloomfield. A set of postulates for the science of language. *Language*, vol.I, pages 799–805, 1926. Reprinted in: Hayden, D.E. et al. (Eds.). *Classics in Linguistics*. Philosophical Library Inc., NY, 1967.

[3] E.J. Crothers. *Paragraph Structure Inference*. Ablex Publishing, Norwood, New Jersey, 1979.

[4] M.G. Dyer. *In-Depth Understanding*. MIT Press, Cambridge, MA, 1983.

[5] A.C. Graesser. *Prose Comprehension Beyond the Word*. Springer, New York, NY, 1981.

[6] S. B. Huffman. Learning information extraction patterns from examples. *Proc. IJCAI-95 workshop on New Approaches to Learning for Natural Language Processing*, 1995.

[7] W. Lehnert, M.G.Dyer, P.N.Johnson, C.J.Yang, and S. Harley. Boris – an experiment in in-depth understanding of narratives. *Artificial Intelligence*, 20(1):15–62, 1983.

[8] M.Li. and P.Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, New York, 1993.

[9] *Proc. of the 5th Message Understanding Conference*. Morgan Kaufman, 1993.

[10] J. Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11:416–431, 1982.

[11] R.Quirk and S.Greenbaum. *A Concise Grammar of Contemporary English*. Harcourt Brace Jovanovich, Inc., New York, NY, 1973.

[12] W. J. Savitch. Why it might pay to assume that languages are infinite. *Annals of Mathematics and Artificial Intelligence*, 8(1,2):17–26, 1993.

[13] R. C. Schank, editor. *Conceptual Information Processing*. Americal Elsevier, New York, NY, 1975.

[14] J.A. Simpson and E.S.C. Weiner, editors. *The Oxford English Dictionary*. Clarendon Press, Oxford, England, 1989.

[15] J. Weizenbaum. Eliza. *Communications of the ACM*, 9(1):36–45, 1966.

[16] R. Wilensky, D.N. Chin, M. Luria, J. Martin, J. Mayfield, and D. Wu. The berkeley unix consultant project. *Computational Linguistics*, 14(4):35–84, 1988.

[17] T. Winograd and F. Flores. *Understanding Computers and Cognition*. Ablex, Norwood, NJ, 1986.

[18] W. Zadrozny. Measuring semantic complexity. In *Proc. BISFAI'95, The Fourth Bar-Ilan Symposium on Foundations of Artificial Intelligence*, Ramat-Gan and Jerusalem, Israel, 1995. Bar-Ilan University Press.

[19] W. Zadrozny, M. Szummer, S. Jarecki, D. E. Johnson, and L. Morgenstern. NL understanding with a grammar of constructions. *Proc. Coling'94*, 1994.

USING HIERARCHIES OF MACRO CELLS TO LINEARIZE SEARCH COSTS FOR REAL-TIME ROUTE PLANNING

Sean Henry
Golden Monkey Software
4 Park Avenue, Ste. 21-G
New York, NY 10016
(212) 532-1791
seann@pipeline.com

Marty Hall
Milton S. Eisenhower Research Center
The Johns Hopkins University Applied Physics Lab
Johns Hopkins Rd., Laurel, MD 20723
(410) 792-6221
marty_hall@jhuapl.edu

Abstract

In recent years the field of Intelligent Vehicle/ Highway Systems has experienced a tremendous surge of interest[9]. The development of automatic routing and dispatch programs is a vital component of these IVHS systems[11]. Unfortunately, real-time route planning on graphs representing large, complex networks of city streets can be extremely difficult. While heuristic path planning is in general significantly more efficient than uninformed planning, the improvement is only a constant factor for city street routing problems[10]. For long routes the problem becomes intractable in practice, requiring that enhancements to standard search algorithms be used.

This paper traces the development of a commercial, real-time route planning system, *MasterMap: Taipei*[12], showing the methods used to evaluate competing cost reduction schemes.

The method selected, a hierarchy of macro cells, trades the cost of a systematic preexploration of the search space in exchange for linearizing both search time and memory costs on future searches on the city street map. Unlike all other methods investigated, macro cells deliver these savings without sacrificing either completeness (the guarantee that a solution will be found whenever one exists) or admissibility (the guarantee that the solution found is optimal)[10].

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/76 ©1996 FLAIRS

1 THE PROBLEM

A beta release of a PC-based mapping package MasterMap:Taipei contained a module for planning the quickest (shortest driving time) route between any two points on the Taipei street map. The method used was heuristic path-planning using the A* algorithm[7]. The cost function for the A* search took into account both street type and the cost of turning from one street to the next. The test data included forty square kilometers of a streets in Taipei, Taiwan. This search space included over 12,700 states as represented by endpoints of directed street segments (see Figure 1).

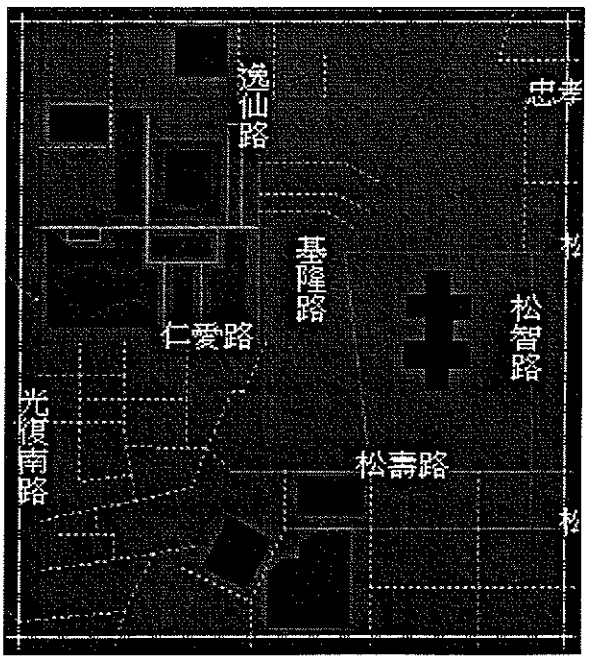


Figure 1: One square kilometer of Taipei, Taiwan test data.

While the A* search used was able to, in principle, return an optimal route between any two map locations, both the memory requirements of the search and the time required to complete the search exceeded the capacity of the targeted low-end laptop machine. An A* search on a tightly interconnected network of randomly scattered location states with a node density of P_0 has a worst-case complexity of $\Theta(P_0 d^2)$, where d is the distance between the start and goal states[10]. This search cost placed an unreasonably low bound on the length of routes that could be calculated in real-time by the application (about 5 Km).

The standard method for reducing the cost of A* path planning is to sacrifice solution optimality by using a inadmissible (pessimistic) heuristic function. In practice--on city street route planning--sizable reductions in search costs were achieved only through unacceptably suboptimal solution paths. In light of this, various other methods for reducing search costs were investigated.

2 EVALUATION CRITERIA

The following criteria were used in evaluating the strengths and weakness of various cost reduction methods:

Cost Savings: Improvement when compared to the $\Theta(P_0 d^2)$ performance of unaugmented path planning algorithms.

Solution Quality: Ability to guarantee finding a minimal cost solution, or the bounds the approach can place on the ratio of the cost of the found plan to the cost of the optimal plan.

Memory and Offline Processing Time Required.

Incrementality: Ability to update the method to reflect real-time changes in road conditions.

3 ALTERNATIVE APPROACHES

3.1 All Pairs

As our first consideration was cost savings, we initially considered an all pairs algorithm such as the Floyd-Worshall algorithm[1], where extensive off-line processing would be used to create a lookup table containing optimal paths between every point on the map to every other point on the map. In view of the first two criteria, this method seemed promising: computation time for a given route would be just a constant factor, regardless of the length between the route's start and goal nodes, and the method promised to deliver an optimal route wherever one existed. However in examining our third criteria--the resources required--insurmountable obstacles became apparent.

Floyd-Worshall runs in $O(n^3)$ time and requires $\Theta(n^2)$ storage space (where n is the number of nodes in the graph). As the test base of forty square kilometers of Taipei's city streets contained nearly thirteen thousand nodes, and as the application would be required to handle much larger maps, an all pairs method was judged to be impractical.

3.2 Knowledge-based and Case-based Systems

Consideration was also given to the approach used by the National University of Singapore's R-Router[8]. While R-Router is based upon an uninformed search method, Dijkstra's Algorithm, the search is augmented by a knowledge base and a case based system.

The standard use of Dijkstra's algorithm requires searching every node on the problem graph[2], so it is impractical for large and complex networks. R-Router's response is to add a knowledge component to limit the search to a subgraph. This consists of a set of 'rules of thumb' for pruning nodes from the search. While the knowledge component passes one of our criteria easily (the off-line calculation costs are minimal), the complexity of the resulting search problem (Dijkstra with a knowledge base) is no better than that of a heuristic search by itself. Both are $O(P_0 d^2)$. Worse still, the method cannot make any claims about the quality of the solution delivered, or even guarantee completeness. Pruning the search space is a good way of speeding up a search, but arbitrary prunes may lead to catastrophically poor solutions or no solution at all.

R-Router's case-based component consists of two parts, a data base of pre-computed routes (essentially a sparse version of the lookup table used in by all-pairs) and a problem solver. The table is not created by systematic off-line pre-exploration as in the all-pairs method, instead it is filled on-line as the router is being used. Whenever a path is computed using Dijkstra and the knowledge base, it is stored in the lookup

table (until the storage space allocated for the sparse table is exhausted). Whenever a origin-destination path request is made, the table is consulted to determine if the same request had been made in the past. If it was made, the table supplies the precalculated path in constant time. For most route requests, this in not the case. At this point rather than to initiate an expensive search, the problem solver is called into action to use a combination of *similar* routes stored in the lookup table to present a quick solution. For instance if a path from point A to point C is requested, and is not found in the database, yet two paths A-to-B and B-to-C were in the database, then the A-to-B and B-to-C routes are linked to provide a solution. Unfortunately this can provide extremely poor results at times: the shortest route from Austin to Chicago would not be an Austin to Boston, Boston to Chicago trek. Breaking a problem into subproblems can speed up the search, but doing so arbitrarily can lead to extremely poor solutions.

Using our evaluation criteria on R-Router's case based system highlights some shortcomings. Resource savings: while in very limited cases the savings are great, in many cases, due to the sparseness of the database, no savings are seen. Solution quality: inconsistent, in many cases the solution can differ greatly from the optimal path. The question of how well the cost reduction scheme incorporates map modifications presents additional problems for R-Router, as it is unclear how a given map modification could be reflected in an existing database.

3.3 Abstraction

Another cost reduction scheme investigated was problem space abstraction. The main theme behind abstraction is to increase the efficiency of a problem solver by initially ignoring low level details and concentrating on higher level subproblems. In the case of a graph search, preprocessing is done on the graph to group nearby nodes into supernodes. The supernode graph is an "abstraction" of the original graph, and, as it contains far less nodes than the original graph, a search performed upon the abstraction is much less expensive than a search done on the original graph. The path returned by a search on the abstraction is really an ordered set of subgoals. The fleshing out of the subgoals, *refinement*, completes the process, resulting in a solution.[4,5]

Abstraction has a very attractive property: hierarchies of abstractions can be created. Abstraction reduces a graph's complexity by clustering nodes into node groups. The process can be continued (abstracting the abstraction), and in fact it could be repeated until the graph consists of a single node and a search is trivial. While each layer of abstraction requires an additional level of refinement that must be performed before the search can arrive at a solution, the application of a hierarchy of abstractions can reduce the complexity of the problem from $O(P_0d^2)$ to $O(P_0d)$.

In addition to the complexity reduction, abstraction has a reasonable overhead cost and a hierarchy of abstractions can be selectively updated to reflect changes in the city street graph. But as was the case with R-Router, there is no guarantee that breaking the problem into subgoals and then solving the subproblems will result in an optimal solution. Given that three out of our four evaluation criteria are met readily by abstraction, is the requirement of an optimal solution unreasonable? Generally speaking, the price for a reduction in the cost of a solution is a reduction in solution quality[6]. Need this be the case with routing on city streets?

4 THE IMPORTANCE OF ADMISSIBILITY

Our goal is to create an efficient system for routing vehicles on city streets. Technology is providing more and more accurate vehicle positioning technology (GPS)[9], which in turn is enabling cartographers to create more and more accurate digital maps. Vehicles can now be located to within meters[3,11]. Maps likewise can be accurate down to the meter. Traffic monitoring systems can update digital maps to indicate exactly how fast traffic is moving throughout a city.

Cities are starting to develop complex systems for dispatching huge fleets of emergency vehicles. Fire trucks, ambulances and police cars are being automatically directed to emergency sites. Directed via paths that are *nearly* optimal? If the best model for traffic flow is applied to a near-perfect graphical representation of the city for vehicles whose location are known with pinpoint accuracy, must graph search to be the weak link in the chain? Must efficiency preclude optimality?

5 HIERARCHIES OF MACRO CELLS: EFFICIENCY *AND* OPTIMALITY

5.1 Macro Operators

In creating a path from one point on a street map to another point, there are a limited number of primitive operations that can be performed. When you are at a street intersection you can either continue to follow that street, or you can turn onto an adjacent street (traffic laws permitting). A *macro* is a set of primitive operators that are combined into one higher level operation. For instance if you are a New York City cabby you may possess a New-Jersey-to-Queens macro which is simply the set of fifty-seven different intersections that must be navigated to take you from the Lincoln Tunnel into Manhattan to the Queens-Midtown tunnel out of Manhattan.

If you were a particularly methodical taxi driver and you wanted to be able to easily calculate the quickest path from anywhere in New Jersey to anywhere in Queens you could invest a week of your time in memorizing the best routes across Manhattan from each of the bridges and tunnels to all of the other bridges and tunnels. This would give you a complete set of Manhattan-traversing macros. From that point on whenever you were calculating routes from New Jersey to Queens, the Manhattan portion of your trip would be a "black box": you'd consider all of the ways of entering and exiting the island, but the actual traversal paths and their costs would be known quantities. While the portion of your search taking place in New Jersey and in Queens has not been simplified, the Manhattan portion of the search has been greatly streamlined (see Figures 2 and 3). But is the usefulness of macro sets for routing on city streets limited to special cases (such as islands or peninsulas) or can this tool be used for all city maps?

5.2 Cells Of Macro Operators

The property of Manhattan that made our Manhattan-traversing macro set useful was the fact that, while there were a large number of graph nodes composing the representation of the island's street map, relatively few of these nodes had any links to points outside of Manhattan.

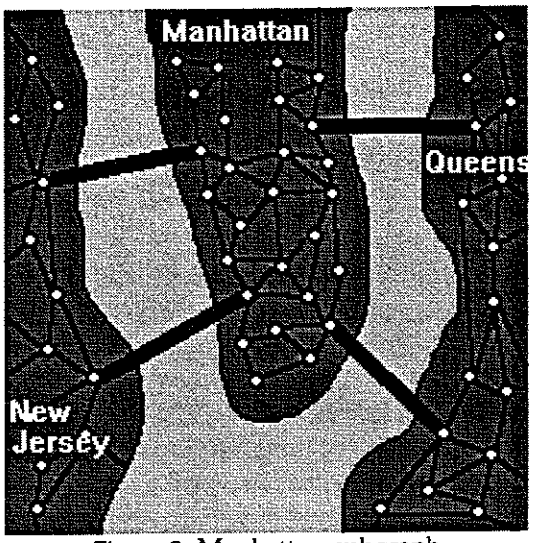


Figure 2: Manhattan subgraph without macro set.

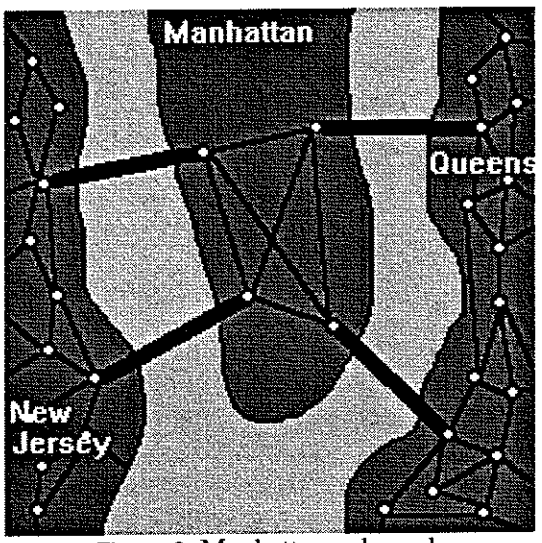


Figure 3: Manhattan subgraph simplified with macro set.

If these few nodes with off-island connections are termed *border* nodes (with the remaining nodes being termed *interior* nodes), then the cost-saving value of the macro set is proportional to the number of border nodes divided by the total number of nodes in the region.

But what about regions that do not contain geographic features suitable for creating a set of macro operators? In general a city graph can be divided into a set of subgraphs or *cells*. Each of these cells can, through pre-exploration, be given a set of macro operators that define all of the possible (optimal) traversals of that cell.

For the Taipei city map, the cells consisted of one square kilometer regions.

For the Taipei sample data the one kilometer subdivisions created grids that each had, on the average, 319 total states (nodes), of which 29 were border nodes. So any search on the sample data would involve at most two one square kilometers regions with (on the average) 319 possible nodes expanded, with each of the remaining graph regions having no more than (on the average) 29 nodes expanded.

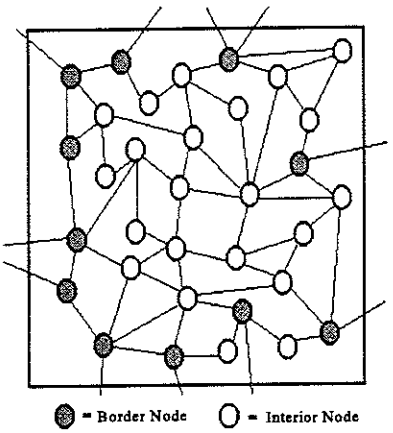


Figure 4: Cell marked for macro creation.

A complete set of macro cells was created for the Taipei map by running Dijkstra's Algorithm once for each border node in each cell and storing the paths and the path traversal costs for each border node encountered. Using a PC, the creation of all the macro operators needed for the forty square kilometer map took 25 minutes with the resulting data requiring two megabytes of storage (see Figure 5).

5.3. Hierarchies Of Macro Cells

Using one level of macro cells can significantly speed up search, but it can only do so by a constant factor. We can achieve additional savings by going one step further. In addition to systematically combining operators together to pre-solve all cell traversals, we can combine macro operators together into macros of macro operators. Four adjacent first-order cells can be combined into one second-order macro cell (and four second-order into one third-order, etc.).

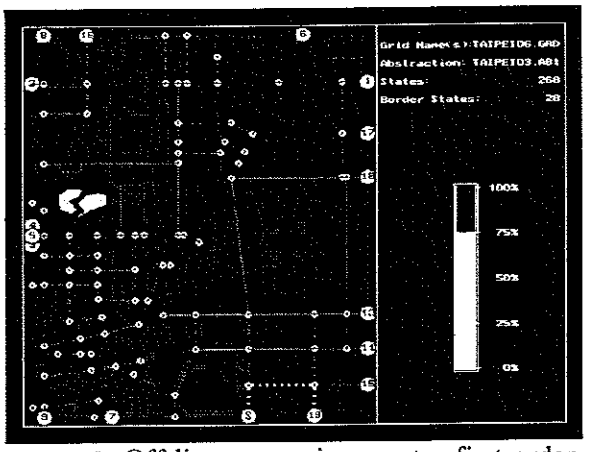


Figure 5 : Off-line processing creates first order macro cells using Dijkstra's Algorithm.

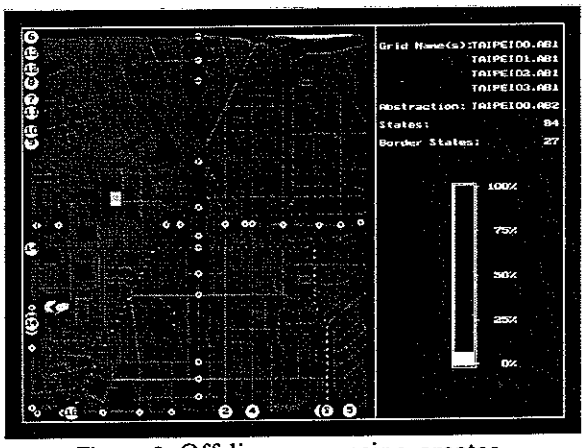


Figure 6: Off-line processing creates second-order macro cells using Dijkstra's Algorithm with macro operators.

As only border-to-border solutions are required and as the solutions are given as combinations of (typically two) macro operators, the second-order cells can be computed and stored in an efficient manner. The time required to compute all of the second order cells for the Taipei map was roughly the same as was required for the first order cells, about half an hour. The storage requirement for the second level was significantly smaller than for the first order (0.6 MB versus 2 MB), since most traversal paths for the second order had only two (macro operator) elements. First order traversal paths typically had six or seven (base operator) elements.

5.4 Using Macro Cells In An A* Search

Given that a hierarchy of macro cells has been created for a map, there are very few changes to the basic A* algorithm that are required to perform a path search on that map. The main modification is in the process of node expansion. If a parent node is in the same first-order grid as the start or goal nodes, the set of child nodes is found in the usual manner. Otherwise, the child set is defined as all the border nodes of the largest-order macro cell containing the parent but not containing either the start or the goal.

5.5 Search Costs

The worst-case performance of an A* search is determined by the maximum number of nodes that can be expanded in the course of the search. This number is the product of the area of the region that contains expandable nodes and the density of nodes per unit volume within this region (Figure 7). The formula for the maximum number of nodes expanded by an A* search of a given length, d, using a hierarchy of macro cells with H levels is:

$$N_{max}(d, H) \leq$$

$$kP_0d^2 \qquad \text{for } H = 0$$

$$MIN[kP_1 2^{1-H}d^2 + 2L^2(P_0+P_1(2^H-3)),\ N_{max}(d,H-1)] \qquad \text{for } H > 0$$

Equation 1: Maximum Nodes Expanded With Varying Macro Cell Levels.

Where P_0 is the node density of the graph and P_1 is the density of border nodes. If *a full* macro cell structure is used (i.e., if for a map of M by M size with a base cell size of L by L, the level of macro hierarchies, H, is equal to $\log_2(M/L)$) then the maximum number of nodes expanded in an A* search is given by:

$$N_{Max}(d) \leq 2L^2(P_0-3P_1) + (3LP_1\sqrt{2k})d$$

$$\text{for } 0 \leq d \leq M\sqrt{2/k}$$

Equation 2: Maximum Nodes Expanded With Full Macro Cell Structure

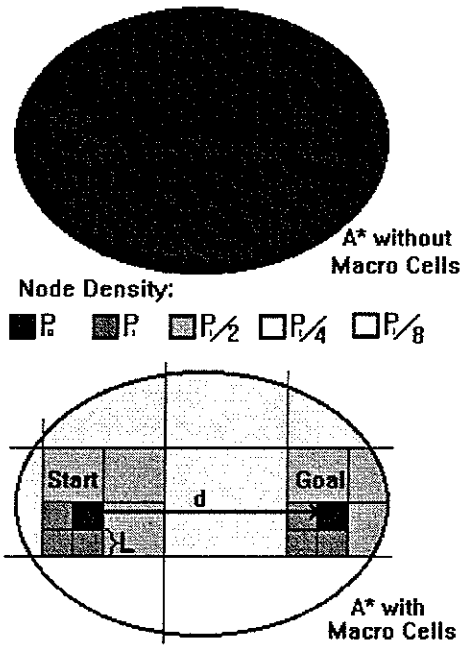


Figure 7:Search Cost (search area times node density)

As a result, the upper bound on the number of nodes expanded is linear to the distance, d, of the route planned--given that $k \leq 1$. Sample routes on the Taipei street map grid yielded a k value of 0.70. With these results a 100 kilometer route plan, which would require 2,233,000 node expansions without macro cells, requires just 10,758 node expansions using macro cells.

Figure 8: Nodes expanded in an A* search using two levels of macro cells (H=2).

5.6 Evaluation Of Macro Cells

Using the four evaluation criteria outlined earlier, the use of hierarchies of macro cells looks to be an effective method of computing routes on city street maps. The cost savings are significant: O(d) vs. $O(d^2)$. The resulting search is both complete and admissible. The resources required to create and use macro cells are reasonable, even for immense maps: a map the size of the entire U.S. would require a (per unit area) overhead only four times the size required by the Taipei map. Finally, macro cells allow for efficient incremental updating of the cell representation: a change to the map data due to changing traffic conditions would require only the recomputation of a limited number of cells.

6 FUTURE WORK

Work is underway to extend the use of macro cells to include graphs with discrete time-varying features (for example, traffic patterns that change at different times of the day). Preliminary results indicate optimal results with linear costs are attainable.

7 REFERENCES

[1] Thomas H. Corman, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 1990.

[2] Edsger Dijkstra. "A Note on Two Problems in Connection with Graphs". *Numerische Mathematik* 1/1959, 269-271.

[3] Yasuomi Fujita, Takayoshi Nawaoka, and Eriko Hiroata. "Development of a Mapping and Guidance Database for Automobile Navigation Systems". *IEEE Computer Society Press.* CH2944-7/91/0000/0869, 869-874.

[4] R.C. Holte, M.B. Perez, R.M. Zimmer, and A.J. MacDonald. "Hierarchical A*". *Symposium on Abstraction, Reformulation, and Approximation.* August 1995.

[5] R.C. Holte, T. Mkadni, R.M. Zimmer, A.J. MacDonald. "Speeding Up Problem-Solving by Abstraction: A Graph-Oriented Approach". technical report TR-95-07, Computer Science Dept., University of Ottawa.

[6] Richard E. Korf. "Search: A Survey of Recent Results". H.E. Shrobe, ed., *Exploring Artificial Intelligence*, Morgan-Kaufmann, 1988, 197-237.

[7] Richard E.Korf." Optimal Path Finding Algorithms", *Search in Artificial Intelligence.* Springer-Verlag, 223-267.

[8] Bing Liu, Siew-Hwee Choo, Shee-Ling Lok, Sing-Meng Leong, Soo-Chee Lee, Foong-Ping Poon, and Hwee-Har Tan, National University of Singapore. "Finding the Shortest Route Using Cases, Knowledge, and Dijkstra's Algorithm." *IEEE Expert. Oct. 1994, 7-11.*

[9] Hale Montgomery. "New U.S. Leadership Takes Up GPS, GLONASS Issues", *GPS World,* June 1993.

[10] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, Reading, MA, 1984.

[11] Denny Rock, Doug Hoskins, and Don Malkoff. "Intelligent Road Transit: The Next Generation". *AI Expert,* April 1996, 16-24.

CONTINUITY-GUIDED REGENERATION: AN APPROACH TO REACTIVE REPLANNING AND RESCHEDULING

Alexander Kott
and
Victor Saks

Carnegie Group, Inc.
Five PPG Place
Pittsburgh, PA 15222
saks@cgi.com

Abstract

We describe a reactive replanning technique developed to enable transportation planning in a highly dynamic, complex and critical domain: the military aeromedical evacuation of patients to medical treatment facilities. In early 1993, the Department of Defense tasked the U.S. Transportation Command (USTRANSCOM) to consolidate the command and control of medical regulation (assigning patients to medical treatment facilities) and aeromedical evacuation operations during peace, war and projected contingencies. The ensuing analysis led to TRAC2ES (TRANSCOM Regulating and Command and Control Evacuation System), a decision support system under development for USTRANSCOM (Johnson, 1994). Probably the most challenging aspect in planning and scheduling medical evacuation operations has to do with the dynamics of a domain in which requirements and constraints continuously change over time. Reactive replanning is a key capability of TRAC2ES. The key concern in reactive replanning is plan continuity -- the new plan should take into account the current plan and should not introduce unnecessary disruptions. In this paper, we discuss this aspect of reactive replanning in depth, and propose an approach to reactive replanning called Continuity-Guided Regeneration (CGR) that provides explicit treatment of the requirement of plan continuity.

1 INTRODUCTION

This paper describes an approach to reactively modifying a plan and a schedule of multiple interdependent activities in near real-time. The predictive (also called static) formulation of this problem assumes that all activities are to take place in the future. In the reactive (dynamic or real-time) problem, some of the activities are occurring at the same time that the planning problem is being solved.

We argue that the key concern in reactive replanning is plan continuity -- the new plan should take into account the current plan and should not introduce unnecessary disruptions. In this paper, we discuss this aspect of reactive replanning in detail, and propose an approach to reactive replanning called Continuity-Guided Regeneration (CGR) that provides explicit treatment of the requirement of plan continuity.

Although this work and the CGR approach in general is not limited to a particular problem instance, it was particularly motivated by an application domain in which transportation planning is highly dynamic, complex and critical: the military Aeromedical Regulation and Evacuation (ARE) of patients to medical treatment facilities. The ARE problem is a complex extension of the well-known Dial-A-Ride-Problem (DARP) (Kott, 1995). To these authors, the ARE problem became an important measuring stick for determining viability and practicality of models and techniques for reactive replanning. In this paper, we summarize some of the major characteristics of this problem.

One may distinguish two major classes of approaches to both predictive and reactive replanning -- constructive techniques and iterative techniques.

Constructive techniques e.g., (Psaraftis, 1983, SmithD, 1992, Kikuchi, 1989, Sadeh, 1991, Sadeh, 1993, Johnson, 1994, Saks, 1992), build solutions by incrementally instantiating decision variables (e.g. assignment of patients to mission and creation of new missions) until a complete feasible and satisfactory solution is obtained. These techniques can be seen as a search in a space of partial feasible solutions.

In contrast, iterative procedures (many of which are called iterative repair) can be viewed as a search in a

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/83 ©1996 FLAIRS

space of complete solutions (i.e., solutions in which all decision variables have been instantiated). Some solutions within this space may be infeasible -- they violate constraints. The iterative solution process uses transformation operators to move from one complete solution to another until a feasible/satisfactory solution has been found. For example, in the ARE domain such transformation could include reassigning a patient from one mission to another or modifying the itinerary of a mission. Examples of different classes of iterative techniques include (Miyashita, 1993, Minton, 1990, Zweben, 1991, Osman, 1992).

Constructive and iterative techniques are not incompatible but can sometimes be viewed as complementary: simpler constructive approaches can provide a basis for quickly producing new though potentially highly sub-optimal solutions while iterative techniques offer a way to spend additional time post-processing these new solutions to improve their quality.

We argue that when searching for a solution to a reactive rescheduling problem, one must attempt to minimize the extent of disruptions that the new plan introduces into current execution activities. Indeed, at any time that we issue reactive changes to a current plan, we violate some commitments and waste some investments made into preparing or executing the activities scheduled in the current plan.

Thus, the issue of continuity/stability is key to effective reactive replanning. In fact, it is this dimension that distinguishes reactive replanning from repetitive application of predictive planning. Prior research has not addressed this issue in a systematic and explicit manner. Our approach addresses the plan continuity requirement explicitly and in a manner that provides a degree of control over the extent of plan continuity.

2 AN EXAMPLE PROBLEM

U.S. Transportation Command (USTRANSCOM) is the DoD agency responsible for evacuating patients during wartime and peace. Patients requiring extended treatment must be evacuated by air to a suitable Medical Treatment Facility (MTF). The process of identifying an MTF that constitutes a suitable destination for a given patient (based on matching the patient's medical condition and MTF's capability, and on economics and transportation availability) is called regulating. The process of routing and scheduling the required aeromedical evacuation flights (missions) and assigning patients to suitable missions is evacuation planning and execution.

The Persian Gulf war was the first significant armed conflict in which this concept has been put to a serious test. The results were far from satisfactory -- about 60% of the patients ended up at the wrong destinations and half in the wrong country (Endoso, 1994).

In early 1993, the Department of Defense tasked USTRANSCOM to consolidate the command and control of medical regulation and aeromedical evacuation operations during peace, war and projected contingencies. The ensuing analysis led to TRAC2ES (TRANSCOM Regulating and Command and Control Evacuation System), a decision support system under development for USTRANSCOM (Johnson, 1994).

The integrated medical regulation/evacuation problem requires the dynamic identification of appropriate Medical Treatment Facilities (MTFs) for new patients and the planning/scheduling of aeromedical evacuation operations to transport these patients from current locations to selected MTFs. This is a large-scale, highly dynamic planning and scheduling problem that can involve hundreds or even thousands of simultaneous patient movement requests. Each patient has one or several medical requirements that constrain the type of MTF to which he or she can be evacuated and a ready-time prior to which evacuation cannot start. Additional constraints can include a maximum number of hours that a patient can spend in a flight before requiring an overnight rest, a maximum number of stops the patient can tolerate during evacuation, etc. Planning/scheduling operations in this domain require the dynamic coordination and (re)allocation of a large number of resources subject to a wide variety of constraints. Key assets/resources and associated constraints include aircrafts and their different characteristics (e.g. capacity, refueling requirements), air and medical crews and restrictions on the number of hours they can work in any given day, airports and their different characteristics (e.g. capacity, types of aircraft they can accommodate), hospital beds at MTFs located all around the globe and the types of patients each MTF can accommodate, etc.

Probably the most challenging aspect in planning and scheduling medical evacuation operations has to do with the dynamics of a domain in which requirements and constraints continuously change over time. New patient requests come in, others get canceled. Patient conditions change over time, both prior to and during evacuation, possibly requiring the delay, acceleration or cancellation of a patient's evacuation or a change in the patient's destination. Availability of key assets is also subject to unpredictable events (e.g., aircraft maintenance problems, hospital beds not getting freed on time, airfield attrition). Further complicating the problem, weather conditions can affect evacuation plans at anytime, requiring that a mission be delayed, rerouted or simply canceled.

In building and revising evacuation plans a number of objectives and preferences need to be taken into account. The number of patients evacuated to adequate MTFs has to be maximized, with urgent patients given priority over routine ones. The time to pickup and deliver patients to their MTFs, especially urgent ones, also has to be as short as possible. Simultaneously, the time each patient flies and the number of stops during his/her evacuation have to be minimized. Other important considerations include minimizing the number of missions and maximizing aircraft capacity utilization.

One preference is particularly critical and requires a special discussion. When searching for a solution to a reactive rescheduling problem, one must attempt to minimize the extent of disruptions that the new plan introduces into current execution activities. Indeed, at any time that we issue reactive changes to a current plan, we violate some commitments and waste some investments made into preparing or executing the activities scheduled in the current plan.

For example, significant efforts are expended to prepare for a particular mission: flight and medical crew are assembled and briefed, paper work is issued, flight controllers insert this mission into their plans, maintenance and refueling resources are identified, scheduled and allocated, etc. By changing a mission, we negate some of these investments and force expenditure of additional resources to undo the effects of some of the activities.

Similarly, execution of a patient itinerary involves preparation and investment of resources: medical crew flight planning, configuring the plane, procuring the necessary instruments, preparing the food according to the patient's diet, issuing the necessary paper work and notifications. When an itinerary is modified, all or most of these investments are lost, and additional effort is needed to cancel the preparations for the initial itinerary and to prepare for the new itinerary.

Investment of personnel and material resources is not the only casualty of plan changes. Increased risk is another harmful effect. A plan change increases the risk of mishaps, miscommunications, erroneous data entries and erroneous decision-making. Thus, a mandatory feature of a reactive replanning process is the ability to minimize the number of changes -- violations of commitments made by the earlier plan.

It is useful to note that reactive replanning problems, e.g., the ARE problem, can be seen as constrained optimization problems. The example problem formulation can be summarized as follows:

- Given: current data of patients, missions, MTFs, ASFs, airfields, current executing Plan...
- Find: mission schedules and patient itineraries
- Subject to: list of constraints...
- Optimize or Prefer: minimum time to delivery, maximum patient coverage, minimum resources; minimum violations of prior commitments, etc.

4 OUR APPROACH -- CONTINUITY-GUIDED REGENERATION

4.1 Experience with Redesign Problems

Continuity-Guided Regeneration(in earlier publications we referred to the same approach as "Commitment-Constrained Rescheduling.") has been developed as an extension of the ideas originally developed by the first author for solving re-design problems, circa 1990, within the SPEx design and configuration shell (Kott et al., 1992, Kott et al., 1990, Berry, 1992). The key idea is to regenerate the plan while using the currently executing plan as a constraint on the solution. One significance of this idea is that it assures plan continuity without having to answer the difficult question of how much of the current plan to undo.

Based on our experience with SPEx, a key advantage of the Continuity-Guided Regeneration can be summarized as follows. It eliminates the key dilemma of the redesign problem: how much of the prior design to undo. If too much is undone, too many design decisions will be revised, unnecessarily. If too little is undone, the problem solving procedure does not have enough "wiggle room" to make optimized or even feasible decisions. Even though there are approaches to resolving this dilemma, e.g., by encoding the domain-specific redesign rules, or by analyzing the dependencies within the space of prior design decisions, the Continuity-Guided Regeneration approach allows this difficult issue to be avoided altogether by allowing all prior decisions to be undone and then relying on CGR to ensure that that the new solution minimize disruptions.

The key idea, like in SPEx, is to use the current plan as an additional preference constraint in generating a new plan. Unlike in SPEx, however, we use a variable degree of commitment that depends on the lead time of the activity and possibly on other domain-specific factors.

4.2 Sliding Scale of Commitment

It is important to note that the less time is left between the current moment and the time when the activity is planned to happen, the costlier is the change. In other words, the cost of disruption and rework associated with a change to a currently executing plan tend to be much higher for those changes that affect activities scheduled for the near future. For example, it is much more expensive to make a change to a mission that takes off in 20

minutes than to a mission that is scheduled to take off in 20 hours.

Other factors in addition to time increase the cost of disruption. One such factor is the extent to which the decision has been communicated to the world. For example, if an MTF has already been notified of the arrival of a patient, the cost of sending the patient elsewhere is increased. Lead time required to implement the decision is also significant. Thus if the MTF has already begun preparations to receive the patient, then the cost goes up even more.

These observations led to Sliding Scale of Commitment (SSC) - a concept proposed by the second author and W. Elm at Carnegie Group, Inc., circa 1993. The observation is that the key to reactive rescheduling is to minimize violations of commitment made in the current executing schedule.

Specific components of the SSC concept include:

- there is a cost (we call it the commitment cost) to be paid for rescheduling, even for actions yet to be executed;
- the cost is a function of time, the closer -- the costlier;
- the cost may also be dependent on domain-specific details of commitment, e.g., who has been notified and what preparations are necessary.

5 CONTINUITY-GUIDED REGENERATION IN PLANNING AND SCHEDULING

We extended the Continuity-Guided Regeneration idea of SPEx to make it capable of accounting for the sliding scale of commitment.

The key idea, like in SPEx, is to use the current Plan as an additional preference constraint in generating a new Plan. Unlike in SPEx, however, we use a variable degree of commitment that depends on the lead time of the activity and possibly on other domain-specific factors. To make this idea more concrete, we describe how we applied it to modify a particular scheduling approach, micro-opportunistic search, an instantiation of Constrained Heuristic Search (Sadeh, 1991, Sadeh, 1994, Saks, 1992). The main steps of the micro-opportunistic search procedure can be summarized as follows:

1. A set of candidate plans is created for each requirement (e.g., a manufacturing "job" or a request to transport a particular entity);
2. "Reliance" (dependencies) of the requirement on each of the available resources are computed;
3. Each requirement distributes its demand to the resources and time line;
4. Overall contention for each resource is computed by time bucket ;
5. The most contended for resource is selected;
6. The requirement that is most reliant (dependent) on the selected resource is assigned to the resource.
7. Go to 2 until all requirements are assigned.

We modified this approach using the idea of commitment constraint. We elected to use the reliance computation as the area of the search procedure where the commitment constraints enter the picture. For each requirement (in our specific case - a patient movement request):

1. retrieve the resource which has been committed to this requirement in the existing plan:
2. compute a measure of Importance of Preserving the same Commitment (IPC) based on how far in the future the use of this resource is to take place and also considering other domain-dependent commitment measures;
3. increase reliance measure associated with this requirement-resource pair using the IPC;
4. use the assignment mechanism to assign a resource to the requirement. The higher reliance value leads to a better chance that the requirement will be assigned to the same resource as in the existing plan, if such an assignment is feasible.

We believe that the Continuity-Guided Regeneration paradigm can be used as an add-on with a variety of planning/scheduling approaches of either iterative or constructive nature. We found it straightforward to implement this idea within the framework of the micro-opportunistic search.

Just as in the redesign problem (discussed earlier in relation to SPEx), a key advantage of the Continuity-Guided Regeneration approach is that it eliminates the key dilemma of the re-scheduling problem: how much of the prior schedule to undo. The Continuity-Guided Regeneration approach avoids this difficult issue altogether. However, a run-time performance issue may surface in the case of a large plan with very few disruptive events. In such a case, a program that unschedules only a small portion of the plan, and then replans only the unscheduled assignments, may run significantly faster than the CGR-based algorithm that regenerates the entire plan. Even in this case, however, the CGR approach can be beneficial since relatively crude and inexpensive techniques may be used to decide which assignments to unschedule, followed by continuity-guided regeneration of the unscheduled portion of plan.

6 RELATION TO PRIOR WORK

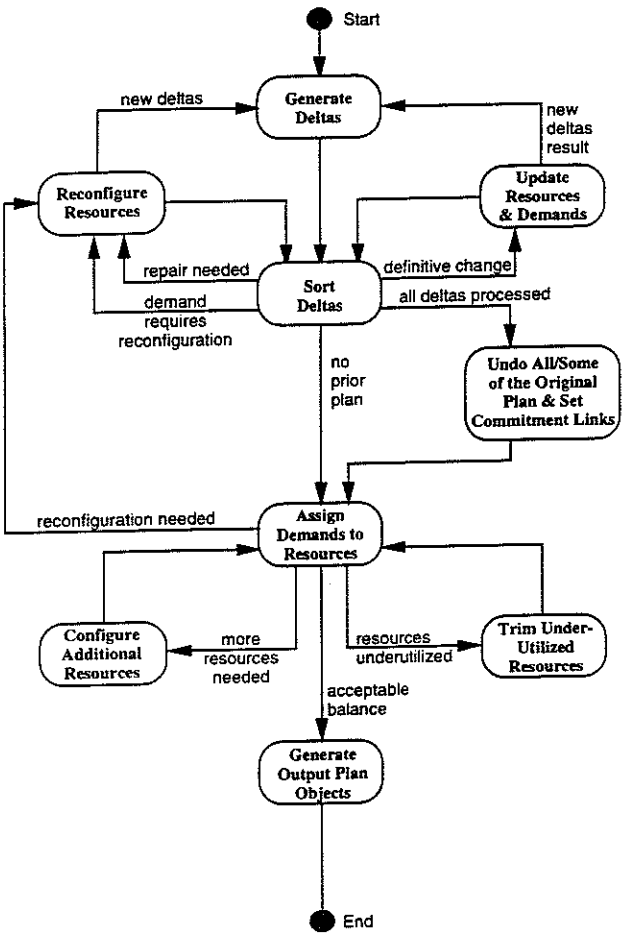
Fox (Fox, 1994) describes the ISIS schedule repair method which has a key similarity to the Continuity-Guided Regeneration (CGR), although there are also

some important differences. The similarity is that some reservations of the old schedule are turned into preference constraints for construction of the new schedule. Wilkens (Wilkens) describes a chess playing program that used previously assembled plans to focus search for plan extensions, in light of the opponent's moves. In should be pointed out that our approach has evolved independently, based on the SPEx work. The fact that other researchers have been intrigued by similar ideas in the past reinforces our belief that this class of approaches is highly promising.

7 AN EXAMPLE OF APPLYING THE CGR APPROACH

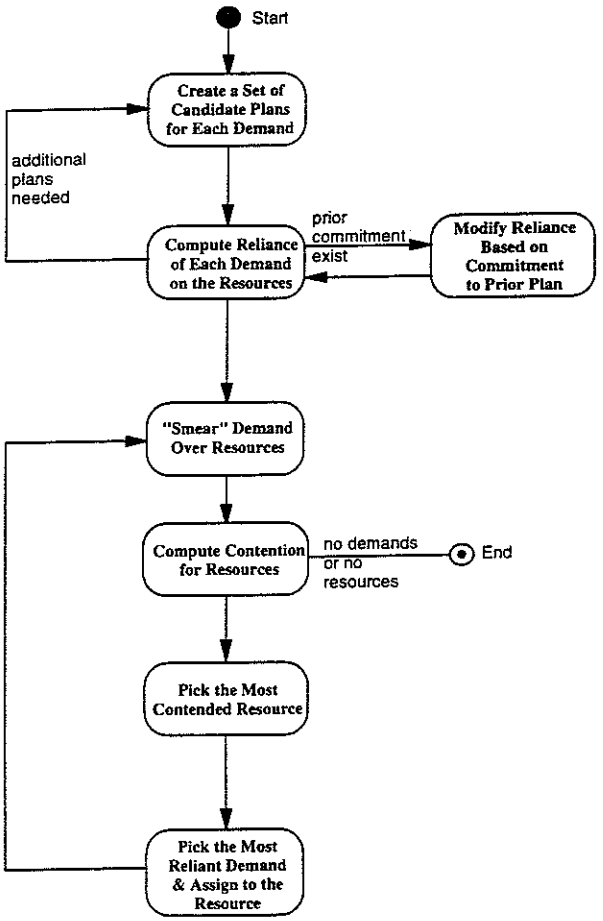
We applied the CGR approach to solving the ARE problem described earlier in this paper. The flow of the solution process is depicted in Figures 1 and 2.

Figure 1. Overall Flow of the Reactive Replanning Process.



In our initial implementation, the disruptive events, such as closure of an airport, are loaded into the system as textual statements. Each statement is parsed and decomposed into one or more of so called "world deltas." Each world delta is a description of a difference between the expected state of an entity (e.g., airport is open) and its actual state (e.g., airport is closed). All deltas are entered into the database.

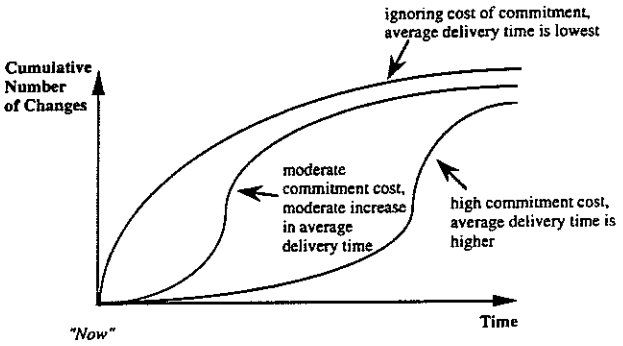
Figure 2. Details of Assigning Demands to Resources.



The deltas are sorted, using domain-specific heuristics, in the order of significance and downstream impact. This queue of deltas is then processed. Depending on the nature of each delta, the algorithm then may create new deltas, update attributes and associations of existing resources and demands, or reconfigure existing resources (missions). In particular, the following operations are performed on missions:

- update modified missions with a complete route/schedule given;

Figure 3. In general, the expectation is that higher commitment costs cause the shift of plan changes to later times.



- "repair" missions which have been diverted for some reason and for which continuation of the mission is not given;
- re-route missions, if necessary, to provide suitable route for new urgent patients.

All patient itineraries that have been in any way affected by any of the deltas are marked as invalid. For example, a patient itinerary that includes missions which have been modified is marked as invalid. This "undoing" of itineraries in the original plan is done liberally, i.e., we prefer to err on side of undoing more itineraries than necessary given that the Continuity-Guided Regeneration provides continuity. In fact, some of these authors argue that this undoing step is unnecessary -- all itineraries of the original plan can be marked as invalid.

Now the process of assigning patients to resources (e.g., missions and hospitals) is begun, in the following order.

1. Itineraries marked invalid are undone.
2. The Assign Patients method is then performed (Figure 2). The method implements the Continuity-Guided Regeneration approach along the lines of section 5. Specifically, an assignment of a patient to the same resources (missions, hospitals, etc.) as in the original plan is given a higher score. The increase in score is computed in inverse proportion to how far in the future the use of the resources begin (to account for the Sliding Scale of Commitment).
3. A report is generated which describes the impact of the world deltas.

8 EXPERIMENTAL RESULTS

The basic experimental scenario involved 4 airfields, 5 hospitals, 34 missions and 136 patient movement requests (PMRs), including urgent and routine patients with a variety of treatment needs. The operations of this scenario extended over a 5 day period. Scenarios involved a variety of disruptive events and caused large-scale changes in the plan (around 50% of all itineraries had to be modified to accommodate the effects of disruptive events).

The tests confirmed that this algorithm is capable of performing replanning for arbitrary combinations of several types of disruptive events. We also focused on the effect of commitment cost on the number of plan changes and on other characteristics of the solution. Figure 3 depicts the idealized expected effect of commitment cost on the number of plan changes. In general, the test results confirm our expectations. Details of the experimental results are being published elsewhere.

9 CONCLUSIONS

Plan continuity is a key concern in devising a reactive replanning technique. It is the key differentiator between predictive and reactive planning.

Approaches to reactive replanning can be divided into two broad classes -- constructive and iterative -- and we found that these two approaches can be, and possibly should be, combined. Iterative techniques can be used as both pre-processing and post-processing mechanisms with constructive techniques. The prior work in both of these two schools of thought has not addressed the plan continuity issue explicitly.

Particularly in the constructive approaches to reactive replanning, one of the key difficulties is to determine how much of the current plan should be "undone."

Our prior work in design and redesign led to the idea of Continuity-Guided Regeneration -- regenerate the plan while using the currently executing plan as a constraint on the solution. One significance of this idea is that it assures the plan continuity without having to answer the difficult question of how much of the current plan to undo.

Our implementation of this approach in application to aeromedical evacuation demonstrated that this approach fully addresses the continuity issue, eliminates the "extent of undoing" question, and provides an effective mechanism to control the balance between the demands of continuity and optimality. Depending on the characteristics of the command and control system, as well as cost concerns, the replanning algorithm can be tuned to trade plan optimality (impact on cost) against plan continuity (impact on command and control efficacy). The tuning involves merely adjustment to the commitment cost parameter. Eventually we may want to control different decisions/activities separately.

This approach provides a natural mechanism to address the fact that it is more important to preserve

some planning decisions than others (sliding scale of commitment) and leads to a solution process that makes no distinction between predictive and reactive planning.

Finally, the approach provides a way to convert a variety of predictive planning techniques into integrated predictive-reactive techniques.

10 ACKNOWLEDGMENTS

This work was supported by the United States Transportation Command. The authors are indebted to MA J. Samson (USTRANSCOM), Mr. R. Samson (MITRE) and Mr. W. Elm (Carnegie Group, Inc.) who provided valuable insights, to Carnegie Group engineers Alida Skogsholm, Richard Nill, Sandeep Chand and Jeff Stonebrook, who designed and built the experimental software, and to Ivan Johnson who suggested numerous improvements in this paper.

References

Berry, F. and Kott, A. The SPEx Shell: an Approach to Automating Application and Sales Engineering Processes. In Proceedings of the 1992 ASME Computers in Engineering Conference. ASME, 1992.

Endoso, J. TRANSCOM wants casualties to get where they belong. Government Computing News, May 1994.

Mark S. Fox. ISIS: A Retrospective. In Mark Fox and Monte Zweben (Eds.). Intelligent Scheduling. Morgan Kaufmann Publishers, 1994.

Johnson, I, V. Saks, MAJ. J. Simpson and R.H. Simpson. Constraint-Directed Object-Oriented Medical Evacuation Planning. In American Defense Preparedness Association Conference. , 1994.

Kikuchi, Shinya and Jong-Ho Rhee. Scheduling Method for Demand-Responsive Transportation System. Journal of Transportation Engineering, November 1989, 115(6), 630-645.

Kott, A., Sadeh, N. Survey Paper: Models and Techniques of Dynamic Demand-Responsive Transportation Planning. Working Paper, Carnegie Group, Inc., Pittsburgh, PA, 1995

Kott, A., Agin, G., and Fawcett, D. . Configuration Tree Solver. A Technology for Automated Design and Configuration . ASME Journal of Mechanical Design , March 1992, 114(1), 187-195.

Minton, S, M.D. Johnston, A.B. Philips, P. Laird. Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method. In Proceedings of the Eighth National Conference on Artificial Intelligence. , 1990.

Miyashita, K. and Sycara, K., Predictive and Reactive Scheduling through Iterative Revision. In Proceedings of the IJCAI-93 Workshop on Knowledge-based Production Planning, Scheduling, and Control. Chambery, France: , 1993.

Osman, I. Meta-Strategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem (Tech. Rep.). Canterbury, Kent CT2 7NF, UK: Institute of Mathematics and Statistics, University of Kent at Canterbury, 1992.

Psaraftis, H.N. An Exact Algorithm for the Single Vehicle Many-to-Many Dial-A-Ride Problem with Time Windows. Transportation Science, August 1983, 17(3), 351-357. Technical Note.

Sadeh, N. Look-ahead Techniques for Micro-opportunistic Job Shop Scheduling. Doctoral dissertation, School of Computer Science, Carnegie Mellon University, March 1991.

Sadeh, N., S. Otsuka, and R. Schnelbach. Predictive and Reactive Scheduling with the Micro-Boss Production Scheduling and Control System. In Proceedings of the IJCAI-93 Workshop on Knowledge-based Production Planning, Scheduling, and Control. Chambery, France: , 1993.

Sadeh, N. Micro-Opportunistic Scheduling: The MICRO-BOSS Factory Scheduler. In Mark Fox and Monte Zweben (Eds.), Intelligent Scheduling. Morgan Kaufmann Publishers, 1994.

Saks, V., Al Kepner, and Ivan Johnson. Knowledge Based Distribution Planning. In American Defense Preparedness Association Conference. , 1992.

Wilkens, D. Using Knowledge to Control Tree Searching. In Artificial Intelligence, 1982, 18, 1-51.

Zweben, M., Eugene Davis, and Michael Deale. Iterative Repair for Scheduling and Rescheduling (Tech. Rep.). MS 244-17, Moffett Field, CA 94035: NASA Ames Reserch Center, 1991.

FLEXIBLE SIMULATION SCENARIOS FOR REAL-TIME PLANNING IN DYNAMIC ENVIRONMENTS

Klaus P. Jantke[†], Oksana Arnold[‡] and Torsten Lehmann[†]
[†]HTWK Leipzig
{jantke,torsten}@informatik.th-leipzig.de
[‡]Leipzig University
oki@wifa.uni-leipzig.de

Abstract

The problems and solutions adressed in this paper belong to the area of knowledge-based process supervision and control. The focus is on the interaction between a therapy planning and execution system, on the one hand, and a simulation module, on the other hand. Generating therapy plans for seriously disturbed dynamic processes inevitably needs at least approximate knowledge about the future. At a first glance, there is a simple scenario of interaction where a therapy planning system is asking a simulation system for the expected values of certain process parameters within some time interval of interest. A closer look exhibits a number of difficulties. The key problem is that the intermediate results of planning are essential for the work of the simulation module. As a consequence, there has to be a proper interaction of these knowledge processing components. This requires extra efforts of control. The interaction itself is becoming subject to in-depth investigations. There is a tradeoff between efficiency and accuracy. A timely response of the therapy planning system requires efficient reasoning procedures in many places. On the other hand, erroneous decisions during plan generation lead to the necessity of replanning. The present investigation of simulation scenarios is intended to identify parameters of the reasoning process which are fundamental for meeting real-time requirements.

1 INTRODUCTION

The present paper is embedded into a collection of individual approaches dealing with several aspects of knowledge-based process supervision and control. Other approaches deal with *knowledge representation*, *knowledge acquisition*, *system architecture*, *fault diagnosis*, *therapy planning*, and *simulation* itself. The focus of our present paper is *not* on one of the individual inference processes, but on *interaction*. To make it explicit, this is *not* a paper on simulation or planning, but on *scenarios of flexible interaction of planning and simulation*. Additionally, it will turn out that certain aspect of *representing dynamic knowledge* play a crucial role for the overall approach.

All the individual issues of knowledge representation and processing, of system architecture and interaction and so on are coordinated within a comprehensive approach to *knowledge-based process supervision and control* which has been developed within a larger joint research project. It's the crux of the present paper that understanding the local problems and solutions inevitably assumes global knowledge. Therefore, we decided to put more emphasis than usual on secondary issues related to the main topic.

1.1 Motivation

In German industry, more than 60% of all larger accidents happen in chemical installations. The dominating reason (almost 40%) for loosing control of a disturbed dynamical process is the inability of human operators to find appropriate action plans when faced to an overwhelming flood of information. The difficulty is more the complexity of underlying knowledge and recent information than the requirement of a timely response.

The idea of knowledge-based process supervision and control is to extend conventional process supervision and control systems by a knowledge-based component. The aim of the present paper is twofold. First, it is intended to propagate the overall concept of knowledge-based process supervision and control and to point to a few interesting aspects of this comprehensive approach. Second, the focus of the paper is on *communication issues* which have not been considered so far.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/90

Technical therapy planning is only invoked, when some given target process is seriously disturbed such that conventional process control fails to drive the process back into an acceptable working regime. The following aspects have an essential impact on the present investigation:

- Complex dynamic systems like chemical processes, for instance, have a score or more peculiarities which have to be taken into account for developing, implementing and applying appropriate planning concepts. In fact, these peculiarities are motivating the need for flexible process simulation.
- Planning is integrated into a comprehensive approach to knowledge-based process supervision and control. In fact, one had to develop this overall concept first, before going into details of simulation scenarios.

[AJ94b] contains a comprehensive discussion of peculiarities of typical target processes.

It is one of our key insights (cf. [AJ94b], [JA95b] and [JA96]) that classical STRIPS-like planning approaches (cf. [FN71], e.g.) are not adequate for really complex dynamic processes. One of the crucial problems is that actions may not have well-defined post conditions. As a consequence, plans are more hypothetical in character and plan generation is becoming inductive program synthesis (cf. [AJ94a] and [JA96]).

1.2 Focus of Presentation

This paper is based on two implemented planning systems running in Allegro Common Lisp (cf. [Arn92] and [Arn93]) and in Quintus Prolog (cf. [Mat95] and [JM96]), respectively, under SUN/OS on Sparc Stations and on a multi-processor simulation system implemented in Occam on a cluster of transputers (cf. [MK94]). There have been three essentially different prototypical applications for generating technical therapy plans: a flood prevention system, a chemical fibre production installation, and a ballast tank system for off-shore platforms. Last but not least, it is based on the second author's Ph.D. and on the third author's student's project work.

Nevertheless, the present paper is exclusively theoretical in spririt. The reason is that we identified a serious deficiency in the area of communication between planning systems and other knowledge processing modules. Therefore, we decided to do some investigations into the communication between our planning system and our simulation system to qualify the planning process for meeting real-time requirements.

2 KNOWLEDGE-BASED PROCESS SUPERVISION AND CONTROL

Before discussing the communication of certain knowledge processing modules, one needs to know about their particular role within the overall knowledge processing system, about the position in the knowledge processing architecture, and about the available knowledge and its peculiarities. We have to keep this in the present chapter as short as possible.

2.1 System Architecture

In the overall approach to knowledge-based process supervision and control (cf. [BKM92] and [MAM92]), conventional systems are enriched by a knowledge-based component for automatic diagonsis and technical therapy as depicted in the first figure.

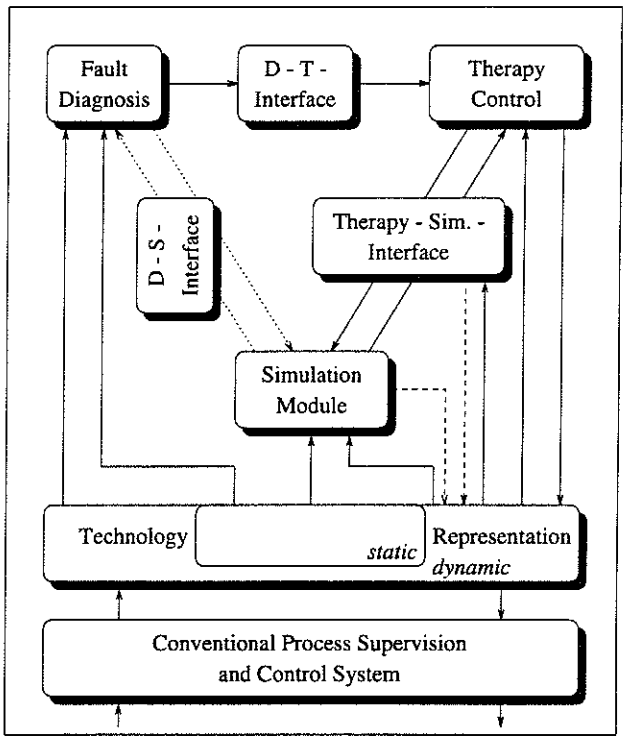


Figure 1: Simplified System Architecture

This architecture is oversimplified, but it suffice to illustrate the main issues to be taken into account within the present paper.

The underlying knowledge base is called *technology reprsentation*. The interesting aspects of adequately representing static technological knowledge are not discussed here (cf. [Arn94b], for a comprehensive treatment). We also do not consider knowledge acqusition, within the present paper (cf. [Zsc95]

for the acquisition of therapy knowledge within our overall project).

When the knowledge-based component is triggered by alarms written into the dynamic part of the technology representation, the ultimate duty of the module for fault diagnosis is to generate a goal specification for the technical therapy control system. This is not only a planning system, but an integrated module

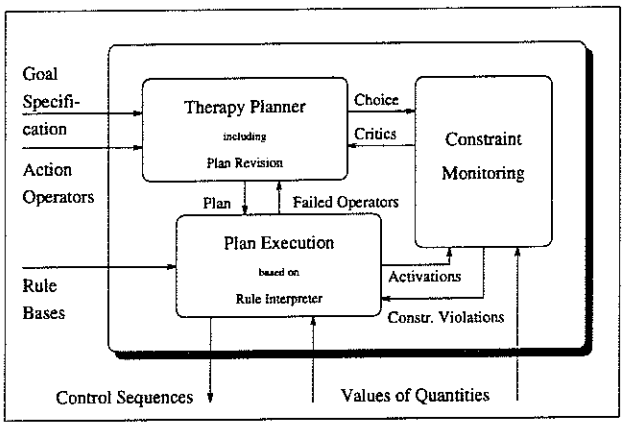


Figure 2: Architecture of Therapy Control

for *technical therapy planning*, for *plan execution*, and for *constraint monitoring* as depicted in figure 2.

2.2 Therapy Plan Generation

Planning basically consists in choosing and assembling actions to be performed in the future. Under complex and dynamically changing conditions, it may be quite difficult to check the validity of constraints necessary for the execution of actions in advance. This is the crux of all planning problems.

We are especially interested in complex dynamic systems like chemical processes, e.g. Our focus is on the generation of technical therapy plans. Therapy planning is only invoked, if the underlying process is seriously disturbed. In such situations, one can not expect to have sufficiently complete knowledge about process parameters (like the percentage of some fluids mixed in some reactor after some valve has been broken, e.g.) and measuring these parameters is usually much to costly and time-consuming. Additionally, mathematical models for unforeseen situations are rarely available.

Thus, we consider technical therapy plan generation as an induction problem (cf. [AJ94a] and [JA96]). Based on incomplete information, plan revision may be frequently necessary. There is an obvious need to narrow the search space of planning. For this, simulation is an important reasoning process.

Formally, technical therapy plans are hierarchically structured graphs which result from stepwise node expansion (cf. [AJ94c], [JA95a], [Kir95], and [JA95b]).

3 LOGICAL BACKGROUND

Both during plan generation and during plan execution there is some need to check process conditions for validity. This inevitably needs a formal language of admissible formulae, a concept of validity and some algorithms for proving or disproving formulae. In other words, we do need some logic.

In [Arn94a], there has been developed and investigated some *modal temporal logic* underlying our knowledge processing approach. The logical problems relevant for the interaction of therapy planning and simulation are discussed in [JLA96] in some more detail. In particular, one needs to develop an appropriate formal language of formulae with emphasis both on representing time and vagueness resp. incompleteness of information. Furthermore, one needs a concept of models to establish validity as well as an implication of the entailment relation. Here, a sketch will do.

3.1 Episodes and Histories

Episodes and histories (cf. [Wil86], for a similar, but quite informal approach) are the basic concepts for representing dynamic process information. An episode is a rectangle spanned by a time interval and a value interval. The least upper bound ω of time points may occur as the right delimiter of some time interval. Histories are finite collections of episodes which do not overlap in time. There is an arithmetic of histories (cf. [Arn94a]).

The first figure illustrates a typical history of one process parameter where some information is miss-

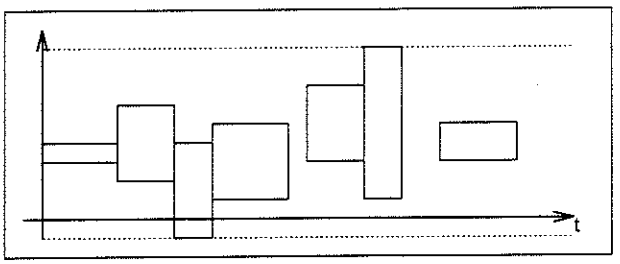


Figure 3: Incomplete History of 7 Episodes

ing. The dotted lines indicate minimal and maximal values known in advance, due to technological conditions and physical laws. The next figure depicts a completion of the preceding history. In a sense, sets

of histories form models for the underlying logic of constraints including guesses of future values.

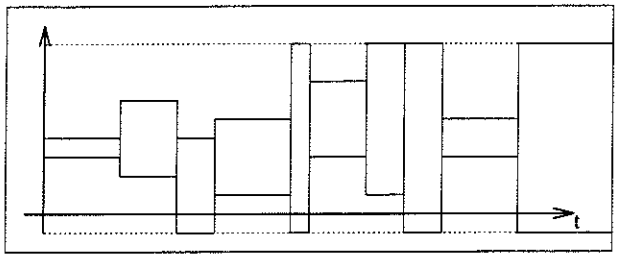


Figure 4: Complete History of 10 Episodes

Based on this model concept, validity of formulae is well-defined (cf. [Arn94a]). The essence is that an atomic ground formula is valid in some interval of time, in case it is valid at any time point or in any atomic interval (in dependence on the underlying model of time, cf. [Sho88]).

As validity necessarily needs the reference to some interval of time, formulae are built as pairs (i, φ) consisting of some time interval i and some constraint φ.

3.2 Refutability of Constraints

Constraints are particular formulae attached to actions. If a planner is trying to plug an action into a therapy plan under construction, it needs to check the corresponding constraints. From the actions chosen before, the planner may estimate a time interval $[t_1, t_2)$ in which the corresponding action may possibly be called. A constraint φ describing some necessary precondition must not be violated at starting time. Thus, the action had to be rejected, in case φ could be refuted within $[t_1, t_2)$ based on the underlying histories of process parameters occurring in φ.

More formally, if $\mathcal{TR}$ is the (dynamic part of the) current technology representation understood as a collection of complete histories as above,

$$\mathcal{TR} \not\models ([t_1, t_2), \neg\varphi)$$

has to be verified for node expansion under the constraint φ.

Simulation is intended to provide the information underlying constraint refutation. Providing the appropriate information is crucial for reducing

- *backtracking* during stepwise node expansion and
- *replanning* in response to the process dynamics during planning or execution.

It is worth to be mentioned that the logic in use is downward-hereditary, but not upward-hereditary (cf. [Sho88]).

4 SCENARIOS OF SIMULATION

First of all, one needs to specify the form of a request sent from the technical therapy planner to the simulation module. It is not sufficient just to ask for the value of some parameter during some future time interval. This insight motivates the investigation of refined scenarios of interaction.

For estimating the value of some process parameter during $[t_1, t_2)$, one usually needs to know about the actions scheduled before. This leads to the first naive scenarios.

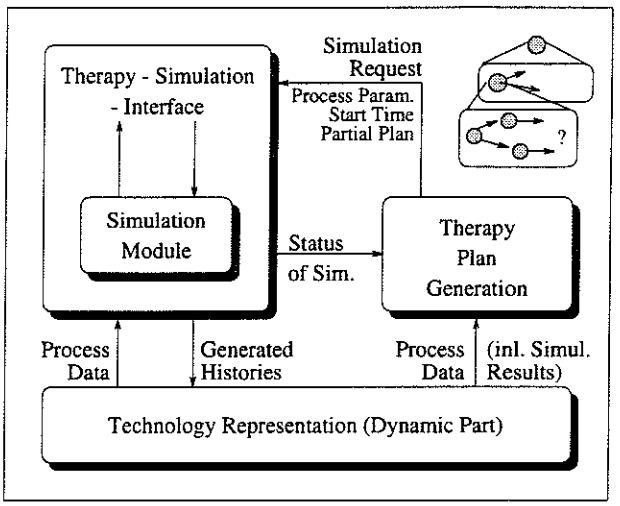


Figure 5: Therapy–Simulation–Interface

From a certain perspective, one may understand the simulation module to be embedded in the therapy-simulation-interface, because of the large amount of necessary control.

4.1 Basic Scenarios

A given planner is generating a plan to be started at some time point t_0 in the future. Iteratively, it is expanding nodes of the plan under construction. If the planner needs some simulation result, it is sending out some request $[t_0, Plan, PP, t_1, t_2]$, where t_0 is the intended start point of the whole plan, *Plan* is the incomplete initial plan preceding the current node under consideration, *PP* is the set of process parameters to be estimated, and $[t_1, t_2)$ is the time interval of interest.

Although this introductory scenario may seem convincing, it is quite naive and may not work sufficiently well, in many cases. To explain only one difficulty, the planner usually does not know $[t_1, t_2)$ sufficiently precise, as the duration of actions in *Plan* may depend on parameter values in the future. Let us consider an

almost trivial example: The necessary time to empty some reactor of a disturbed installation may depend on numerous parameters including the amount of material still flowing into this reactor, by accident. Thus, estimating the time necessary for the one action may require some other estimates before.

Therefore, it seems more appropriate to leave the specification of $[t_1, t_2)$ to the simulation system. This is a first idea how to refine the naive scenario illustrated by figure 5.

4.2 Advanced Scenarios

There is a whole bunch of refined simulation scenarios depicted by the figure below. Arrows illustrate successive refinements, whereas branches denote different concepts which can be reasonably combined.

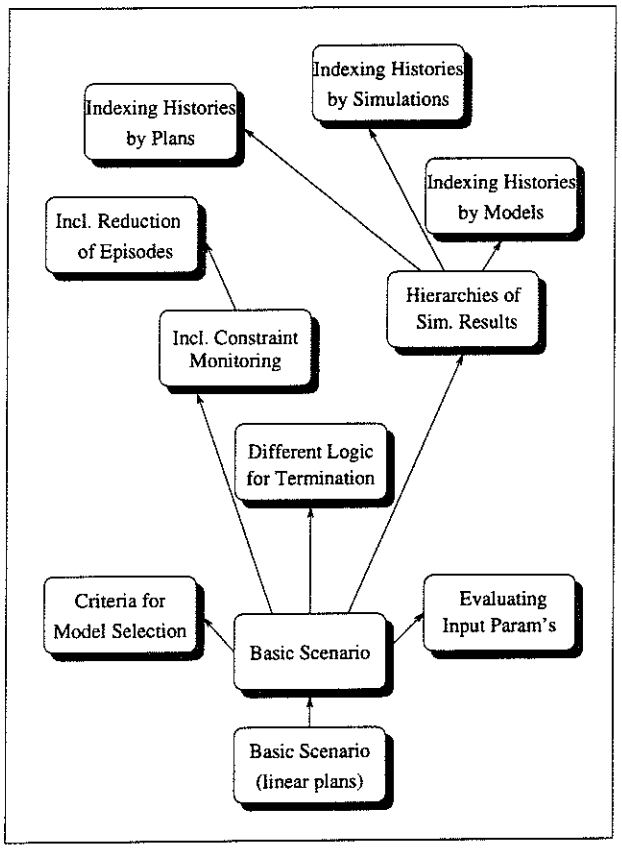


Figure 6: Hierarchy of Simulation Scenarios

We confine ourselves to just one prototypical explanation: It makes a considerable difference whether or not simulation results are checked for respecting constraints. This option is represented by the box "Incl. Constraint Monitoring". Additionally, one may decide to cut down simulation results to only those episodes which do not exceed intervals determined by constraints. In particular, if a simulation run forecasts an interval of process values such that for some of them, the next action could not be executed, one may cut down the resulting episode correspondingly. This (especially, its iteration) may narrow the search space of simulation remarkably. This option is named "Incl. Reduction of Episodes" above.

One refinement may be understood as filtering the simulation results by constraint checking and technological restrictions, e.g.

4.3 Conclusions

Real-time generation of technical therapy plans needs a minimization of *backtracking* and *replanning*. This requires to base plan generation on more accurate knowledge about future process parameters. Therefore, plan generation in complex dynamic environments meeting real-time constraints inevitably needs simulation.

Useful simulation results need to take into account both the dynamics of the process itself and the impact of plan execution.

We have identified a collection of parameters for tuning the communication between therapy planning and simulation. The underlying logic turned out to be a crucial parameter.

5 ACKNOWLEDGEMENT

The comprehensive approach towards knowledge-based process supervision and control has been developed within the joint project WISCON funded by the German Federal Ministry for Research and Technology under grant no. 413–4001–01 IW 204 B. The conference participation of the first author has been supported by the German Research Fund (DFG) under grant 477/378/96.

References

[AJ94a] Oksana Arnold and Klaus P. Jantke. Therapy plan generation as program synthesis. In Setsuo Arikawa and K.P. Jantke, editors, *Algorithmic Learning Theory, Proc. AII'94 and ALT'94, October 10-15, 1994, Reinhardsbrunn Castle, Germany*, volume 872 of *Lecture Notes in Artificial Intelligence*, pages 40–55. Springer–Verlag, 1994.

[AJ94b] Oksana Arnold and Klaus P. Jantke. Therapy plan generation in complex dynamic environments. ICSI Report TR–94–054, International Computer Science Institute, Berkeley, California, October 1994.

[AJ94c] Oksana Arnold and Klaus P. Jantke. Therapy plans as hierarchically structured graphs. In *Fifth International Workshop on Graph Grammars and their Application to Computer Science, Williamsburg, Virginia, USA*, November 1994.

[Arn92] Oksana Arnold. Reaktive Therapieplanung in dynamischen Prozeßumgebungen. WISCON Report 03/92, HTWK Leipzig (FH), Fachbereich IMN, October 1992.

[Arn93] Oksana Arnold. Entwicklungswerkzeug zur Analyse generierter Pläne "PlanTracer". WISCON Report 03/93, HTWK Leipzig (FH), Fachbereich IMN, December 1993.

[Arn94a] Oksana Arnold. A logic of constraints for dynamic process control. WISCON Report 09/94, HTWK Leipzig (FH), Fachbereich IMN, December 1994.

[Arn94b] Oksana Arnold. Towards structure and management of knowledge bases for controlling technological equipments. In Eberhard Köhler, editor, *39. Internationales Wissenschaftliches Kolloquium der TU Ilmenau, Band 3*, pages 30–36. Technische Universität Ilmenau, 1994.

[BKM92] Dietrich Balzer, Volkmar Kirbach, and Volker May. Knowledge based process control. In Hartwig Steusloff and Martin Polke, editors, *Integration of Design, Implementation and Application on Measurement, Automation and Control,* INTERCAMA *Congress 1992*, pages 33–49. Oldenbourg, 1992.

[FN71] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to theorem proving in problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[JA95a] Klaus P. Jantke and Oksana Arnold. Graphgrammatik-Konzepte in der Therapieplanung für komplexe dynamische Prozesse. In Markus Holzer, editor, *4. GI Theorietag "Automaten und Formale Sprachen", 1994, Proc., Herrsching*, pages 42–47. Universität Tübingen, Wilhelm-Schickard-Institut, 1995.

[JA95b] Klaus P. Jantke and Oksana Arnold. Variants of plan generation for complex dynamic systems. In Xin Rao, editor, *Proc. 8th Australian Conference on Artificial Intelligence (AI'95), November 13-17, 1995, Canberra, Australia*, pages 531–538. World Scientific Publ. Co., 1995.

[JA96] Klaus P. Jantke and Oksana Arnold. Inductive program synthesis for therapy plan generation. *New Generation Computing*, 14(4), 1996.

[JLA96] Klaus P. Jantke, Torsten Lehmann, and Oksana Arnold. Meeting real-time constraints in therapy plan generation for complex dynamic systems by flexible simulation scenarios. In *Proc.* ECHTZEIT'96, *Karlsruhe, June 18.-20., 1996.* Franzis-Verlag, 1996.

[JM96] Klaus P. Jantke and Daniel Matuschek. Developing and implementing planning heuristics in Prolog. In *Proc. Florida AI Research Symposium (FLAIRS-96), Key West, FL, USA, May 20.-22., 1996*, 1996.

[Kir95] Daniel Kirsten. Properties of formal languages of therapy plans created by graph grammars. Comm. of the Algorithmic Learning Group CALG–04/95, Hochschule für Technik, Wirtschaft und Kultur Leipzig, FB Informatik, Mathematik und Naturwissenschaften, September 1995.

[MAM92] Volker May, Oksana Arnold, and Uwe Metzner. Wissensverarbeitung in dynamischen Prozeßumgebungen – Eine Anforderungsspezifikation. WISCON Report 01/92, HTWK Leipzig (FH), Fachbereich IMN, March 1992.

[Mat95] Daniel Matuschek. PIG: Therapieplanung mit Mitteln der logischen Programmierung. Comm. of the Algorithmic Learning Group CALG–02/95, HTWK Leipzig, Fachbereich IMN, July 1995.

[MK94] Rolf Müller and Andreas Kroll. Parallele Echtzeitsimulation zur Führung komplexer technischer Prozesse. WISCON Report 03/94, HTWK Leipzig (FH), Fachbereich Elektrotechnik, April 1994.

[Sho88] Yoav Shoham. *Reasoning about Change.* MIT Press, Cambridge, MA, 1988.

[Wil86] B. Williams. Doing time: Putting qualitative reasoning on firmer ground. In 5^{th} *National Conference on Artificial Intelligence, AAAI-86, Philadelphia, Pennsylvania, USA, 1986*, pages 105–112, 1986.

[Zsc95] Christoph Zschiesche. Erstellung und Analyse von Wissensbasen der Prozeßsicherung und -optimierung zur Verbesserung und Erweiterung geeigneter Wissensrepräsentationen. Master's thesis, Leipzig Univ. of Technology, August 1995.

REAL-TIME SATISFICING AGENTS FOR COMPLEX DOMAINS

John Anderson and Mark Evans
Department of Computer Science
University of Manitoba, Winnipeg, Manitoba, Canada R3T 2N2
andersj@cs.umanitoba.ca / evans@cs.umanitoba.ca

Abstract

We describe real-time aspects of *Waffler*, a novel architecture that allows an agent to perform in complex, dynamic environments through improvisation. Improvisation involves using a routine method of accomplishing an activity as a guide to satisficing behaviour, adhering to that method as closely as the current situation permits for economic reasons, and exploring the background knowledge from which the routine has arisen to supplement the routine and move beyond it when necessary. Agents employing this approach can follow a routine in the face of uncertainty and variability, and can apply a routine in a situation with novel aspects, satisficing to the degree that time is available.

1. HUMAN REAL-TIME PERFORMANCE

The goal of Artificial Intelligence is to design and build systems that behave intelligently. The field of planning directly shares this goal: we wish to build systems that can function in complex domains and manipulate the world around them in interesting ways. While systems that reason about what to do have been a focus of research for as long as AI has been in existence, current research in planning stresses real-time aspects of such reasoning, including the ability to function with minimal delays for planning and the ability to react to new situations in the world as they arise dynamically - abilities that are crucial to humans as we go about our own everyday activities.

In order to deal with the constraints of performing in real time, we clearly need to limit the amount of time we spend reasoning about what to do. Taking this limitation to the extreme results in reacting to each new situation in the environment as it arises, typically using some form of pre-compiled stimulus-response network to allow an agent immediate access to actions (e.g. [1, 9] and many others). While such approaches have proven useful in limited domains, many difficulties are associated with them: many domains simply do not have the kind of regularity and structure to support complete compilation, or the ability to make action decisions based solely on local information, and implementing large networks for complex domains is unworkable (these criticisms are extensive and well-known; see [20, 4, 13] for reviews).

There is much more, however, to human real-time processing than reacting as quickly as possible. Producing timely responses is important, certainly, but equally important is our ability to balance the quality of solutions obtained with the effort we put into them - our ability to *satisfice* [21]. We have the ability to react immediately considering only local circumstances when time is of the essence (e.g. jumping out of the way of a moving truck). Equally, we have the ability to use our judgment and deliberate to a greater or lesser degree about the immediate alternatives available to us and those that would be available given greater effort, considering the effects of an immediate decision on our future plans, and considering the effects of novelties in the environment on our routine ways of accomplishing activities. Delaying immediate reactions in order to potentially better our outcome is also real-time performance, no matter how long the delay, so long as we do not assume that the world will stop while we deliberate. We must also recognize that suboptimality goes hand in hand with real-time satisficing: accepting less than optimal solutions is an unavoidable part of being rationally bounded in a complex world [21].

We as humans use many methods to restrict the amount of information we need to consider for any given action decision in order to achieve a balance between the timeliness of our response and the amount of cognitive effort necessary to produce it. We structure the world around us to reduce the number of potential actions that must to be considered at any time [16] and design tools and devices that embody problem-solving knowledge and serve the same purpose [2, 18]. We also structure our problem-solving knowledge into routine ways of performing activities that we rely upon heavily to limit the number of potential actions we must consider.: That is, we rely on the predictable ways in which things have worked before [1, 4, 18]. Portions of any activity may be familiar enough to be compiled into a complete collection of responses usable by a purely reactive system; however, in most of our activities such routines cannot be direct guides to activity simply because they can never be complete or even structured to this degree. This includes any activity with which we are not *completely* familiar, performing an activity in conjunction with others in the short- or long-term, or performing an activity in a different situation than usual [4].

In order to apply a routine effectively and flexibly in the face of greater variability than can be completely anticipated, we possess a vast collection of more general knowledge that allows us to integrate alternatives seamlessly with our routine. We divert from our routine when it makes sense to do so, and return to it without anything like the kind of effort known to be required (e.g. [14]) to alter a stored symbolic plan. We can also use our routine as a weaker guide in conjunction with background knowledge to cope with even greater degrees of variation, in a satisficing manner and in real time . For example, one can shop successfully even when in a hurry and in a strange supermarket; can prepare a meal easily in a friend's kitchen; and can sharpen a pencil without a lot of effort even if no sharpener is available. We commonly call the mode of activity behind such efforts *improvisation*.

Improvisation as creating minor or extensive variations on a routine in real time occurs in the vast majority of human behaviour, from theatre and music to cooking, driving, and architecture [17]. Space prevents a detailed examination of improvisation in human activities here; however, see also [4]. During the course of improvisation, our compiled plan knowledge represents a resource that is relied upon to reduce the intellectual effort that would normally be associated with the activity, in order to perform in a timely manner. We rely on this resource strongly in cases where the current situation follows our previous experience. In situations where our previous experience differs, we can use associated background knowledge out of which the routine has arisen to improvise on our previous experience to the degree that the situation warrants. Beyond coping with difficulties, we can use this background knowledge in creative problem solving, deliberately challenging the boundaries normally placed on our intellectual efforts.

We have developed an approach to real-time satisficing performance for complex domains based on this view of improvisation, that does not assume the degree of knowledge structuring and completeness required by purely reactive systems. Where extensive structuring is available, the approach takes advantage of it, and where it is unavailable the system improvises

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/96 ©1996 FLAIRS

upon the structure that is present. This approach is embodied in an agent architecture known as *Waffler*, and allows an agent to react to its environment immediately (following both local and global goals), as well as to devote as much deliberative activity as time permits and knowledge of the situation deems warranted. This system employs constraint-directed reasoning mechanisms to represent the agent's knowledge of activity, to control the extent to which the agent improvises (and thus the degree of satisficing), and to control agent components (e.g. deliberation) that directly affect the agent's ability to perform in real time.

2. INTENTIONS AND IMPROVISATION

A Waffler agent's compiled plan knowledge and background knowledge are incorporated into distributed constraint-based knowledge structures known as *intentions*. An intention consists partly of a general description of how to perform some particular activity (the agent's compiled plan knowledge). For example, the core of the intention shown in Figure 1 is a collection of steps that involve making tea, and constraints describing the relationship between them. Plan knowledge in intentions includes not only direct recommendations for physical actions, but also cognitive actions (including adopting further intentions, which would be the result of following the compiled knowledge shown in Figure 1) and constraints - strong or weak restrictions, preferences and guidelines for resources and methods of performing actions, as well as more abstract preferences, such as those for methods of controlling computational processes within the agent itself.

An intention also contains links to the background information out of which the agent's compiled routine has evolved. This knowledge (concrete and abstract concepts and knowledge of activity settings) is organized as a highly-interconnected network, and individual concepts may be linked to the intention as a whole (Figure 1 illustrates general knowledge of the kitchen setting linked to this intention), or to specific portions of the intention (individual steps, resources, or constraints). Other background information will be connected indirectly to this intention and others through connections to concepts that are directly connected. Because this network of concepts represents the knowledge from which the agent's compiled plan has been derived, the more likely a particular concept is to be of use when following a particular intention, the closer it will be connected to it. Additional concepts not directly connected to this intention may also supply alternatives for activity, through a gradual process of recollection.

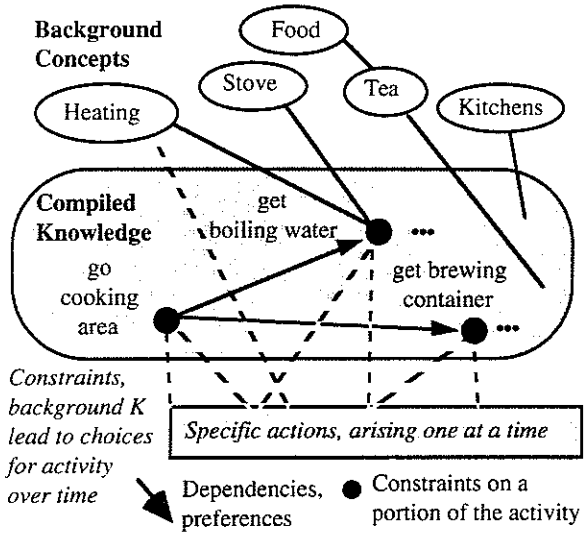


Figure 1. Example of an Intention.

This structuring allows the agent's routine to contribute alternatives for action and knowledge for decision-making immediately, and background information to do the same using a search process whose length will vary depending on how closely associated particular pieces of background knowledge are to the agent's routine. This in turn allows the agent access to immediate responses that are useful in the typical case of the activity, and the ability to search and deliberate as time and the significance of the situation permit. Intentions themselves are also linked hierarchically as they are adopted, allowing more general knowledge of the activities in which the agent is participating to be accessed in the same fashion. The concept of intentions has been used previously, most notably by Bratman et al. [8]; however, intentions in the Waffler architecture are much more sophisticated than those used by Bratman. In the spirit of Boden's [7] use of intentions as both goals and means for achieving activity, a Waffler agent's intentions are *active* guides - they contribute alternatives for action and connections to background knowledge that can indirectly supply alternatives and constraints on behaviour through knowledge of activity, as opposed to influencing activity strictly through a sense of commitment to the intention itself.

The knowledge encompassed by an intention is represented mainly through the use of constraints. Constraints as a representation of strong or weak restrictions have been used previously to represent a broad range of concepts necessary for activity, from physical restrictions and requirements [12] to expectations of actions or other agents and control of agent components such as memory retention and deliberation [11, 4]. In addition to these, constraints within this architecture are used to represent direct preferences for resources, actions, or activities; restrictions on agent focus (to particular tasks, knowledge or particular perceptual information); and to represent agent policies for behaviour and normative responses to particular situations [4, 6]. Constraints operate at multiple levels, from restrictions on objects and individual actions, to constraints associated with the agent's behaviour as a whole. High-level constraints in the latter group aid in selecting one action over another, limit processing at lower levels, and affect how constraints at lower levels are interpreted. Each level provides context to those beneath it, the same way the knowledge associated with one intention provides context for knowledge in intentions adopted in light of it. Constraints may also serve to directly modify the agent's routines (e.g. performing a routine activity in a hurry).

As individual actions are selected (based on the constraints available), further intentions may be adopted. Actions may also be selected based on recommendations (constraints) from relevant background knowledge, as well as other intentions. A series of actions thus emerges over time as a result of the initial adoption of an intention in conjunction with others adopted at the same time and independent events that occur in the environment as the intention unfolds. This is illustrated in the lower portion of Figure 1.

The use of constraints as the primary knowledge representation mechanism directly supports the ability of an agent to perform in real time. For example, in an intention such as the one shown in Figure 1, the core routine may contain among other things a constraint indicating that the agent should prefer working with an electric kettle when boiling water as opposed to some other tool. This constraint expresses a preference that is normally applied in the course of the activity with no exploration as to the reasons behind the preference constraint. When this tool is unavailable or the agent wishes to reason beyond the routine (due to error, knowledge of potential error, or to high-level constraints such as *be-careful* that affect how an agent performs an activity), the agent can make use of further constraints behind that preference (background knowledge) that describe the role and function of the kettle in the overall routine. The agent can then use those constraints as a basis for reasoning about alternative ways of performing the activity, to the degree the agent wishes to devote intellectual effort to do so. For example, constraints about heating water will lead the agent to a set of objects with characteristics suitable for this purpose.

There will clearly be a large number of constraints available in any significant domain. However, the agent's cognitive effort is for the most part not spent on looking for constraint violations, as in most constraint directed reasoning systems. While we are concerned about violations in some cases (e.g. expectations), here most constraints act positively: their presence compels the agent toward or away from specific courses of reasoning or activity, just as the landscape influences the direction of one's travel. The key to real-time performance is the selective processing of constraints in order to make satisficing decisions in the time available.

3. THE WAFFLER ARCHITECTURE

An overview of the Waffler architecture is shown in Figure 2. The agent itself is divided into several computational processes and two major stores of knowledge. The agent's *long-term memory* contains all the possible intention and conceptual knowledge possessed by the agent. The agent's *working memory* is of a very limited size and contains the active portions of the agent's knowledge base (current intentions and a restricted amount of background knowledge). The structures collectively contained in working memory contribute a pool of pending constraints, which determine a particular set of alternatives for action. Current constraints can also cause new items to be recalled from working memory (as can new desires or perceptions) over time, changing the milieu of constraints and thus the alternatives available. Current constraints in working memory will affect the choice of memory management policy used to decide which concepts to remove in order to make room for incoming information, and will also affect the agent's deliberative component, which selects actions to be performed from the alternatives available at that point in time. The number of alternatives to be deliberated upon is directly proportional to the strength of the constraints in the agent's knowledge base, and will be small in cases where the agent is reasonably familiar with its situation (if the agent is unfamiliar, there will be additional possibilities and more reasoning required - the price to be paid for being a rationally bounded agent in a complex world. Deliberation may result in a choice for action, but may also involve waiting for more background information to be considered. Waiting to make a choice may also cause constraints in the agent's working memory to be altered, through additional information acquired by perception or recollection, or through changes in internal state. More detailed information about all of these components may be found in [4, 6].

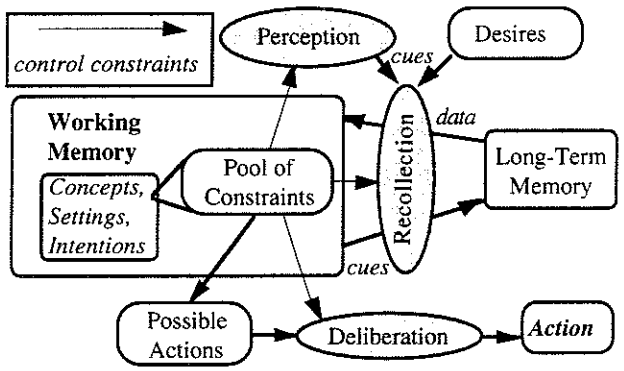


Figure 2. The Waffler Architecture.

The central role in this architecture is played by the agent's working memory, which directly supports the selective recall and processing of constraint knowledge. Working memory is of limited size, and represents the amount of information the agent can process in parallel (in our implementation a timeshared simulator is used, and the size of working memory represents the amount of information the agent can process in a time-slice). Any constraint in working memory is viewed as having immediate effects, and relations among concepts in working memory are assumed to be immediately realizable (via direct memory connections). These are not unrealistic assumptions, since the number of concepts or intentions allowed in working memory at the same time can always be set to a small enough limit to make this so. Items migrate from long-term to working memory through the use of memory triggers. Each object, concept, or intention may have trigger conditions associated with it that, when satisfied, allow this item to be brought into working memory. These conditions may be satisfied by perceptual information, desires, or items already in working memory. Recollection is largely another responsibility of working memory (though external desires can also match concepts and cause their recall), and only trigger conditions currently in working memory are considered. When an item is brought into working memory, the triggers of objects connected to it follow, allowing them to be brought in if conditions permit. This process is inspired by that used by Pauker et al. [19] to deal with the large volume of knowledge in internal medicine, and directly supports the kind of background reasoning necessary in improvisation: beginning with the intention itself and gradually moving to concepts further associated with the intention as necessary.

The Waffler architecture derives most of its abilities through the use of constraints. Not only do constraints represent the structure of an agent's activities, they also control the architecture itself. Constraints are used to define a perceptual focus for the agent, and can also focus the agent toward retrieving particular concepts from long-term memory and thus follow a particular line of reasoning. As will shortly be illustrated, constraints can also directly affect deliberation, limiting the effort put into any particular decision or toward any particular activity.

4. REAL-TIME PROCESSING IN WAFFLER

As mentioned above, the ability of the Waffler architecture to perform in real time is due to its basis in constraint-directed reasoning. When an agent adopts an intention, the information on which it can base immediate decisions is the collection of constraints at the core of its intention and any external concepts that are retrieved automatically. This knowledge supports the agent's ability to immediately react to the situation around itself, and the responses based on this information will as good as that information is complete; that is, as good as a purely reactive approach relying on limited, completely compiled knowledge can be. In the majority of situations, this compiled knowledge will not be complete, and thus the initial set of constraints, while possibly giving the agent a set of alternative actions from which to choose, will represent an incomplete picture of the agent's alternatives. In these situations, and indeed, even if an initial reaction would be somewhat appropriate and the agent wishes to explore further, the agent has the ability to recall additional information over time, changing the milieu of constraints in its working memory and thus the potential responses at its disposal. This is a search process, directed by constraints in working memory and the links associating useful information in the agent's long-term memory.

Given the ability to react or search for a better solution, the agent must have some means of deciding when it is appropriate to act. In many situations, there will be domain knowledge that will allow the agent to judge when it has a good enough solution, or alternatively when it should be suspicious of one or more alternatives. This may be embodied in the agent's intention for the activity itself (this is very likely in activities with which the agent has a great deal of familiarity), or may be represented through external constraints such as *be-careful* that will influence the deliberation mechanism in the agent itself, in a manner to be described shortly. In situations where there are known time limits, the agent can also know when its solution is as good as it will be able to achieve.

Domain knowledge such as that described above works in tandem with a more general means of deliberation control in a Waffler agent, in order to allow the agent to judge the propriety of its solution given its external situation and thereby perform in real time. Individual constraints in working memory influence the agent toward (or away from) some alternative for action through a quantitative or qualitative measure of *utility*. This measure is composed of an innate estimate of the constraint's importance, modified by the importance of the chain of intentions through which the constraint has come into working memory and by specific conditions the constraint is concerned with. Utility is thus situation- and activity-dependent. At any time, a given number of constraints (those that have passed through working memory) will have contributed utility measures to the overall assessment of the agent's choices. In order to make decisions using incomplete information such as this, the agent has in its working memory a constraint on utility, representing the minimum utility necessary for an alternative to be acted upon. This constraint may be global or local to a particular activity the agent is performing, and serves to limit deliberation by allowing the agent to cease examining its knowledge when a good enough solution for the situation is available, or alternatively, to cause the agent to refrain from committing to an action when the alternatives available are deemed less than satisfactory. Utility constraints are not static, and can be affected by intentions and concepts in working memory as well as higher-level knowledge. For example, when conflicting intentions are present, high-level knowledge may alter the agent's utility constraint to be higher in order that the agent have greater confidence that it is

selecting the most appropriate action in these situations. If no action can be performed, the agent may wait for more information to be processed through working memory. This affords the possibility of a higher utility for one or more alternatives (thereby meeting the utility constraint), or for constraints in working memory to alter the agent's utility constraint, allowing less highly-ranked alternatives to be selected. It also presents the possibility that the agent could miss some time-constrained opportunity while deliberating further - this is also part of the agent's knowledge of activity and will be included in estimates of utility. The situation-specific knowledge described earlier fits into this methodology in two ways - in a situation where the agent should be cautious (for example) of particular alternatives, constraints can remove utility from those alternatives; if the agent is to be cautious in general, the utility constraint can be altered to ensure the agent has greater confidence in the utility of its actions before committing to them. Such constraints can also affect the agent's memory management policies, recollection, and perceptual focus.

It is also useful to split up utility into separate, domain-specific factors. Figure 3 illustrates an example wherein utility has been split into measures of effort (the amount of effort associated with the alternative) and compatibility (how compatible it is with the agent's desires). Along with timeliness, we have found these to be the most important factors in the consideration of potential actions in the course of human everyday activities [4]. In this example, the agent needs to add milk to a dish it is cooking, and has realized it has none. Several alternatives arise out of the constraints involved, each of which has particular estimates of effort and utility put forward by constraints in the agent's working memory (e.g. buying milk may be affected by the constraint that a store must be open, which affects the effort of this alternative). None of these choices is suitable for these constraints, and so the only action the agent can perform (unless other intentions bring about additional possibilities having nothing to do with this) is further deliberation through examination of its knowledge of the situation at hand. In addition to examining its own knowledge, the agent may wait for additional perceptual information to alter either these measurements or the constraints restricting the agent's choice of actions. Waiting may also affect the importance of the intention that brought about this activity (or intentions further back in abstraction), which may further alter these constraints.

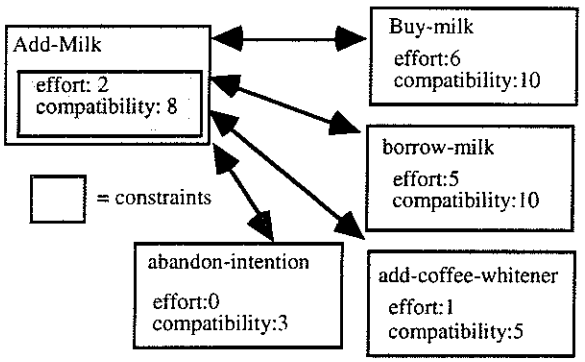


Figure 3. Effort/Compatibility Constraints

5. IMPLEMENTED EXAMPLES

We have implemented this architecture using the Gensim object-oriented simulation testbed [5] under Macintosh Common LISP to provide a simulated world for the agent to inhabit. The implemented environment is a simulated kitchen, in which accidents and other unexpected timely events can occur in the midst of the agent's ongoing activities: tasks such as making tea, answering the telephone, cleaning up, and other activities that normally take place in a kitchen. The agent can make use of tools such as electric kettles, spoons, etc., that are stored in cupboards and drawers. The extent of the knowledge required for these activities necessitates that the implementation of the agent's intentions and background knowledge be incomplete. Portions of intentions and the background knowledge behind them are implemented in detail (e.g. the mechanics and background of acquiring boiling water) as opposed to having more extensive intentions in less detail.

The Waffler agent inhabiting this environment consists of five processes implementing the components discussed earlier. The first of these processes takes object descriptions (attributes and values) provided by the simulator as perception, and integrates these with objects in the agent's world model, effectively performing high-level perception. The second process simulates the parallelism inherent in the agent's working memory: it polls each of the intention instances in working memory for its recommendations and attempts to activate all of the constraints in working memory. The compiled core of an intention can consist of a group of steps with loose or tight ordering constraints, and in this case any step not blocked by constraints is recommended; intentions can also consist of a function returning a reaction given the agent's current perceptions and world model, and in this case the function is called to produce immediate alternatives. In either case, immediate alternatives based on compiled knowledge are available. This process also processes each object in working memory, checking for constraints that may make direct recommendations or indirectly alter working memory.

The third process implements deliberation by selecting the alternative with the highest utility, given the general and task-specific constraints on minimal utility described earlier. It is entirely possible to have no action reach a sufficient utility to be viewed as a suitable action. In such a situation deliberation will ensue over a number of cycles, as described in the previous section. Following deliberation, another process is used to specify a focus for sensory information from the simulator and to commit to the agent's chosen course of action (if any) by communicating both of these items to the simulator. The final process is not part of the overall architecture, and is used to simulate the need for working memory management that would accompany much larger domains. The current size of the domain (approx. 25 classes of physical objects and abstract concepts) does not adequately overload working memory to the extent that a real kitchen would, and rather than accepting the behavioural bias that would come from restricting working memory to a ridiculously small size or simply allowing everything in working memory at once, we instead remove unused components from working memory on each cycle. Items in working memory are timestamped, and if a concept or intention has not been referenced during the current cycle or the previous one, it is "forgotten" by removing the item from working memory

We have used this implementation in several detailed examples illustrating its ability to cope with unexpected events (errors, adapting to missing tools, competing intentions) during the course of an agent's activities in real time in the simulated world. We will consider a simple example here; further examples and a complete transcript of all reasoning performed by the agent may be found in [4]. During the course of making a cup of tea, one of the necessary steps is obtaining boiling water. A partial intention for tea-making was created to focus on this particular portion of the activity. The intention included a preference constraint to use an electric kettle, a link to (among other background concepts) a concept describing the role of the kettle as a container for heating, and knowledge of various other kitchen objects. The intention immediately recommends an alternative of using an electric kettle (i.e. its compiled preference), and if an electric kettle is immediately available in the environment, this is adopted and further activity ensues. If the electric kettle is removed from the environment, several different alternatives can be arranged depending on the extent of knowledge the agent is given. If we want the agent to be very experienced with the activity (i.e. have additional compiled knowledge), other concepts can be brought in at the time this intention is adopted, and other alternatives will be recommended by associated constraints (indeed, these could also be placed directly into the core of the routine, along with making the preference constraint indicating the use of an electric kettle weaker). In a less experienced case, the agent must initiate a search of its background knowledge for acceptable alternatives. It begins to examine the role of the kettle, and when this role concept is recalled, constraints describing the function of the kettle are available. We can arrange for alternative recommendations to be made directly (e.g. give the agent knowledge that a stovetop kettle would be a useful substitute) or give it weaker constraints, indicating the properties of the electric kettle that make it useful for this purpose. In an example run where a pot was placed on a stove in front of the agent, the latter constraints, along with the agent's perception, put the pot forward as an alternative. If the agent is not given a high compatibility constraint, it can make use of this alternative. The agent can also be given a high constraint on compatibility, indicating that it is better to

devote more effort to finding a better solution despite lost time, and it will ignore the alternative and keep looking for a better substitute. If the agent is given knowledge that a stovetop kettle (a suitable substitute) is available somewhere in the domain, it will begin looking for it in places where such an object can be found. If there are no other alternatives (or if some other time constraint is placed on the agent's activities, e.g. from another intention that is more time-limited), the agent will eventually lower its constraints on compatibility and accept the less-suitable but more timely alternative.

As can be seen from examples like this, we can give the agent varying degrees of knowledge and affect its ability to satisfice in a timely manner. We have also implemented more temporally extensive examples, the largest of which involves the agent being interrupted in a mundane task (cleaning up) with a time-constrained opportunity (answering a ringing telephone), dealing with deliberating about multiple goals and resuming blocked goals, perceiving abstract concepts from perceptual information (e.g. inferring a mess from seeing dirty dishes), and dealing with simulated overflow of working memory during the course of reasoning.

This implementation is limited in several ways. As described above, it is limited to partial intentions because of the extent of the common-sense knowledge required to perform an everyday activity such as making tea (and our resources to implement it). More practically, this implementation is limited in that it simulates the parallelism inherent in working memory (though limits it through mechanisms described above), and in the fact that parallel actions are not considered. Cognitive and physical actions are also treated similarly (for greater realism, so that cognitive actions will take up time and affect the agent's response time). However, a lack of parallel actions means that an agent cannot perform a cognitive action (e.g. adopting an intention) and a physical action at the same time.

6. SUMMARY

By using compiled plan knowledge as one resource on which to make moment-by-moment decisions, this approach improves (like other reactive approaches) on earlier work in planning that concentrated on restructuring plans symbolically to adapt them to new situations (e.g. [3]). At a surface level, this approach is most immediately comparable to that of Bratman et al. [8] in its use of intentions and in the ability to derive new options and deliberate about them. In addition to the differences in the concept of intentions described earlier, there are further important differences. This approach provides specific methods for generating and dealing with options as a core of the architecture itself (and has a means of controlling option generation), rather than leaving these and other components as implementation-dependent details. Constraints as a knowledge representation mechanism also allow the agent to reason as deeply or shallowly about a particular situation as time allows (and necessity demands), and support more sophisticated reasoning in terms of applying routines in novel situations. This architecture is also comparable to recent work by Dean et al. [10] on partial universal planning. Dean's approach allows a partial universal plan to be employed by halting and extending the situation-action mapping when an unknown situation develops. This is useful only when the agent can afford to stop and plan (Dean assumes that only minor extensions will be required, and that a deadline for plan completion is known), and does not capture the ability to do the best the agent can given partially-compiled knowledge, general background knowledge, and a limited time to explore that this architecture embodies. Finally, this work may also be compared with the case-based activity work of Hammond et al. [15], which allows goals in a routine to be blocked, suspended, and later recalled. Through the application of constraints, this approach also supports this ability (when constraints do not block an activity, it can resume), but also supports the ability to perform in a satisficing fashion in real time.

While the major contributions of this architecture lie in extending mechanisms for real-time deliberation in complex domains, the development of this approach also makes contributions to extending constraint-directed reasoning and in understanding human cognition in everyday activities. We are currently working on applying improvisation as a mode of interaction in multi-agent scenarios in dynamic, complex simulated environments.

References

1. Agre, Philip, *The Dynamic Structure of Everyday Life*, Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, MIT, 1988. 282 pp.
2. Agre, Philip, and Ian Horswill, "Cultural Support for Improvisation", *AAAI-92*, San Jose, CA, 1992, pp. 363-368.
3. Alterman, Richard, "Adaptive Planning", *Cognitive Science* 12 (393-421), 1988.
4. Anderson, John, *Constraint-Directed Improvisation for Everyday Activities*, Ph.D. Dissertation, Department of Computer Science, University of Manitoba, 1995. 389 pp.
5. Anderson, John, and Mark Evans, "A Generic Simulation System for Intelligent Agent Designs", *Applied Artificial Intelligence* 9:5(527-562), 1995.
6. Anderson, John, and Mark Evans, "Constraints as a Basis for Real-Time Planning", submitted to the *Second International Workshop on Constraint-Based Reasoning*, Key West FL, May, 1996.
7. Boden, Margaret, "The Structure of Intentions", *Journal for the Study of Social Behaviour* 3:1(23-46), 1973.
8. Bratman, Michael, David Israel, and Martha Pollack, *Plans and Resource-Bounded Practical Reasoning*, Technical Note 425R, SRI International, 1988. 28 pp.
9. Chapman, David, *Vision, Instruction, and Action*, Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, MIT, 1990. 244 pp.
10. Dean, Thomas, Leslie Pack Kaelbling, Jak Kirman, and Ann Nicholson, "Planning with Deadlines in Stochastic Domains", *AAAI-93*, Washington, DC, 1993, pp. 574-579.
11. Evans, Mark, John Anderson, and Geoff Crysdale, "Achieving Flexible Autonomy in Multi-Agent Systems using Constraints", *Applied Artificial Intelligence* 6:1(103-126), 1992.
12. Fox, Mark S., *Constraint-Directed Search*, Ph.D. Dissertation, School of Computer Science, Carnegie-Mellon University, 1983. 184 pp.
13. Ginsberg, Matthew, "Universal Planning: An (Almost) Universally Bad Idea", *AI Magazine* 10:4(40-44), 1989.
14. Hammond, Kristian, *Case-Based Planning* (Boston: Academic Pr.), 1989. 277 pp.
15. Hammond, Kristian, Tim Converse, and Charles Martin, "Integrating Planning and Acting in a Case-Based Framework", *AAAI-90*, Boston, 1990, pp. 292-297.
16. Hammond, Kristian, and Tim Converse, "Stabilizing Environments to Facilitate Planning and Activity", *AAAI-91*, Anaheim, 1991, pp. 787-793.
17. Jencks, Charles, and Nathan Silver, *Adhocism: The Case for Improvisation* (New York: Doubleday), 1972. 216 pp.
18. Norman, Donald A., *The Psychology of Everyday Things* (New York: Basic Books), 1988. 257 pp.
19. Pauker, Stephen, G. Anthony Gorry, Jerome Kassirer, and William Schwartz, "Towards the Simulation of Clinical Cognition", *American Journal of Medicine* 60(981-996), 1976.
20. Pollack, Martha E., "The Uses of Plans", *Artificial Intelligence* 57(1):43-68, 1992.
21. Simon, Herbert A., *The Sciences of the Artificial* (Cambridge, MA: MIT Press), 1969. 247 pp.

Semantics of an Efficient Propositional Reasoner : Preliminary Report*

Mukesh Dalal†
Columbia University
Department of Computer Science
New York, NY 10027

Abstract

We present two incomplete and tractable reasoners for clausal propositional logic. We prove that the efficient linear-time boolean constraint propagation is complete for one and refutation complete for the other. We also provide model-theoretic semantics for the two reasoners, in which atoms can be mapped to a real value between 0 and 1. We compare our reasoners with some other previously-known incomplete and tractable reasoners.

1 INTRODUCTION

Clausal Boolean Constraint Propagation (BCP) [McA80] is widely used in several AI systems and applications [McA90, de 90] for linear-time incomplete reasoning with clausal propositional theories. We present two new tractable incomplete reasoners and prove that BCP is complete for one and refutation complete for the other. (For the purposes of this paper, a problem is considered tractable iff the corresponding decision problem is known to be in PTIME [GJ79].) We also present model-theoretic semantics for the two reasoners. The importance of model-theoretic semantics for reasoners has been convincingly argued in several papers (c.f. [Lev84]).

We restrict our attention to disjunctive propositional formulas and theories. Disjunctive formulas slightly generalize clauses by allowing arbitrary grouping and multiple occurrences of literals. We associate each formula with a set of literals called support set. Instead of the classical two-valued interpretations, our model-theory is based on valuations that assign real numbers in the closed interval [0,1] to the atoms. A valuation on atoms is extended to formulas using the support sets. In addition to the notion of models, we have an independent notion of counter-models, leading to two distinct entailments, $\not\approx$ and $\not\approx'$. Any entailment relation between theories and formulas implicitly defines a reasoner. We show that the reasoner $\not\approx'$ is strictly weaker than $\not\approx$, and that a refutational variant of BCP is complete for $\not\approx$. BCP is also refutation complete for $\not\approx'$, that is, BCP can be used to determine whether inconsistency can be inferred using $\not\approx'$. However, BCP is not complete for $\not\approx'$, that is, BCP can not be used in general to determine whether a formula (other than "false") can be inferred from a theory using $\not\approx'$.

We are currently working on extending this logic to arbitrary propositional formulas, specially those with arbitrary nesting of conjunctions and disjunctions, and restricted first-order formulas, with or without probabilities. Many of the notions presented here are defined so that they are amenable to this extension.

The rest of the paper is organized as follows. In Section 2, we present the semantics of the two reasoners. In Section 3, we present BCP for clausal formulas. In Section 4, we use BCP to obtain tractable algorithms for the two reasoners. In Section 5, we compare our tractable reasoners with previously-known reasoners, before concluding in Section 6.

2 SYNTAX AND SEMANTICS

Consider a denumerable set $\mathcal{A} = \{P, Q, \ldots\}$ of propositional symbols, called *atoms*. A *literal* is either an atom or its complement; for example, P and $\neg P$. *Disjunctive formulas* (*formulas*) of our language $\mathcal{L}$ are built finitely from the literals using the binary connective $\vee$ called *disjunction*, and the parenthesis "("

*This work is partially supported by NSF Grant No. IRI-94-10117 and ARPA/ARL Contract No. DAAL01-94-K-0119.

†Email: dalal@cs.columbia.edu. Phone: (212) 939-7114. WWW: http://www.cs.columbia.edu/~dalal/.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/101

and ")" used for grouping subformulas; for example, $P \vee Q$ and $(P \vee \neg Q) \vee Q$. A *theory* is a finite set of formulas; for example, $\{\neg P, P \vee Q, (P \vee \neg Q) \vee Q\}$. Our notion of formulas is a slight generalization of the standard notion of clauses. In particular, the formulas of $\mathcal{L}$ are the formulas of the classical Propositional Calculus (PC) [Men64] in negation normal form, but without the conjunction connective $\wedge$. This allows us to use the standard notions of interpretation, equivalence ($\equiv$), entailment ($\models$), etc. Note that any theory in PC can be converted into an equivalent theory in $\mathcal{L}$.

We normally use variables p, q, etc. to denote atoms; variables α, β, etc. to denote literals $P, \neg P$, etc.; variables ψ, φ, etc. to denote formulas; and variables Γ, Δ, etc. to denote theories.

The set of literals occurring in a formula is called the support set of the formula:

Definition: The *support set*, $S(\psi)$, of any formula ψ is defined inductively as follows:

1. $S(\alpha) = \{\alpha\}$, for any literal α;
2. $S(\psi_1 \vee \psi_2) = S(\psi_1) \cup S(\psi_2)$, for any formulas ψ_1 and ψ_2;
3. $S((\psi)) = S(\psi)$, for any formula ψ.

Since supports are sets, multiple occurrences of literals in a formula are ignored; for example, $S(P \vee P) = S(P) = \{P\}$. Support sets are used for extending valuations on the set of atoms to the set of formulas. Intuitively, a valuation maps atoms to real numbers in the closed interval [0,1]. This mapping naturally extends to negative literals (one minus the valuation of the atom) and formulas (sum of valuations of the literals in the support set):

Definition: A *valuation* is a function $\mathcal{A} \rightarrow [0,1]$. A valuation V is extended to the set of formulas as follows:

1. $V(\neg p) = 1 - V(p)$, for any atom p;
2. $V(\psi) = \Sigma\{V(\alpha) \mid \alpha \in S(\psi)\}$, for any formula ψ.

Although valuations of atoms and literals are real numbers in [0,1], valuations of other formulas may exceed 1. For example, a valuation that maps both P and Q to 1, maps $P \vee Q$ to 2, and maps $P \vee P$ to 1. Although we do not exploit this feature in this paper, higher values may signify higher epistemic entrenchment, a notion useful for belief revision [Gar88].

Definition: Formulas ψ_1 and ψ_2 are *equivalent*, denoted by $\psi_1 \approx \psi_2$, iff $V(\psi_1) = V(\psi_2)$ for each valuation V.

Note that if $S(\psi_1) = S(\psi_2)$ then $\psi_1 \approx \psi_2$. For any formulas ψ, ψ_1, ψ_2, and ψ_3, the following equivalences then follow directly:

1. $\psi_1 \vee \psi_2 \approx \psi_2 \vee \psi_1$, that is, $\vee$ is commutative;
2. $(\psi_1 \vee \psi_2) \vee \psi_3 \approx \psi_1 \vee (\psi_2 \vee \psi_3)$, that is, $\vee$ is associative;
3. $\psi \approx (\psi)$, that is, unnecessary parenthesis are redundant;
4. $\psi \vee \psi \approx \psi$, that is, duplicates may be removed without changing the logical content.

Note that all the above equivalences also hold in PC.

Definition: A valuation V is a *model* of a formula ψ iff $V(\psi) \geq 1$. A valuation V is a *counter-model* of a formula ψ iff $V(\psi) = 0$.

In PC, each formula is either *true* or *false* in an interpretation. In our valuations, a value 0 corresponds to *false* and a value of at least 1 corresponds to *true*. Given a formula ψ and a valuation V, it is possible that V is neither a model nor a counter-model of ψ. For example, a valuation that maps atom P to 0.2 is neither a model nor a counter-model of P. The notions of models and counter-models lead to two obvious notions of entailments:

Definition: For a theory Γ and a formula ψ:

1. $\Gamma \mid\!\approx \psi$ iff no model of Γ is a counter-model of ψ;
2. $\Gamma \mid\!\approx' \psi$ iff each model of Γ is a model of ψ.

Since no model of ψ is a counter-model of ψ, it follows that $\mid\!\approx'$ is no stronger than $\mid\!\approx$. In other words, if $\Gamma \mid\!\approx' \psi$ then $\Gamma \mid\!\approx \psi$, for any theory Γ and any formula ψ. Since each interpretation of PC corresponds to a valuation, both $\mid\!\approx$ and $\mid\!\approx'$ are sound with respect to $\models$.

Consider any model V of the theory $\Gamma_1 = \{P, \neg P \vee Q\}$. Since $P \in \Gamma_1$ and P is an atom, $V(P) = 1$ and $V(\neg P) = 0$. Since $\neg P \vee Q \in \Gamma_1$, it then follows that $V(Q) = 1$. Thus, the only model of Γ_1 maps both P and Q to 1. Since $\Gamma_1 \mid\!\approx' Q$, our logic allows some form of chaining.

Consider the theory $\Gamma_2 = \{P \vee Q, \neg P \vee Q\}$. Since the valuation that maps both P and Q to 0.5 is a

model of Γ_2, it follows that $\Gamma_2 \not\approx' Q$. Since no model of Γ_2 maps Q to 0 (otherwise P should be mapped to both 0 and 1), it follows that $\Gamma_2 \not\approx Q$. Thus, $\not\approx$ is strictly stronger than $\not\approx'$.

Consider the theory $\Gamma_3 = \{P \vee Q \vee R, \neg P \vee Q \vee R, P \vee \neg Q \vee R, \neg P \vee \neg Q \vee R\}$. Since the valuation that maps both P and Q to 0.5 and R to 0 is a model of Γ_3, it follows that $\Gamma_3 \not\models R$. Thus, $\models$ is strictly stronger than $\not\approx$, since $\Gamma_3 \models R$.

3 CLAUSAL BCP

We review clausal BCP in this section. First, we define clauses by generalizing the disjunction connective by allowing arbitrary number of arguments (c.f. [Fit90]).

Definition: A *clause* ψ is a finite disjunction $\alpha_1 \vee \ldots \vee \alpha_n$ of literals, where $n \geq 0$. The empty clause, for $n = 0$, is denoted by $\mathbf{f}$. A *clausal theory* is a finite set of clauses. For a clause $\psi = \alpha_1 \vee \ldots \vee \alpha_n$, the *complement* $\sim(\psi)$ is defined to be the theory $\{\sim\alpha_1, \ldots, \sim\alpha_n\}$, where $\sim p = \neg p$ and $\sim\neg p = p$ for any atom p.

Note that each literal is a unit clause, i.e., with one literal. The notion of set of support is easily extended to clauses:

$$S(\alpha_1 \vee \ldots \vee \alpha_n) = S(\alpha_1) \cup \ldots \cup S(\alpha_n).$$

This also extends the notion of valuations to clauses. Since $S(\mathbf{f}) = \emptyset$, it follows that all valuations are counter-models of $\mathbf{f}$. It also follows that re-ordering the literals in a clause produces an equivalent clause; for example, $P \vee \neg Q \approx \neg Q \vee P$.

BCP is a hyper-resolution variant of unit resolution [CL73], which is also called *unit-resulting* resolution. Given any clausal theory Γ, BCP monotonically expands it by adding literals as follows: in each step, if any single clause in Γ and all the literals in Γ taken together logically entail any other literal (or $\mathbf{f}$), then this literal (or $\mathbf{f}$) is added to the theory Γ. This step is repeated until $\mathbf{f}$ is obtained or no new formula can be added to the theory.

For example, starting with the theory $\{\neg P, P \vee \neg Q, P \vee Q \vee \neg R, P \vee Q \vee R\}$, BCP first obtains $\neg Q$ from $\neg P$ and $P \vee \neg Q$, then $\neg R$ from $\neg P$ and $\neg Q$ and $P \vee Q \vee \neg R$, and finally $\mathbf{f}$ from $\neg P$ and $\neg Q$ and $\neg R$ and $P \vee Q \vee R$. Note that there are other ways of obtaining $\mathbf{f}$ from this theory using BCP. In contrast, unit resolution must obtain some additional intermediate clauses, like $Q \vee R$.

BCP is sound, that is, any literal added by BCP to a theory is logically entailed by the theory. BCP is incomplete, for example, it will not obtain $\mathbf{f}$ from the unsatisfiable theory $\{P \vee Q, \neg P \vee Q, P \vee \neg Q, \neg P \vee \neg Q\}$. However, it is known to be complete for several special classes of theories, for example, Horn theories.

Using ideas similar to that in [DG84], McAllester [McA90] presented an efficient algorithm for BCP, which runs in linear time if all the atoms of the input theory are known in advance. The algorithm maintains the count of literals in each clause and pointers to clauses containing the literal from each literal. It is quite efficient in practice.

4 TRACTABLE ALGORITHMS

In this section we present a complete algorithm for $\not\approx$ and a refutation complete algorithm for $\not\approx'$. Both of these algorithms are variants of BCP. First, we define translation of formulas to clauses.

Definition: The *clausal form* $\widehat{\psi}$ of a formula ψ is obtained by removing all parentheses from ψ. The clausal form $\widehat{\Gamma}$ of a theory Γ is the set $\{\widehat{\psi} \mid \psi \in \Gamma\}$.

For example, if the theory $\Gamma = \{\neg P, P \vee Q, (P \vee \neg Q) \vee Q\}$ then its clausal form is $\widehat{\Gamma} = \{\neg P, P \vee Q, P \vee Q \vee Q\}$.

It immediately follows that $\psi \approx \widehat{\psi}$ for any formula ψ. Theorem 1 shows that BCP is sound and refutation complete for $\not\approx'$:

Theorem 1 *For any theory* Γ, $\Gamma \not\approx' \mathbf{f}$ *iff BCP obtains* $\mathbf{f}$ *from the theory* $\widehat{\Gamma}$.

Proof: Consider any model v of clauses α and $\neg\alpha \vee \alpha_1 \vee \ldots \vee \alpha_n$, where $n \geq 0$ and α's are literals. Thus, $v(\alpha) = 1$ and $v(\neg\alpha) + v(\alpha_1) + \cdots + v(\alpha_n) \geq 1$. Thus, $v(\alpha_1) + \cdots + v(\alpha_n) \geq 1$. Thus, v is a model of $\alpha_1 \vee \ldots \vee \alpha_n$. By repeating this argument, it follows that if BCP obtains a clause ψ from a theory $\widehat{\Gamma}$, then $\Gamma \not\approx' \psi$. The "if" case of the theorem follows when $\psi = \mathbf{f}$.

It also follows from the above argument that if BCP obtains a theory Δ from a theory Γ, then any valuation v is a model of Δ iff it is a model of Γ. This observation helps us to prove the "only-if" case.Let $\Gamma' \neq \{\mathbf{f}\}$ be the final theory obtained from $\widehat{\Gamma}$ using BCP. Consider the valuation v that assigns 1 to all unit clauses in Γ' and 0.5 to all other atoms. Since each non-clause in Γ' contains at least two literals that are assigned 0.5 by v, it follows that v is a model of Γ', and thus of $\widehat{\Gamma}$ and Γ. Thus, $\Gamma \not\approx' \mathbf{f}$. ∎

We now show that BCP is not complete for $\not\approx'$. Consider the theory $\Gamma_4 = \{P_1 \vee Q, P_1 \vee \neg Q, P_2 \vee R, P_2 \vee \neg R\}$. It can be easily verified that $V(P_1) \geq 0.5 \leq$

$V(P_2)$ for any model V of Γ_4. Thus, $\Gamma_4 \not\approx' P_1 \vee P_2$, although $P_1 \vee P_2$ can not be obtained from Γ_4 using BCP.

Theorem 2 shows that refutation variant of BCP is sound and complete for $\not\approx$:

Theorem 2 *For any theory* Γ *and any formula* ψ, $\Gamma \not\approx \psi$ *iff BCP obtains* $\mathbf{f}$ *from the theory* $\widehat{\Gamma} \cup \sim(\psi)$.

Proof: $\Gamma \not\approx \psi$ iff there is no model of Γ which is a counter-model of ψ iff $\Gamma \cup \sim(\psi)$ has no model iff $\widehat{\Gamma} \cup \sim(\psi) \not\approx' \mathbf{f}$ iff BCP obtains $\mathbf{f}$ from the theory $\widehat{\Gamma} \cup \sim(\psi)$. The last step follows from Theorem 1. ∎

In other words, to determine whether $\Gamma \not\approx \psi$, add complements of all literals occurring in formula ψ to the clausal form of theory Γ and determine whether BCP obtains $\mathbf{f}$.

Since clausal forms of theories and formulas are easily obtained by a single scan of the input, BCP provides linear time algorithms for testing whether $\Gamma \not\approx' \mathbf{f}$ and $\Gamma \not\approx \psi$ for any theory Γ and any formula ψ.

5 COMPARISON

Several incomplete but tractable propositional logics have been presented in the literature.

Based on the earlier work of Belnap [Bel77] on relevance logic and Levesque [Lev84] on a a logic of implicit and explicit beliefs, Frisch [Fri87] presented a 3-valued model-theory for PC, whose entailment relation $\models_{RP}$ is the strongest propositional consequence relation that is sound but allows no chaining. Although intractable in general, the algorithm presented for $\models_{RP}$ takes quadratic-time in the worst-case for clausal theories. However, our reasoner $\not\approx$ is in linear-time and is strictly stronger than $\models_{RP}$. For example, $\{(P), (\neg P \vee Q)\} \not\models_{RP} Q$ and $\{(P), (\neg P \vee Q)\} \not\approx Q$.

After transforming the propositional satisfiability problem to integer programming problem, Blair, Jeroslow, and Lowe [BJL86] showed that clausal BCP is identical to the relaxed liner programming problem. Although their approach is similar in spirit to ours, they do not consider arbitrary grouping and multiple occurrences of literals.

Cadoli and Schaerf [SC95] obtained several reasoners by parameterizing $\models_{RP}$ by sets of propositions. However, we have shown [Dal95] that none of their incomplete reasoners is stronger than $\not\approx$.

In a related work [Dal96], we have extended $\not\approx$ to an anytime family $\vdash_0, \vdash_1, \ldots$ of reasoners such that each $\vdash_i$ is tractable, each $\vdash_{i+1}$ is at least as complete as $\vdash_i$, and each theory has a $\vdash_i$ complete for reasoning with it. By using term rewriting systems, we have also extended these reasoners to non-clausal formulas and theories [Dal92a, Dal92b].

6 CONCLUSIONS

We defined two notions of sound and incomplete entailment for restricted propositional theories using a new model-theoretic semantics, and proved that BCP is complete for one and refutation complete for the other. This is only one step of our general research in defining new logics and algorithms for efficient reasoning in knowledge representation systems. Our future work includes extending our semantics to arbitrary propositional formulas, namely, those with nested $\wedge$ and $\vee$, and formulas in predicate logic.

References

[Bel77] N. D. Belnap. A useful four-valued logic. In G. Epstein and J. M. Dunn, editors, *Modern Uses of Multiple-Valued Logics.* Reidel, 1977.

[BJL86] C. E. Blair, R. G. Jeroslow, and J. K. Lowe. Some results and experiments in programming techniques for propositional logc. *Comput. and Operations Research,* 13(5):633–645, 1986.

[CL73] C. Chang and R.C. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, London, 1973.

[Dal92a] M. Dalal. Efficient propositional constraint propagation. In *Proceedings Tenth National Conference on Artificial Intelligence (AAAI-92),* pages 409–414, San Jose, California, 1992. American Association for Artificial Intelligence.

[Dal92b] M. Dalal. Tractable deduction in knowledge representation systems. In B. Nebel, C. Rich, and W. Swartout, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92),* pages 393–402, Cambridge, Massachusetts, 1992. Morgan Kaufmann Publishers.

[Dal95] M. Dalal. Tractable reasoning in knowledge representation systems. Technical Report CUCS-017-95, Department of Computer Science, Columbia University, New York, NY, July 1995. file://ftp.cs.columbia.edu/reports/reports-1995/cucs-017-95.ps.gz.

[Dal96] M. Dalal. Anytime families of tractable propositional reasoners. In *Fourth International Symposium on Artificial Intelligence and Mathematics (AI/MATH-96)*, pages 42–45, Florida, 1996.

[de 90] J. de Kleer. Exploiting locality in a TMS. In *Proceedings Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 264–271, 1990.

[DG84] W.F. Dowling and J.H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 1(3):267–284, 1984.

[Fit90] M. Fitting. *First-order logic and automated theorem proving.* Texts and monographs in computer science. Springer-Verlag, 1990.

[Fri87] A.M. Frisch. Inference without chaining. In *Proceedings Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 515–519, 1987.

[Gar88] P. Gardenfors. *Knowledge in flux : modeling the dynamics of epistemic states.* MIT Press, Cambridge, Mass., 1988.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, W. H., NY, 1979.

[Lev84] H.J. Levesque. A logic of implicit and explicit belief. *Proceedings National Conference on Artificial Intelligence (AAAI-84)*, pages 198–202, 1984.

[McA80] D. McAllester. An outlook on truth maintenance. Memo 551, MIT AI Lab, August 1980.

[McA90] D. McAllester. Truth maintenance. In *Proceedings Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 1109–1116, 1990.

[Men64] E. Mendelson. *Introduction to Mathematical Logic.* Van Nostrand, Princeton, N.J., 1964.

[SC95] M. Schaerf and M. Cadoli. Tractable reasonong via approximation. *Artificial Intelligence*, 74(2):249–310, 1995.

Abductive Inference of Events: Diagnosing Cardiac Arrhythmias

Margaret Guertin
Boston University
guertin@cs.bu.edu

Abstract

Abductive inference of events is the inference of plausible causes of observed events and when they occurred — an important type of reasoning in such diverse areas as archaeology, criminal investigation, and medicine. The major difficulty with abductive inference is the exponential number of explanations that exist for even the simplest sets of observations. Josephson has shown abductive inference for most real-world problems to be NP-hard.[6]

We introduce a new approach to abductive inference: the use of temporal constraint propagation to achieve early refutation of inconsistent hypotheses. HOLMES is an object-oriented abductive reasoning system with a method for propagating quantitative temporal constraints.[4] It is currently implemented in the heart domain, where it generates explanations of cardiac arrhythmias (irregular heart beats) from an electrocardiogram (ECG). HOLMES demonstrates how constraint propagation techniques can lead to very efficient generation of detailed explanations. It also introduces a new approach to expert system diagnosis, showing that immense amounts of domain-specific knowledge are not always necessary, and that sometimes a loosely-coupled network of not-very-knowledgeable agents with a good, quantitative model of causality may be enough.

1 Introduction

The central problem of abductive inference, "the process that generates explanations" [3], is the exponential number of explanations it generates. Nevertheless, many programs have been written to automate abductive reasoning in a variety of areas such as language understanding, electronic circuitry, and medical diagnosis. Their strategies to combat the combinatorics can all be characterized as focusing on the "best" hypotheses. They require heuristics both for measuring the quality of each hypothesis, and for then deciding which hypotheses are of sufficiently high quality to be considered.

Abductive inference has been most thoroughly explored in domains of medical diagnosis, where a second major problem became recognized in the early 1980's — the inability of diagnostic programs to reason about temporal information. As work has progressed in this area, researchers have developed a number of general purpose temporal representation schemes, logics, and utilities, beginning with Allen's interval-based temporal logic, interval representation scheme, and algorithm for temporal reasoning based on constraint propagation[1].

Specific applications of temporal reasoning are few and recent, but address a number of important diagnostic questions that cannot be handled by atemporal diagnostic systems. The TOPAZ system monitors chemotherapy by analyzing time series of white blood cell counts, flagging existing problems, and warning of potential complications.[7] Russ's Temporal Control Structure analyzes temporal data over a course of treatment, and can reconsider and revise an earlier diagnosis in the light of new information.[12]. The TrenDx program makes a significant contribution to diagnosis, identifying clinically significant trends in patterns of temporal growth data in pediatric patients.[5] An important expert system for diagnosing heart disease (HDP) has been significantly improved with an enhanced rule-base that allows it to consider temporal as well as atemporal data in its diagnoses.[9]

Temporal reasoning plays an especially important role in the diagnosis of heart arrhythmias through ECG analysis. The KARDIO program uses machine learning techniques to learn the arrhythmias associated with particular ECG patterns, and then diagnoses new ECGs by attempting to match them to its learned patterns.[2] Widman and Tong's EINTHOVEN program goes a significant step further — not merely diagnosing a particular arrhythmia indicated by an ECG, but actually showing the path (or possible paths) the beat could have taken through the heart to have produced the ECG in a two-dimensional diagram called a *laddergram*.[14] The EINTHOVEN system uses an extensive knowledge base of electrophysiological information, and diagnoses an impressive variety of arrhythmias with an efficient hypothesize-and-test search

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/106 ©1996 FLAIRS

strategy.

The HOLMES system achieves a greater level of efficiency and detail by combining abductive inference with Kohane's quantitative temporal constraint propagation techniques.[8] Like EINTHOVEN, it diagnoses arrhythmias in the form of laddergrams, but with several important differences.

- HOLMES achieves a significant level of efficiency through the propagation of temporal constraints, allowing it to refute most inconsistent hypotheses early. Additional search efficiency comes from using a hypothesize-and-test search strategy, like EINTHOVEN's.

- HOLMES achieves a greater level of temporal detail than KARDIO or EINTHOVEN with its ability to estimate the timing of heart events that are invisible on the ECG. This increased detail allows HOLMES to represent its explanations in laddergrams of *three* dimensions, instead of the usual two.

- HOLMES requires a very small amount of domain knowledge to generate its ECG explanations. Instead of a large knowledge base of expertise, it has a good quantitative model for estimating the timing of causes and effects. This model, used with constraint propagation and little other expertise, allows the system to generate detailed explanations very efficiently. (At present, however, HOLMES diagnoses fewer arrhythmias than KARDIO or EINTHOVEN, due to its smaller knowledge base.)

2 Electrical conduction in the heart

The information on heart physiology and electrocardiography below comes primarily from three sources [10, 11, 13]. The heart can by thought of as a fist-sized electric pump for moving blood through the body. When all is going well, the sino-atrial node (SA node) at the top generates a steady electrical pulse which is conducted down through the other parts of the heart. This current first activates the upper chambers of the heart (the atria), which contract and squeeze the blood into the lower chambers (the left and right ventricles). Very soon after the SA node activates the atria, it activates the atrioventricular node (AV node) and the "Bundle of His", and the current travels through them to the left and right bundle branches. From the bundle branches it travels into the left and right ventricles and the septum between them, normally activating all three simultaneously. As the current travels through the ventricles, they contract powerfully, forcing the blood just received from the atria on out into the body.

When any part of the heart is activated, it begins to *depolarize*, which has the effect of spreading the current throughout, and soon to adjacent parts. Once the process of depolarization is complete, the longer process of *repolarization* begins. This can be thought of as a recovery process, during which that part of the heart cannot be activated by any impulse. Once the repolarization process is complete, the part is in its initial resting state, ready to receive and carry any impulse from an adjacent part.

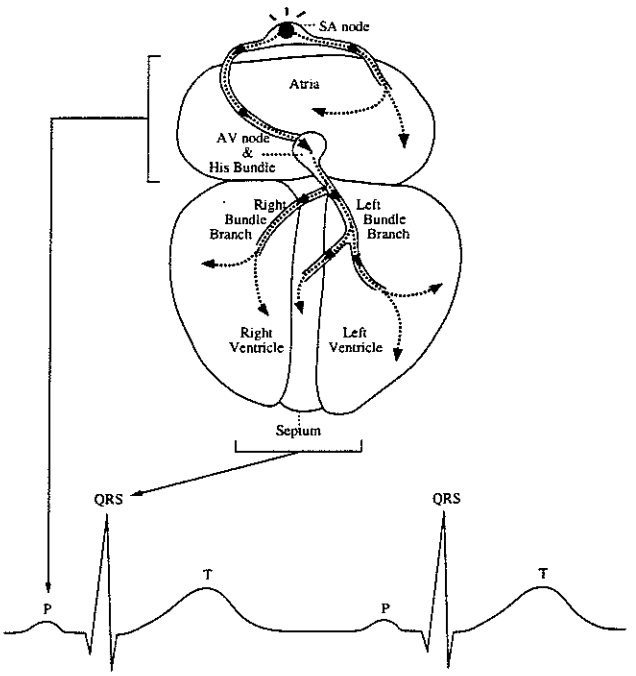


Figure 1: Heart conduction and the ECG

The *electrocardiogram* is a trace of the heart's electrical activity as measured from different points on the surface of the body. It provides incomplete evidence of the impulse paths through the heart, because some parts of the heart produce a current strong enough to be measured from the outside, while others do not. The atria depolarizing give rise to the P wave, the ventricles depolarizing produce the QRS complex, and the T wave is the result of the ventricles repolarizing. All other events are invisible on the ECG, and must be inferred by HOLMES.

In the HOLMES heart model, an *event* is defined to be the activation of one part of the heart, either by an adjacent part, or occasionally, by itself. (Although it is possible for any part of the heart to activate itself by producing its own electrical impulse spontaneously, this normally happens only in the SA node.) The representation of an event in the model includes temporal interval parameters for the start of its depolarization, duration of depolarization, start of repolarization, duration of repolarization, and start of its rest state. A *hypothesis* is a list of assumed events, each of which could account for one bump on the ECG. An *explanation* is a complete causal network of events which could have resulted in the ECG. It includes all events visible on the ECG, as well as predicted events, whose effects would be too small to show up.

3 How Holmes generates explanations

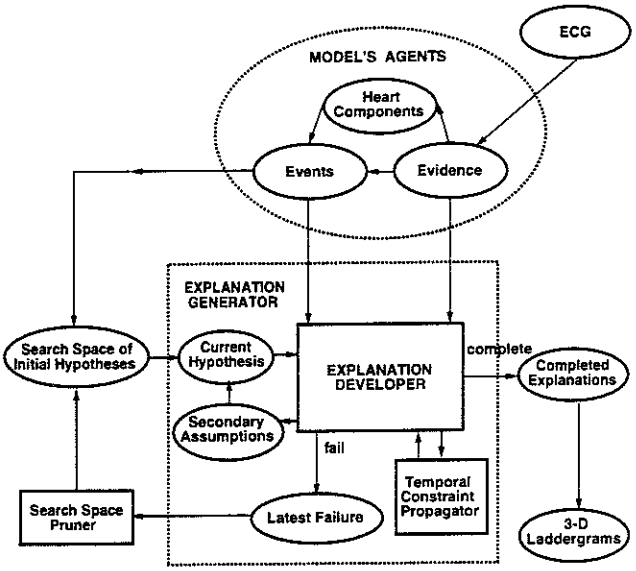


Figure 2: Architecture of the HOLMES system

HOLMES' knowledge of heart conduction is embodied in the model's agents. The *evidence agent* extracts relevant data from the ECG, hypothesizes different events that could have caused a particular ECG wave, and constrains predicted events to be consistent with the ECG. For each component of the heart (SA, AT, AV, LBB, RBB, LV, RV, and Sep), there is an agent containing structural knowledge of that part — what parts are connected to it and where, and having the ability to determine the duration of an activation of that part, based on the time it started and the ECG evidence. For each event, there is an agent which stores information about the activation of that particular part of the heart — the start and duration of depolarization and repolarization, its cause (the event that triggered it), and its effects (the events *it* triggered). Each event agent also has the ability to determine whether or not a particular event would be masked by a recent previous activation, to determine when each of its effects would start, and to hypothesize its possible causes.

HOLMES' explanation generation process begins with the formation of the initial hypotheses that make up its search space. An *initial hypothesis* is an ordered list of isolated assumptions — each one being a possible heart event that could have produced one of the ECG waves. These isolated assumptions are all generated by the evidence agent. Each hypothesis in the space is then fed into the *explanation generator*, which passes it on to the *explanation developer* where an attempt is made to "flesh out" or develop the isolated events of the hypothesis into at least one complete, consistent explanation — one or more causal chains of events accounting for everything on the ECG. The algorithms follow:

Generate Explanations

FOR each hypothesis in the search space:

1. Attempt to **Develop** consistent explanations that include each of its assumptions, adding secondary assumptions when necessary.
2. **IF** at least one attempt is successful, save the completed explanation(s)
3. **ELSE**
   (a) Note the latest assumption precipitating failure
   (b) Prune all hypotheses from the search space that start with the same set of assumptions that lead to this failure

Develop

1. Start with the initial hypothesis — an ordered list of isolated assumptions
2. **FOR** each event on the list:
   - **IF** it is consistent with the ECG **AND** it is consistent with all previous events in the explanation
   - **THEN**
     (a) Link it into the explanation
     (b) Propagate its constraints back to previous events
     (c) Predict all of its possible effects and insert them into the list
   - ELSE
     (a) Note the event precipitating the inconsistency
     (b) **RETURN** FAIL
3. **RETURN** SUCCESS

4 An example

We now illustrate the HOLMES system with a short example showing how it generates explanations for an abnormal ECG, from input to output.

A small amount of essential information is transcribed by hand from the ECG to an input file. For each visible ECG wave, there is a line in the file with the wave type, an indication of whether or not it is normal, and its start and stop times. (Time is expressed in milliseconds from the beginning of the ECG.) The first 4 beats in the ECG used in this example are indicated in a file as follows:

P	normal	100	200
QRS	normal	240	310
T	normal	310	580
P	normal	480	580
QRS	normal	610	670
T	normal	670	910
P	normal	1210	1320
QRS	normal	1350	1420
T	normal	1420	1670
QRS	abnormal	1710	1860
⋮			

From this file, the model's evidence agent generates a list of primary assumptions (possible causes) for each visible depolarization wave. For example, one possible cause of the P-wave at 100 could be A1, the activation of the atria (AT) by the SA node (SA). Another possible cause could be A2, a spontaneous pulse from the atria. Figure 3 shows the first four beats of the ECG with the set of primary assumptions associated with each visible wave of depolarization.

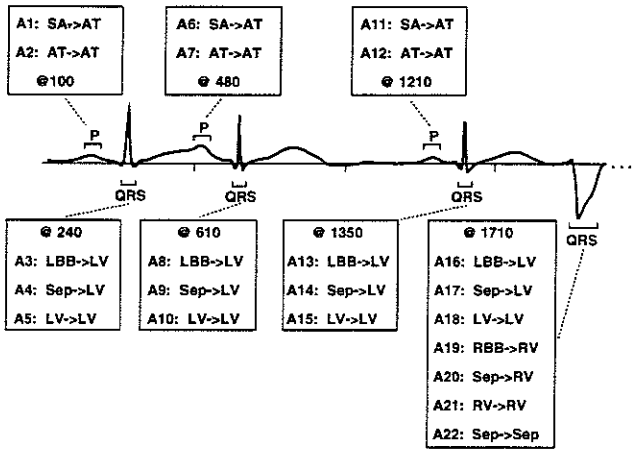


Figure 3: Primary assumptions associated with ECG waves

Since the space of initial hypotheses is searched in depth-first order, the first initial hypothesis is:

$$A1, A3, A6, A8, A11, A13, A16, \ldots$$

Immediately HOLMES discovers that A1 is an *unjustified* assumption, since it has no cause in the (so far empty) explanation. That is, the explanation has no activation of the SA node, which would account for the SA node activating the atria at time 100. Attempting to justify this assumption, HOLMES generates two possible secondary assumptions, either of which could have caused A1, directly or indirectly:

S1: SA → SA @ [91, 98]
This is the spontaneous activation of the SA node starting at some time between 91 and 98, which could have caused A1, the activation of the atria by the SA node at 100. (Since an activation of the SA node is never visible on an ECG, the occurrence of S1 would not be inconsistent with the evidence.)

S2: AV → AV @ [38, 83]
This is the spontaneous activation of the AV-His (AV), which could also have been the indirect cause of event A1.

HOLMES adds S1 to the initial hypothesis, and tries again to develop it into a complete explanation, this time getting as far as the explanation represented in Figure 4.

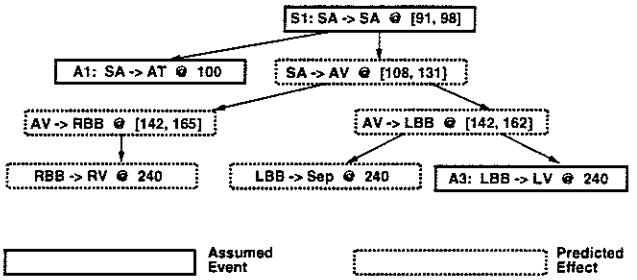


Figure 4: Partial explanation incorporating assumptions S1, A1, and A3

This partial explanation shows a spontaneous activation of the SA node activating the atria at time 100 (accounting for the first P-wave) and also activating the AV-His at [108, 131]. The activated AV-His would then activate the left and right bundle branches (LBB and RBB), with the RBB activating the right ventricle (RV) and the LBB activating the septum (Sep) and the left ventricle (LV). These simultaneous ventricle activations would account for the QRS wave at time 240.

Here again, the process stops upon meeting an unjustified assumption, A6: SA → AT @ 480. The explanation does not indicate a pulse from the SA node at a time that would cause it to activate atria at time 480. Several secondary assumptions are generated that might justify this assumption, but all fail. The first, for example, fails because it leads to an external inconsistency by predicting a T-wave at the beginning of the ECG, and there is none. The next secondary assumption gives rise to an internal inconsistency by predicting AV → AT @ [439, 480]. But if the atria were activated by the AV-His during this time interval, they could not have recovered in time to be reactivated by the SA at 480. Thus, the assumed event, A6, would be masked and effectively canceled, and the hypothesis would be internally inconsistent, contradicting one of its own assumptions. The remaining secondary assumptions fail as well, and so this first initial hypothesis must be abandoned, having failed at an unjustifiable assumption, A6.

The failure of the first initial hypothesis (A1, A3, A6, A8,...) at A6 indicates that *all* hypotheses beginning with A1, A3, and A6 will meet the same failure. Thus, all of them are now pruned from the search space, and the

search continues with an initial hypothesis in which A6 is replaced by A7.

$$A1, A3, A7, A8, A11, A13, A16, \ldots$$

This hypothesis accounts for the P-wave at 480 with a *spontaneous* beat of the atria, rather than an activation of the atria by the SA node. Attempts to develop it fare better, and a consistent explanation is formed until assumption A16 (LBB $\rightarrow$ LV @ 1710) is encountered, and all attempts to develop it further fail. All hypotheses that would lead to the same failure are pruned from the search space, and the next hypothesis retrieved. It is the same as its predecessor, except that A16 is replaced by A17, but this fails at the same place, leading to more pruning. When A17 is replaced by A18, we finally have an initial hypothesis from which a complete explanation can be formed:

$$A1, A3, A7, A8, A11, A13, A18, \ldots$$

This explanation accounts for all waves on the ECG, showing four normal sinus beats, the spontaneous beat initiated by the atria at 480, and two beats that meet and cancel each other out in the AV-His — a beat initiated by a spontaneous pulse from the left ventricle at 1710, and a beat initiated normally in the SA node at [1201, 1208]. All the events of this explanation (those visible on the ECG as well as those *not* visible) are represented in the laddergram shown in Figure 5.

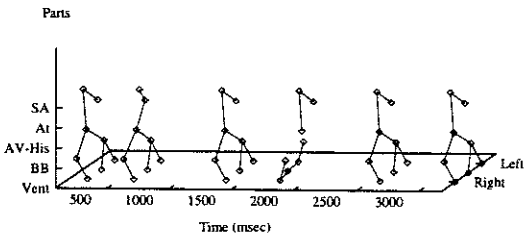


Figure 5: Laddergram of the first explanation generated in this example

But HOLMES, in its search for all possible explanations, presses on and finds three more sets of conduction paths that could also account for the ECG: one, whose paths are the same as those in the first explanation, except that the spontaneous ventricle beat at 1710 is initiated in the RV, and two more — identical to the first two respectively, except that the first beat in each is initiated by a spontaneous pulse in the atria, rather that a normal pulse from the SA node.

5 Results and conclusions

So far, the HOLMES system has been tested on nine different ECGs, and its generated explanations for each have been compared with a cardiologist's interpretation of the same ECG. For each, HOLMES has successfully reproduced physicians' interpretations, but with more details. For example, when analyzing a beat that a physician had judged to come from the area of the ventricles, HOLMES concluded that the beat must have come from the septum, the wall that separates the ventricles, because, had it come from either the left or the right ventricles, the corresponding wave on the ECG would have been wider than it was. In addition, HOLMES provides approximate timings for the electrical activations of *all* the parts of the heart, not just those whose activations are visible on the ECG.

One of HOLMES' most remarkable features is the speed with which it considers all the viable hypotheses in its exponential search space, summarized in the tables below. We evaluate the efficiency of each run with an efficiency ratio — the ratio of hypotheses considered (i.e. *not* ruled out) to the total hypotheses in the search space. The efficiency ratio of the least efficient search is $2.5 : 10^4$. This efficiency results from a search strategy that notes any inconsistency as soon as it arises, and allows the immediate pruning of all hypotheses that would lead to the same conflict. In addition, the propagation of quantitative temporal constraints allows more frequent discovery of inconsistencies. Assisting in this process are the frequent, precise ECG data that are propagated back, enabling the maximal constraint of prior events.

ECG	Number of Explanations	Size of Search Space	No. Hypotheses Examined	Efficiency Ratio
N1	2	2.6×10^{10}	80	$3.1 : 10^9$
N2	2	1.1×10^{12}	98	$8.9 : 10^{11}$
N3	3	2×10^7	57	$2.9 : 10^6$
N4	2	3.6×10^8	66	$1.8 : 10^7$
N5	2	2.8×10^{12}	96	$3.4 : 10^{11}$

HOLMES' performance statistics for the normal ECGs

ECG	Number of Explanations	Size of Search Space	No. Hypotheses Examined	Efficiency Ratio
A1	4	1.2×10^7	78	$6.5 : 10^6$
A2	2	3.3×10^5	48	$1.5 : 10^4$
A3	8	3.8×10^5	96	$2.5 : 10^4$
A4	6	3.8×10^5	66	$1.7 : 10^4$

HOLMES' performance statistics for the abnormal ECGs

These results point to several promising directions for future research. HOLMES' success in achieving a new level of detail in ECG interpretation encourages the continued development of its underlying model of heart conduction. Adding more expertise to the model would give it the capability of diagnosing a wider variety of arrhythmias, and the integrating it with a preprocessor that analyzes ECGs would be an addition welcomed by the current human "preprocessor". It also invites the investigation of other

domains of diagnosis in which temporal information plays an important role.

Moving beyond the domain of electrophysiology, this work points to the importance of developing quantitative models of cause and effect in any domain where efficient and detailed abductive inference is needed. It also suggests a method for filling in sparse data, and might well be used in reconstructing trends in data from sparse measurements. This could be especially useful in domains of medical diagnosis where frequent data collection is intrusive, difficult, or impossible.

Acknowledgments

This paper has benefitted greatly from the insightful comments and questions of Bill Henneman, Isaac Kohane, and John Josephson. I am also very grateful to Bipin Indurkhya for his time and thoughtful criticism on earlier drafts, and to system wizard Lou Hennessy for generous and cheerful production assistance at late hours.

References

[1] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[2] Ivan Bratko, Igor Mozetič, and Nada Lavrač. *KARDIO: A Study in Deep and Qualitative Knowledge for Expert Systems.* The MIT Press, 1989.

[3] Eugene Charniak and Drew McDermott. *Artificial Intelligence.* Addison-Wesley, 1985.

[4] Margaret E. Guertin. *Abductive Inference of Events: Diagnosing Cardiac Arrhythmias.* PhD thesis, Boston University, 1996.

[5] Ira J. Haimowitz and Isaac S. Kohane. Automated trend detection with alternate temporal hypotheses. In *Proceedings of the Thirteenth IJCAI*, pages 146–151, 1993.

[6] John R. Josephson and Susan G. Josephson. *Abductive Inference: Computation, Philosophy, Technology.* Cambridge University Press, 1994.

[7] M.G. Kahn, L.M. Fagan, and L.B. Sheiner. Model-based interpretation of time-varying medical data. In *Proceedings of the Thirteenth Annual Symposium on Computer Applications in Medical Care*, pages 28–32, Los Alamitos, CA, 1989. IEEE Computer Society Press.

[8] Isaac S. Kohane. *Temporal Reasoning in Medical Expert Systems.* PhD thesis, Boston University, 1987.

[9] William Long. Temporal reasoning in a causal probabilistic knowledge base. *Artificial Intelligence in Medicine*, 7, 1995. To appear.

[10] Henry J. L. Marriott. *ECG/PDQ.* Williams & Wilkins, 1987.

[11] Lionel H. Opie. *The Heart: Physiology and Metabolism*, chapter 5, pages 102–126. Raven Press, New York, second edition, 1991.

[12] Thomas A. Russ. Using hindsight in medical decision making. *Computer Methods and Programs in Biomedicine*, 32(1):81–90, 1990.

[13] Allen M. Scher. The electrocardiogram. In Harry D. Patton, Albert F. Fuchs, Bertil Hille, Allen M. Scher, and Robert Steiner, editors, *Textbook of Physiology: Circulation, Respiration, Body Fluids, Metabolism, and Endocrinology*, chapter 38, pages 796–819. W.B. Saunders Company, 21st edition, 1989.

[14] David A. Tong and Lawrence E. Widman. Model-based interpretation of the ECG: A methodology for temporal and spatial reasoning. *Computers and biomedical research*, 26(3):206–219, June 1993.

A RELATIONAL APPROACH TO TABULAR KNOWLEDGE

Rattikorn Hewett and John Leuchner
Department of Computer Science and Engineering
Florida Atlantic University
P.O. Box 3091, Boca Raton, FL 33431
email: hewett@cse.fau.edu, leuchner@cse.fau.edu

Abstract

We present a treatment of tabular knowledge (decision tables) as nested relations with one level of nesting. Our treatment offers advantages for comprehensibility and for application of domain knowledge to compression and extraction of rules.

1. INTRODUCTION

The decision table technique of representing knowledge in tabular form has been recognized as a useful formalism in many areas including verification and validation of knowledge bases, machine learning, and knowledge acquisition [10, 14].

Two major problem encountered in working with decision tables are finding a compact representation, which we refer to as the compression problem, and the extraction of kernel rules (i.e., rules having a minimal number of input variables [9]). The first problem is important since practical problems may have tens of input variables, each with several possible values. Compact representations are easier to comprehend and can improve the efficiency of kernel rule extraction.

The problems of compression and kernel rule extraction have been approached as Boolean function minimization problems in which all prime implicants are generated and a minimal covering is found [1, 5, 8, 11, 15]. Obtaining exact solutions to such problems is computationally expensive, and several heuristic approaches have been developed [2, 6]. Recently, approaches to improving the comprehensibility of tabular knowledge have been advanced [7, 12].

In this paper we present a relational approach that represents and manipulates decision tables as non-first-normal form relations with one level of nesting. Our approach has advantages for comprehensibility and interpretation of decision tables.

2. BASIC DEFINITIONS

A *rule* is an assignment of a set of values to each member of a set of *attributes*, which is called the *scheme* of the rule. The values assigned to a particular attribute are taken from a *domain* of values for that attribute. If each set of values is restricted to be a singleton, this definition of rule is equivalent to the definition of a tuple in relational database theory. The set of attributes, unlike that in a relational database, consists of *input* attributes (also referred to as *input variables* or *features*) and a *decision* attribute (also called the *decision variable* or *class* of the rule). A set of rules over a common set of attributes is a *decision table*. A decision table can be represented as a table with column headings for each of the attributes. Each rule is a row in the table which gives a set of values for each column heading. We refer to rules that have exactly one value for each attribute as *flat* or *simple* rules. A decision table consisting of flat rules is equivalent to a first normal form database relation [4, 13], except for the distinction between input and decision variables. When a relation has sets of values for attributes, a relation is said to be nested with one level of nesting. We refer to a decision table or relation with one level of nesting as *second-order*. Non-empty rules of a second-order table that are not flat are referred to as *complex*. rules

A decision table is used as follows. Given a value for each input attribute, each row of the table is examined. If a row is found whose set of values for each input attribute contains the value given for that attribute, the assignment of values for the decision attributes is returned as part of the result. Decision tables which

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/112

return the same results for all inputs are functionally equivalent.

Any decision table T is functionally equivalent to a unique set of flat rules, called the *flat rules of* T, which we denote by $\varphi(T)$. The set $\varphi(T)$ consists exactly of those flat rules whose attribute values are elements of the corresponding attribute value sets of some rule in T. The *flat size* of a decision table or rule is the size of its set of flat rules. A decision table T *covers* another decision table T' if $\varphi(T') \subseteq \varphi(T)$. Two decision tables (or rules) are *equivalent* if and only if they cover one another, i.e., if their sets of flat rules are the same. A rule in a table is *redundant* if it is covered by the other rules in the table. Decision tables T and S are *disjoint* if $\varphi(T) \cap \varphi(S)$ is empty, otherwise they are *intersecting*. These definitions extend from decision tables to rules in the obvious way.

The advantages of using a second-order decision table can be substantial. Suppose, for example, that a table has a scheme with twenty input variables and consists of a single rule that has exactly two values for input attribute. Using the table requires checking whether the value given for each input attribute is in the set of values assigned to that attribute by the rule. An equivalent table of flat rules would contain over a million rules, and the input would have to be checked against each one. It is not uncommon to be given a set of flat rules as a starting point. For example, a set of rules is often the starting point in machine learning where each flat rule represents a training example. Thus, compression of a decision table of flat rules into a second-order table is a practical problem.

3 SIMPLIFYING DECISION TABLE RELATION

Several operations, useful in simplifying a decision table, can be defined in terms of tuples and relations. The distinction between input and decision attributes is unimportant during the compression process. To clearly illustrate our basic techniques, in this paper, we assume rules to have non-empty set values for each attribute. Thus, we ignore *empty rules*, i.e., rules that have the empty set assigned to some attribute, and eliminate them if they occur as a result of any operation. (We explore the uses of allowing the empty set as an attribute value in a rule in [3].) Let $\mathcal{A}$ be a set of attributes and T a decision table over $\mathcal{A}$. For rules r and s in T and attribute A in $\mathcal{A}$, we use r(A) to indicate the set of values r assigns to A and we say that r *covers* s if r(A) contains s(A) for each A. We define a rule t to be the *A-join* of rules r and s if $t(A) = r(A) \cup s(A)$ and $t(B) = r(B) \cap s(B)$ for $B \neq A$. The *join* of rules r and s is the rule whose value for each attribute A is $r(A) \cup s(A)$. Given rules r and s in T and an attribute B in $\mathcal{A}$, it can be shown that the following operations can be applied without changing the meaning of T.

Add Directly Covered Rule: Rule t can be added to T if r covers t.

Delete Directly Covered Rule: Rule s can be removed from T if r covers s.

Delete Redundant Rule: If rule of r is covered by the other rules in T, then r can be removed from T.

Join: If r and s agree on all attributes except B, then r and s can be replaced by the join of r and s.

Extend: If r covers s for all attributes except B, then s can be replaced by rule t, a B-join of rules r and s. Note that in this case $t(A) = s(A)$ for $A \neq B$.

Reduce: If r covers s for all attributes except B, then s can be replaced by t where $t(A) = s(A)$ for $A \neq B$ and t(B) is the set difference $s(B) - r(B)$. Note when r(B) and s(B) are disjoint, replacing rule s by t gives no change since s and t are the same.

Split: Rule r can be replaced by rules u and v as long as $u(A) = v(A) = r(A)$ for $A \neq B$ and $r(B) = u(B) \cup v(B)$.

Attribute Union: Rule t can be added to T if t is a B-join of rules r and s.

If two decision tables T and S are equivalent, then there exists a sequence of the above operations which transforms T into S. Thus, these operations can be used to compress a decision table into a minimal equivalent table. Note that only three of the above operations reduce the length of the decision table. The rest reshape the decision table by extending or reducing the value sets that rules assign to attributes.

4 EXAMPLE

We now give a simple example, using three attributes A, B and C, to illustrate how these operations can be applied. For simplicity we drop set braces and commas when writing, e.g., in Rule 1 of Table 1, the value set {a, a'} is represented by a string aa'.

We begin with Table 1 consisting of 5 rules. Since Rule 2 covers Rule 4, we can eliminate Rule 4. Also, since rules 2 and 5 agree on all attributes except C, they can be replaced by their join, Rule 2'. The result of applying these operations is Table 2.

Table 1

Rule	A	B	C
1	aa'	b'	c
2	a	b	cc'
3	a'	bb'	c
4	a	b	c'
5	a	b	c"

Table 2

Rule	A	B	C
1	aa'	b'	c
2'	a	b	cc'c"
3	a'	bb'	c

Table 3

Rule	A	B	C
1	aa'	b'	c
2'	a	b	cc'c"
3'	a'	b	c

Table 4

Rule	A	B	C
1	aa'	b'	c
2'	a	b	cc'c"
3	a'	bb'	c
3'	a	bb'	c

In general, there will be many choices for the rules and operations that can be used to transform a table. For example, consider Table 2. Since Rule 1 covers Rule 3 in all

attributes except B, both the Extend and Reduce operations are applicable. Applying Extend to Rules 1 and 3 does not change the table, but Reduce can be used to replace Rule 3 by Rule 3' to obtain Table 3. On the other hand, by adding the B-join of Rules 1 and 2' of Table 2, we obtain Rule 3' as shown in Table 4. Note that since the attribute union for any pair of rules and any attribute can be added to the table, we can add the attribute joins of Rules 1 and 3 to Table 2. Each such addition results in a directly covered rule whose removal returns us to Table 2.

Applying Expand on attribute A to Rules 2' and 3' of Table 3, Rule 3' is replaced by Rule 3" resulting in Table 5. Table 6 is the result of replacing Rules 3 and 3' of Table 4 by their join.

Table 5

Rule	A	B	C
1	aa'	b'	c
2'	a	b	cc'c"
3"	aa'	b	c

Table 6

Rule	A	B	C
1	aa'	b'	c
2'	a	b	cc'c"
3"	aa'	bb'	c

In Table 5, Rules 1 and 3" can be replaced by their join, Rule 1', producing Table 7. Applying Reduce to Table 7, Rule 2' is replaced by 2" to get Table 8. Since Rule 1 of Table 6 is covered by Rule 3", it can be eliminated. Applying Reduce to the resulting table, yields Table 8 (with different Rule labels).

Table 7

Rule	A	B	C
1'	aa'	bb'	c
2'	a	b	cc'c"

Table 8

Rule	A	B	C
1'	aa'	bb'	c
2"	a	b	c'c"

Figure 1 illustrates a partial diagram summarizing the results of these transformations. Each node represents a decision table, numbered as above, and each edge represents a transformation operation on a specific rule or pair of rules. For example, from Table 2, at least four operations are applicable.

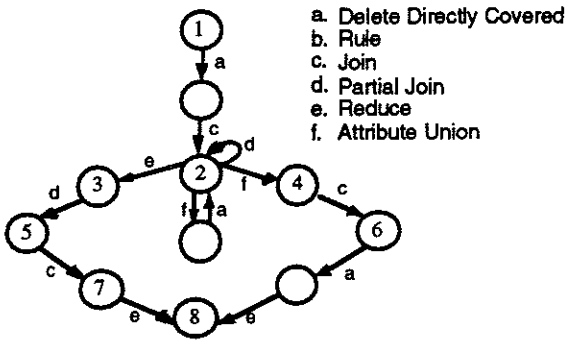


Figure 1. A partial diagram of compression processes.

Each of the above operations maintains equivalence. Equivalent tables may have different length or or the same length but different rules. In the above example, Table 1 of length 5 is compressed to Tables 7 and 8 of length 2. Searching for a path of applicable operations to compress the a table to a minimum faces a combinatoric explosion in the choices for operations and pairs of rules to operate on.

5 R2D2: A TABLE MANIPULATION SYSTEM

We have implemented a system R2D2 (second order relational definition of decision tables) which allows a user to view rules as sets of values for attributes and to apply the operations listed above. For efficiency, rules are represented internally as bit vectors.

Since finding an absolute minimum is not computationally feasible, R2D2 was designed to facilitate flexible experimentation with table compression. Along with the operations that add, remove, or replace rules or change their attribute value sets, R2D2 allows the user to order the rules in a table, e.g., by flat size, and to rank input attributes with respect to their influence on the result attribute.

The length of a decision table is reduced by the removal of redundant rules or by the replacement of a pair of rules by their join. Other operations are used to change the table so that some existing rules either become redundant or can be replaced by their join with another rule. R2D2 provides two methods of finding pairs of rules that can be replaced with their join. The first finds a pair of rules r and s such that the join of r and s is equivalent to {r, s}. Such a pair is called *locally joinable*. The second method finds a pair of rules whose join is covered by the table. Such a pair is called *globally joinable*. Determining whether a pair of rules is locally joinable is relatively easy since this is the case if and only if either one rule covers the other or the two rules are equal on all attributes but one.

Deciding whether the join of a pair or rules is globally joinable requires determining whether the join of the rules is covered by the table. Thus, finding globally joinable pairs and finding redundant rules both reduce to determining whether a given decision table T covers a given rule r. The straightforward way to do this is to iterate through the flat rules of r , checking whether each flat rule is covered by some rule of T. If the flat size of r is very large, this is not computationally feasible, so R2D2 also provides an *intersection method* , described in [3], as an alternative to the iteration method. The intersection method is used if the flat size of the rule exceeds a limit specified by the user.

R2D2 allows the user several options in applying joins. Local or global joining may be applied until a fixed point is reached or until a specified number of joins have been performed. We refer to the latter as *limited* joining. Local joining may also be applied using the ranks of attributes. In this case, R2D2 attempts to find pairs of rules that differ only on a specific attribute, starting with the attribute which is ranked as least important in terms of its

influence on the decision variable. This is effective if the initial table is reasonably complete since the rules in a well compressed table are likely to have larger attribute value sets for less important attributes. Note that a value set consisting of all possible values for an attribute represents a "don't care" condition for that attribute.

For greater efficiency, R2D2 sorts the decision table before searching for globally joinable pairs. Sorting rules in increasing order of flat size smaller rules to be considered first for joining and the resulting join is likely to be relatively small. Thus it is likely to take less time to check if the resulting join is covered by the table and so joinable rules can be joined faster

6 EXPERIMENTS ON COMPRESSION

We run R2D2, which is implemented in C++, on a 75 Meg Hz Pentium PC. In order to test the effectiveness of our compression techniques, we start with a decision table with a small number of disjoint rules, flatten the table (i.e., replace T with $\varphi(T)$), and shuffle the result. We then apply operations to compress the table. Comparing the result to the original set of rules gives us some idea of the effectiveness of our compression process.

6.1 Experimental results

Table 9 shows an example of a decision table of length 8 that we used as input. The table has seven attributes with value sets of size 2, two with value sets of size 3 and two with value sets of size 4. As in our previous example, we drop set braces and commas when writing value sets, e.g., the entry "yn" represents {y, n}, etc.

Rule	A	B	C	D	E	F	G	H	I	J	K
0	n	n	n	yn	n	f	lmh	yn	n	tbai	l'
1	y	n	y	yn	n	f	lmh	yn	n	tbai	l'
2	y	n	yn	yn	n	f	lmh	yn	y	tbai	l'
3	n	y	yn	n	n	f	lmh	yn	yn	tbai	l'
4	y	y	yn	y	n	f	lmh	yn	yn	tbai	l'
5	y	n	yn	yn	y	f	lmh	yn	yn	tbai	m
6	y	yn	yn	yn	n	f	lmh	yn	yn	tbai	m
7	y	yn	yn	yn	n	f	lmh	yn	yn	tbai	s

Table 9. Table before flattening.

Rule	A	B	C	D	E	F	G	H	I	J	K
0	n	n	n	yn	n	f	lmh	yn	n	tbai	l'
1	y	n	y	yn	n	f	lmh	yn	d	tbai	l'
2	y	n	n	yn	n	f	lmh	yn	y	tbai	l'
3	n	y	yn	n	n	f	lmh	yn	yn	tbai	l'
4	y	y	yn	y	n	f	lmh	yn	yn	tbai	l'
5	y	n	yn	yn	y	f	lmh	yn	yn	tbai	m
6	y	yn	yn	yn	n	f	lmh	yn	yn	tbai	sm

Table 10. Table after compression.

The flattened input table had 1344 rules which R2D2 compressed to the 7 rules shown in Table 10. Although the two tables do not contain the same set of rules (as indicated by underlined entries in Table 10, it is easy to see that they are equivalent.

The flat size of a table is no greater than the sum of the flat sizes of its rules. If the flattened size of a table T is equal to the sum of the flattened sizes of its rules, then the rules of T are disjoint and no flat rule is covered by more that one rule of T. Violation of this condition is referred to as *redundancy*. Note that if we start with a table without redundancy and compress it, the average flat size of its rules must increase.

Initial table length	Sum of rule flat sizes	Flattened table length	Result table length	Result Sum of flat sizes	Time (secs)
2	768	768	2	768	25
2	4864	4864	2	4864	2965
3	2112	2112	3	2112	956
6	576	576	6	576	30
8	1344	1344	7	1344	134
14	3072	3072	12	3216	690
16	9216	9216	15	9296	7246

Table 11. Compression results obtained from R2D2.

Table 11 shows partial results of our experiments with R2D2 for compressing several different decision tables. The number of attributes in these tables varied between 11 and 21, and attribute domains contained between 2 and 10 values. The sum of rule flat sizes gives an indication of the average rule "size" and therefore of the amount of redundancy in the table.

6.2 Using heuristics in R2D2

In general, a given decision table has many equivalent forms. R2D2's operations allow us to move between these equivalent forms, but we are primarily interested in operations that result in compression of a table. Hence, we have concentrated our experimentation on using local joining, global joining, reordering of rules by flat size, and reducing the size of attribute value sets. Our experiments have produced the following observations:

1. As expected, local joining is faster than global joining, since only two rules need to be examined rather than the entire table.
2. The time required to determine whether a pair of rules is globally joinable is extremely variable. Although global joining has always worked in our experiments, there can be cases in which testing whether a table covers a join is not computationally feasible. We plan to modify R2D2 to avoid these computations.
3. Local joining over a set of disjoint rules results in a set of disjoint rules, i.e., local joining cannot introduce redundancy. Global joining can introduce redundancy. Thus, although our experiments start with a set of disjoint rules, intermediate results contain intersecting rules.
4. If no pairs are locally joinable but the table contains intersecting rules, reduction may make more local joining possible. Reduction can also speed up global joining

since it decreases the sizes of attribute value sets.

5. The most effective way to use local and global joining for compression seems to be repetition of the sequence: (a) local joining to a fixed point, (b) reduction and removal of redundant rules, (c) local joining to a fixed point, (d) limited global joining.

6. All of our experiments have resulted in a table whose length is close to that of the original, input table, although the rules are usually different and may be intersecting.

7 CONCLUDING REMARKS

The ability to represent a very large set of flat rules with a few complex ones containing large attribute value sets is a great advantage. Shorter decision tables are more comprehensible and more efficient to use. Moreover, it is not always obvious and it is computationally expensive to determine whether a set of flat rules covers a complex rule. On the other hand, the flat rules covered by a complex rule are readily accessible.

Rules with large value sets for particular attributes are also desirable. In the extreme case, a rule may have a value set for attribute A containing all possible values of A, which is a "don't care" condition and thus simplifies some query processing. Rules with some large value sets can also elucidate concepts that are hidden in a collection of rules with only small value sets.

We have shown how we represent tabular knowledge using second order relations and discussed our experiments on compressing second-order tables. Our future work includes extension of R2D2 to make use of rules with empty attribute value sets and mechanisms that increase the informational content of a decision table by extrapolation.

Acknowledgements
This research is supported by NSF grant no. IRI-9308425.

References

[1] Bowman, R. and E. McVey, 1970. A method for the fast approximation solution of large prime implicant charts, *IEEE Transaction on Computing* **C-19**, 169.

[2] Hong, S., R. Cain and D. Ostapko, 1974. MINI: A Heuristic Approach for Logic Minimization, *IBM Journal of Research and Development:* 443-458.

[3] Leuchner, J. and R. Hewett, 1996. *Second Order Relations to Tabular Knowledge,* TR-CSE-96-4, Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, Florida.

[4] Maier, D., 1983. *The Theory of Relational Databases.* Rockville, MD: Computer Science Press.

[5] McCluskey, E., 1956. Minimization of Boolean Functions, *Bell System Technical Journal* **35**, 1417.

[6] Michalski, R., 1969. On the quasi-minimal solution of the general covering problem. In Proceedings of the Fifth International Symposium on Information Processing (FCIP69).

[7] Michalski, R., J. Carbonell and T. Michell, 1983. A theory and methodology of inductive learning, *Machine Learning: An Artificial Intelligence Approach*, Chapter 4, Tioga Press, Palo Alto, 83-134.

[8] Quine, W., 1955. A way to simplify truth functions, *American Math. Monthly* **62**, 627.

[9] Rymon, R., 1994. On Kernel Rules and Prime Implicants. In *Proceedings of the twelfth National Conference on Artificial Intelligence,* pp. 181-186.

[10] Santos-Gomez, L. and M. Darnell, 1992. Empirical Evaluation of Decision Tables for Constructing and Comprehending Expert System Rules, *Knowledge Acquisition,* 4: 427-444.

[11] Slagle, J., C. Chang and R. Lee, 1970. A new algorithm for generating prime implicants, *IEEE Transaction on Computing* **C-19**, 304.

[12] Sugiura, A., M. Riesenhuber and Y. Koseki, 1994. Comprehensibility Improvement of Tabular Knowledge Bases. In *Proceedings of the twelfth National Conference on Artificial Intelligence,* pp. 716-721.

[13] Ullman, J., 1982. *Principles of Database Systems.* Second Edition. Rockville, MD: Computer Science press.

[14] Vanthienen, J. and E. Dries, 1994. Illustration of a decision table tool for specifying and implementing knowledge based systems, *International Journal on Artificial Intelligence Tools,* 3(2): 267-288.

[15] Wagner, E., 1968. An axiomatic treatment of Roth's extraction algorithm, research report RC 2205, IBM Thomas J. Watson Research Center, Yorktown Heights, New York.

A Memory Model Enabling Case-Based Reasoning to Take Advantage both from Experience and Theory in Clinical Psychiatry

Isabelle Bichindaritz
Université René Descartes-Paris 5
LIAP-5, UFR de mathématiques et informatique
45 rue des Saints-pères
75006 Paris, France
tel : (+33) 1 44 55 35 63
email : bici@math-info.univ-paris5.fr

Abstract

The theory of dynamic memory aims to study how memory, reasoning and learning can be made inseparable in a computer system. The case-based reasoning system presented here pursues the same ideal. But one of its peculiarities is to take advantage both from experimental and from theoretical knowledge. This competence is enabled by its memory model, which includes two parts : an experimental memory, composed of cases and concepts, and a theoretical memory, composed of prototypes and models. The reasoning process can then use any of these entities depending on the task to realize. The role of the theoretical entities is essentially to explain the case-based reasoning process, to constrain it, and to compensate for its lack of experience. It is applied to the clinical expertise of eating disorders in psychiatry.

1 INTRODUCTION

Case-based reasoning is well known as an artificial intelligence methodology for the processing of experimental knowledge [1]. Experimental knowledge is some knowledge that has kept some links with the objects from which it has been learnt, whereas theoretical knowledge has kept none of these links. The general idea underlying case-based reasoning is that, in order to process a new case, corresponding to the description of a situation, it is better to use one or several cases previously experienced, eventually after adapting them. If early case-based reasoners were totally empirical systems, the role of theoretical knowledge has been shown to be more and more prominent for them. Hybrid systems, between case-based reasoning, and a theoretical artificial intelligence methodology, such as model-based or rule-based reasoning have been proposed.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/117 ©1996 FLAIRS

Nevertheless, a deeper understanding of the methodology of case-based reasoning points to a different way of taking advantage both from theoretical and from experimental knowledge. A look back to the precursory theory of dynamic memory [2], in the track of which the first case-based reasoning systems have followed, permits to specify the cognitive model underlying case-based reasoning. It can be expressed as a methodology allowing memory, reasoning and learning to be as closely joined as possible. It is this recognition of the prominent role played by memory in reasoning that must be looked for in case-based reasoning.

So a memory model is proposed in this paper to allow a case-based reasoner to take advantage both from theoretical and from experimental knowledge. It consists of two parts, a theoretical memory, composed of prototypes and models, and an experimental memory, composed of cases and concepts. The reasoning process is, at an abstract level, the same for all the memory entities. For that purpose, a unified knowledge representation language has been defined, capable of representing any entity in memory, whether a case, a concept, a prototype or a model, and permitting to the system to deal with them alternately.

2 THE MNAOMIA CASE-BASED REASONING SYSTEM

MNAOMIA is a case-based reasoning system the aim of which is to realize a complete reasoning system taking advantage both from experimental and from theoretical knowledge in order to realize different cognitive tasks. It can provide assistance to experts in the realization of these tasks. It is applied to the clinical expertise of eating disorders in psychiatry, which will be presented later, but profits from a general formalization.

In complex domains, such as medical domains, it proves necessary for case-based reasoning systems both to be able to realize several cognitive tasks, such as diagnosis, treatment [3] and clinical research, and to take advantage both from experimental and from theoretical knowledge, because this is the way the experts reason. The same clinicians perform

diagnosis and clinical research [4]. They start using mostly theoretical knowledge, then they learn through practice [5]. They always keep, and increase, these two types of knowledge and take advantage the most from them depending on the cognitive tasks they perform.

The functional architecture of the system is presented in Figure 1.

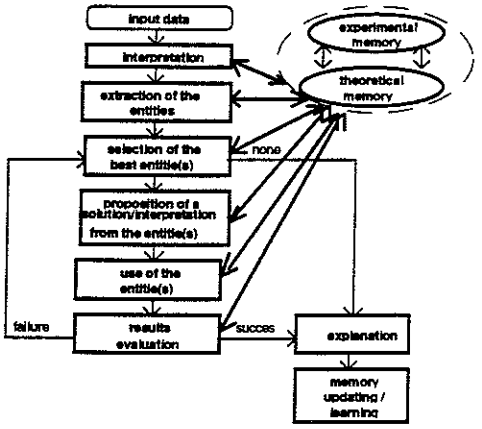


Figure 1. Functional architecture of the system.

3 THE MEMORY MODEL OF MNAOMIA

MNAOMIA's memory is a network of entities linked by relations. The entities can be cases, concepts, prototypes and models.

3.1 Knowledge Representation

The knowledge representation language is that of binary predicates, knowing that binary predicates can express predicates of higher order by conjunctions of binary predicates.

Each entity E in memory is represented as a conjunction of description elements EL_i where each EL_i is a $<e_i, arg_i>$ pair. The e_i are either attributes (in which case the arg_i are values), or relations (in which case the arg_i are nodes, that is to say either other entities in memory, or other description elements). So each entity E is represented by :

$$E = \bigwedge_i <e_i, arg_i> \qquad (1)$$

3.2 Memory Composition

The memory M contains two complementary components : an experimental memory M_E and a theoretical memory M_T :

$$M = M_E \cup M_T$$

3.3 Experimental Memory

The experimental memory M_E is a set of cases c_i and of concepts C_j organized in a network by some relations or links R_k, also called indexes:

$$M_E = <K, C, R>$$
where $K = \{...c_i...\}$, $C = \{...C_j...\}$
and $R = \{...R_k(n1_k, n2_k)...\}$
with $n1_k, n2_k \in K \cup C$.

The concepts are organized in hierarchies by abstraction links, and generalization links. The cases are linked to concepts by instanciation links.

Cases. Cases are represented as in (1), where the description elements can be of the two types : attributes and relations.

$$c_i = \bigwedge_t <Att_t, val_t> \wedge \bigwedge_r <Rel_r, no_r> \qquad (2)$$

Concepts. Concepts are represented as in (1), where the description elements can also be of the two types. The main difference with the cases representation is that they are learnt from them by incremental concept learning [4]. They are constantly updated, and so carry variables enabling the learning process. These variables are called discrimination weighted variables, and represent a confidence in the description element for a specific concept.

$$C_j = \bigwedge_u <Att_u, val_u> \wedge \bigwedge_s <Rel_s, no_s, n+_s, n-_s> \qquad (3)$$

3.4 Theoretical Memory

The theoretical memory M_T is a set of prototypes m_i and of models M_j organized in a network by some relations R_k (also called links or indexes).

$$M_T = <P, D, R>$$
where $P = \{...m_i...\}$, $D = \{...M_j...\}$
and $R = \{...R_k(n1_k, n2_k)...\}$
with $n1_k, n2_k \in P \cup D$.

The models can be organized in hierarchies by abstraction links, and generalization links. The prototypes are linked to models by instantiation links. When no model is available, the prototypes are not linked to models.

Prototypes. Prototypes are represented as in (1), where the description elements can be of the two types : attributes and relations.

$$m_i = \bigwedge_t <Att_t, val_t> \wedge \bigwedge_r <Rel_r, no_r> \qquad (4)$$

Models. Models are also represented as in (1), where the description elements can also be of the two types. The main difference with the other entities is that here the val_u and the no_s can be variables.

$$M_j = \bigwedge_u <Att_u, val_u> \wedge \bigwedge_s <Rel_s, no_s> \qquad (5)$$

3.5 Memory organization

The memory is organized around some specialized models of the theoretical memory called the points of view. A point of

view is a filter for the description elements El_i associated with each entity in memory. Points of view are for instance associated to each cognitive task performed by the system, but also to sub-domains of the application domain. Each can be expressed in the form of a conjunction of $<El_i, n_{i,pert}>$ pairs, where $n_{i,pert}$ is a pertinence weighted variable associated with the description element El_i. This pertinence weight is all the more increased as this element is important according to this point of view. The El_i element may be either an $<Att_i, val_i>$ pair, or a $<Rel_i, arg_i>$ pair, or simply an attribute Att_i or a relation Rel_j.

Consequently a point of view is expressed in this form :

$$P = \wedge_i <El_i, n_{i,pert}> \quad (6)$$

Each point of view element El_i is then associated with a set of entities Ent_j in memory, all sharing this description element. Moreover, a predictivity weighted variable $n_{j,pred}$ is here also associated with each entity in memory. This predictivity weight is the more increased as the corresponding element permitted to favorably select an entity, and the more decreased as this selection leaded to a failure (credit and blame assignment).

So each element El_i drives the search towards a set of entities in memory :

$$El_i \rightarrow \wedge_j <Ent_j, n_{j,pred}> \quad (7)$$

Then each entity in memory is a conjunction of $<e_i, arg_i>$ pairs (1). In addition, a concept associates to each description element e_k two discrimination weighted variables $n+$ *and* $n-$. These variables are updated by learning. The positive discrimination variable is all the more increased as the description element has been matched positively during the search through the memory, and all the more decreased as this element has been unmatched.

4 THE REASONING PROCESS OF MNAOMIA

The reasoning process goes through several steps.

4.1 Input data interpretation

The aim of this step is to determine, from the input data such as Ent_e, the first point of view that will be activated in the theoretical memory. Indicating the task to perform, such as a diagnosis, or clinical research for instance, suffices for this determination.

4.2 Extraction of the entities

It is realized following the several indexing levels from the point of view. The extracted entities correspond to the Ent_j in (7), where the El_i have been initialized with the input data from the Ent_e. The set of the extracted entities is then :

$$\{ Ent_k \} = \cup_i \cup_j \{ Ent_{j,i} \} \quad (8)$$

4.3 Selection of the entities

In this step, the notion of similarity between entities is applied. It is a delicate problem that can be in this system solved in two very different ways. Two notions of similarity have been defined, and are used alternatively depending on the task to realize.

Measure-Based Similarity. The assessment of this kind of similarity is made by calculating, between the new entity presented to the system Ent_e and the set of the extracted entities (8) the entity Ent_j that maximizes the following measure :

$$sim(Ent_e, Ent_j) = \frac{\sum_{k=1}^{n} \alpha * sim(e_{i,k}, e_{l,k}) + \sum_{k=1}^{n} n_{k,i,pred} * n_{i,pert} * sim(e_{i,k}, e_{l,k})}{\alpha * n + \sum_{k=1}^{n} n_{k,j,pred} * n_{i,pert}} \quad (9)$$

where α is a minimal weight associated to the similarity of any two description elements.

Classification-Based Similarity. The other way of assessing the similarity between two entities is to follow the hierarchies in memory under which cases are indexed by instantiation links. This memory traversal starts at the concepts extracted in the set (8) and goes down the hierarchies of concepts following the abstraction/generalization relations. It stops when a concept has at least a description element that contradicts one of the description elements of Ent_e. The cases directly indexed under this concept are the most similar to Ent_e according to this classification-based similarity assessment.

4.4 Use of the selected entities

The entities selected are then used depending upon the kind of case-based reasoning task to realize. In problem-solving, the solution for the new entity is an adaptation of that of the entities selected. In interpretation, this solution is the construction of an argumentation linking the selected entities.

4.5 Evaluation

Two kinds of evaluation are here again performed. The first one is an a priori evaluation based on the quality of the inferences made, and of the degree of similarity between the entities used. The second one is an a posteriori evaluation, by feed-back from the application of the proposed process. It leads to an explanation of the result, whether success or failure.

4.6 Learning

Then learning can take place. The predictivity variables are updated, dependently from the first point of view. Then, an incremental concept learning takes place, so as to modify the organization of the concepts hierarchies, and to synthesize the experimental knowledge of the system, which can then be used to help interpretation tasks. It is detailed in [6].

5 APPLICATION

5.1 The Application Domain

The MNAOMIA system has been designed and realized for the psychiatry domain of eating disorders, in a specialized unit of the Clinique des Maladies Mentales et de l'Encéphale (service of Professor Bertrand Samuel-Lajeunesse). A mental disorder is a psychiatric syndrome, which means a well-defined set of symptoms and signs that can be observed in several different pathological states. The set of problems related to this psychiatry domain is :

1. **the ignorance of the true cause and nature** of this disorder ; no pathophysiological model is available, nor any sufficient pathognomonic one;
2. **the subjectivity of psychiatric signs**, which are set up by a psychiatrist's judgment ; methods such as standardized rating-scales and well-documented nosographies have been developed to reinforce the reliability of psychiatric evaluations ;
3. **the multiplicity of diagnoses** associated to a single clinical syndrome ; so the diagnosis process does not lead to a single diagnosis, but to a set among which a major diagnosis has to be chosen ;
4. **the availability of several possible nosographies**, among which that of the DSMIII-R [7] is the most commonly used ; it is presented by some of its authors as a prototypical model, which means that the belonging to a certain diagnostic category is assessed by the proximity to ideal prototypical patients. The diagnostic categories considered here are Anorexia Nervosa, Bulimia Nervosa and the combination of the two, which will be referred to as Anorexia-Bulimia.

5.2 The Experimental Memory

Cases. A case represents the evolution of a patient between several states. The factors influencing the patient between these states are also memorized. They are recorded in the patient's medical record.

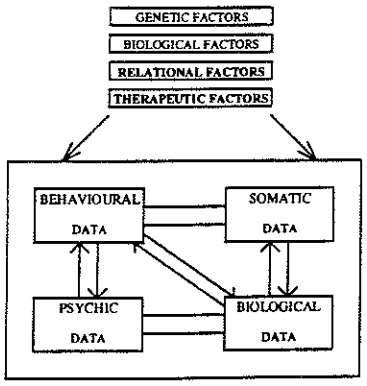


Figure 2. Representation of a contextual patient's state in a patient's case.

Each **state** of a patient corresponds to the data gathered about him or her at a given time in the following dimensions :

1. **General data** (such as civil status for example) ;
2. **Psychic data** gathered in different rating-scales or self rating-scales (affective, cognitive, ...) (such as depression assessment for example) ;
3. **Biological data** gathered through paraclinical examinations (such as blood numeration for example) ;
4. **Somatic data** gathered through somatic examination (such as edema for example) ;
5. **Behavioral data** gathered by the clinical staff's observation (such as anxious behavior for example).

The **factors** influencing the state of the patient are :

1. **Genetic and neurobiological** : they are non-observable and non-measurable, so will not be considered any longer, although probably very influential ;
2. **Relational** : family, cultural, social or professional ;
3. **Therapeutic** : the different treatments administered to the patient.

Figure 2 shows the representation of a contextual patient's state.

Other kinds of cases are represented in this part of the memory, such as the clinical staff cases and control subject cases for example.

5.3 The Theoretical Memory

Prototypes. Several kinds of prototypes have been created :

1. **Patient's prototypes** such as described by the DSMIII-R manual, but also by other important nosographies ;
2. **A normal subject prototype** gathering all standardized data known, with their means and standard deviation for numerical data (kaliemia for instance) ;
3. **Average subject prototypes** gathering the previous data, but for important sub-groups of patients, such as the diagnostic categories ; they are constantly updated ;
4. **Food prototypes** gathering known data about each food from a food composition table ;
5. **Prototypical treatment plans** established for each disorder and in each dimension.
6.

Models. As previously mentioned, the only model available in this application domain is that of Figure 2. It is very general, but still useful to give a synthetic view of a patient's state. It is much more detailed in the system.

Points of View. Several points of view have been defined :

1. **The global point of view** in which every description element is considered with an equal pertinence weight ;
2. **The eating disorders point of view** where only the description elements appearing in one of the nosographies are given ;
3. **The behavioral symptomatology point of view**;
4. **The somatic symptomatology point of view** ;
5. **The biological symptomatology point of view** ;

6. **The psychic symptomatology point of view** ;
7. **The cognitive tasks points of views** such as diagnosis for example.

5.4 The Reasoning Process

The system is capable of realizing several cognitive tasks, and of adapting to them. These tasks are currently diagnosis, treatment planning and clinical research. Patients' follow-up is under investigation. For instance the **diagnosis** problem is not only to classify the patient in an eating disorders diagnostic category, but also to determine his or her associated mental disorders, and his or her personality characteristics even if they are not pathological. The general algorithm of the diagnostic process is the same as that of MNAOMIA's general reasoning process. But here, the point of view of diagnosis is structured in different sub-points of view, the main corresponding to the four dimensions previously mentioned. So in each step, the reasoning process is realized in all the sub-points of view. In the use step, an argumentation linking the different diagnoses found is constructed : it is an example of a combination between a problem-solving task and an interpretation task. The concepts are also updated in each point of view, and the case added to the memory.

Table 1. Description of the patients for the diagnosis evaluation.

Patients	Number
Anorexia Nervosa	41
Bulimia Nervosa	30
Anorexia Bulimia	40
Other Eating Disorders	4
Total	115
Schizophrenia	13
Pathological Personality	43
Depression	3

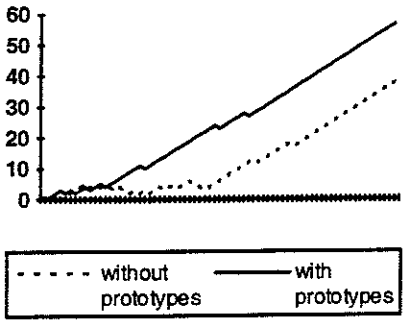


Figure 3. Accuracy of diagnosis (each ascending line between two X-axis points is a diagnosis success, and each descending line a failure).

6 RESULTS

Results are given here for the diagnosis at patients' admission. The accuracy of the diagnosis has been compared with that of the clinical staff after several weeks of hospital care. Table 1 describes the patients population, and Figure 3 shows the results for the first 60 cases processed.

The diagnosis accuracy is about 80% for the first 30 cases, and about 95% from the 30th case to the 115th. It is compared with the results of the same diagnosis process performed only with the data of the alimentary questionnaires of the patients, for which no prototype is available. The results after the 30th case are about the same (93% accuracy), but are very different for the first 30 cases. The advantage of the prototypes in theoretical memory is here obvious at least at the beginning of the reasoning process.

7 CONCLUSION

The case-based reasoning system presented here is an efficient clinical expertise assistant. Its abilities principally stem from its memory model, both in its static components (composition, organization) and in its dynamic components (memory accessing and learning processes). The domain of application did not permit to validate it completely, because of the lack of existing models. The formalization of the system allows to apply it to other psychiatric and medical domains in which it will be more thoroughly evaluated. Interesting work would be to connect this research with that about the patients medical record.

Acknowledgments

I thank Professeur Bertrand Samuel-Lajeunesse for letting me take part in his research about eating disorders at CMME.

References

1. Kolodner JL. Case-Based Reasoning, San Mateo: Morgan Kaufmann Publishers Inc., 1993.
2. Schank RC. Dynamic Memory : a Theory of Learning in Computers and People, Cambridge University Press, 1982.
3. Berger J, Hammond KJ. ROENTGEN : A Memory-Based Approach to Radiation Therapy Treatment, In : Proceedings of a Workshop on case-based reasoning (DARPA), Washington D.C., R. Bareiss (Edt.), San Mateo : Morgan Kaufmann. 1991; 203-214.
4. Bichindaritz I. A Case-Based Assistant for Clinical Psychiatry Expertise, In : Proceedings 18th SCAMC. Washington DC: AMIA, 1994, 673-677.
5. Kolodner JL, Kolodner RM. Using Experience in Clinical Problem Solving: Introduction and Framework, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-17, N°. 3, 1987; 420-431.
6. Bichindaritz I. Apprentissage de Concepts dans une Mémoire Dynamique : Raisonnement a partir de Cas Adaptable a la Tâche Cognitive, Ph.D. dissertation, Université René Descartes - Paris 5, 1994.
7. American Psychiatric Association. Diagnostic and Statistical Manual of Mental Disorders, 3rd Edition, Revised (DSM-III-R), Washington DC, APA, 1987.

Reasoning with Sequences of Intervals for Efficient Constraint Propagation

Lina Khatib
Florida Institute of Technology
lina@cs.fit.edu
http://www.cs.fit.edu/~lina/

Abstract

Traditional interval reasoning assumes that all events pick out convex intervals. We are extending this model by allowing events to take place over sequences of convex intervals. This allows for a more natural treatment of the part-whole structure underlying such entities as an interrupted event or an event that consists of several stages. More significantly, we find that this extension results in a more efficient method for propagating temporal information. This supports the contention that combining non-temporal, domain-specific information with temporal information will tend to improve the efficiency of the temporal reasoning process.

1 INTRODUCTION

It has been noted by Allen [1] and Williamson and Hanks [7], among others, that combining non-temporal, domain-specific information with temporal information will tend to improve the efficiency of the temporal reasoning process. This paper supports this contention by demonstrating rigorously that adding information to the effect that a collection of intervals is part of the same event results in a more efficient method for propagating temporal information about those events. Examples of this part-whole structure include collections of stages of the same event, or collections of parts of an interrupted event. Allowing interrupted events to be represented means more granularity to the characterization of events than in traditional approaches to interval temporal reasoning, which assume that all events pick out convex intervals.

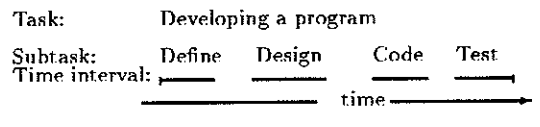
Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/122 ©1996 FLAIRS

The remainder of this paper contains, first, an introduction of an efficient and economical representation of sequences of intervals and their relations; second, a discussion of algorithms for propagating temporal information about temporal relations between these sequences; finally, an analysis and demonstration of the advantages gained by adopting this representation over the standard interval propagation methods.

2 REPRESENTATION

In this section we provide a representation of sequences of intervals and their temporal relations.

In many applications involving temporal reasoning, intervals are related in non-temporal ways. For example, a pair of intervals may represent two parts of an interrupted event, or two stages of a larger event. Consider, for example, the task of developing a program. This task consists of four subtasks: defining the problem, designing a solution, coding, and testing. Visually:



2.1 Convex Interval Approach

In the traditional interval-based temporal reasoning, the time units are convex intervals where a convex interval is a *contiguous* period of time. The interval calculus is based on thirteen basic binary relations between convex intervals and their operations. The thirteen basic relations between two convex intervals i and j are as follows.

1. i **precedes (p)** j: the end of i is before the beginning of j

2. i **meets (m)** j: the end of i is the same as the beginning of j

3. i **overlaps (o)** j: the beginning of i is before the beginning of j and the end of i is between the beginning and end of j

4. i **starts (s)** j: i and j begin at the same time but i ends first

5. i **finishes (f)** j: i starts after j starts but they end at the same time

6. i **during (f)** j: i starts after j starts but ends before j ends

7. i **equals (e)** j: i and j start together and end together

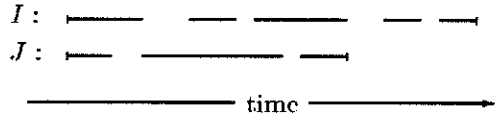
The remaining 6 relations are the inverse, in term of i and j, of the relations 1..6. We represent this fact by putting $\tilde{}$ on the corresponding relation symbol. Thus, the remaining 6 relations are: $\tilde{p}$, $\tilde{m}$, $\tilde{o}$, $\tilde{s}$, $\tilde{f}$, and $\tilde{d}$.

A relation between two intervals is given as a disjunction of one or more of the above basic relations. The interval algebra for these relations has the following major operations: $\times$ for intersection, $\circ$ for composition, and $\bar{}$ for complement. For more details see [1].

2.2 Non-convex Interval Approach

To generalize the interval calculus to allow for collections of (non-overlapping) intervals, it is necessary to relax the assumption of interval convexity. This makes the basic units of temporal reasoning sequences of convex intervals.

Example. Let I and J be two sequences of intervals that can be displayed graphically as follows.



Our framework is based on matrices that represent binary relations between sequences of intervals. The rows of the matrix represent the interval components of one sequence and the columns represent the interval components of the other. The entry in row (i) and column (j) is the relation between the i^{th} component of the first sequence and the j^{th} component of the other. Since the components of the sequences are convex intervals, the entries of the matrices are relations between convex intervals as defined in the traditional interval framework (known as Allen relations).

Thus, the matrix relation between I and J, R(I,J), consists of five rows (the size of I) and three columns (the size of J). The entry in the second row and third column, for example, should be the Allen relation between the second component of I and the third component of J which is (p) (precedes). We describe this as $R(I_2, J_3) = (p)$. Thus, the matrix relation of I and J is of the following form:

$$R(I,J) = \begin{vmatrix} \tilde{s} & m & p \\ \tilde{p} & d & p \\ \tilde{p} & \tilde{o} & \tilde{f} \\ \tilde{p} & \tilde{p} & \tilde{p} \\ \tilde{p} & \tilde{p} & \tilde{p} \end{vmatrix}$$

Matrix temporal relations have specific features that can be exploited to reduce computations and save space used in recording the knowledge. For example, notice in the previous example that the matrix can be viewed as divided into three *regions*: a region in which only p occurs, a region in which only $\tilde{p}$ occurs, and a mixed region in which any of the so-called Allen relations can occur. This reflects the fact that, for example, if an interval i of one sequence I is before interval j of another sequence J, then it is also before any interval that occurs after j in J. It can be easily seen that every consistent temporal relation between pairs of sequences can be divided into regions in this manner. Therefore, a canonical matrix relation has the following form:

M =

$(\tilde{p})$ region	mixed region	(p) region

By a *canonical form of a matrix* we mean a reduced and *consistent* matrix that only explicitly represents the temporal information in the mixed region; the temporal information in the other regions can be inferred, and hence no storage space is needed for them. A fast and simple algorithm to convert a matrix relation into canonical form is given in Figure 1. The canonical conversion algorithm checks the consistency condition for each element, in the matrix, with its four neighbors. By the neighbors of an element $a_{i,j}$ we mean the four elements that are immediately to the left, below, to the right, or above $a_{i,j}$ in the matrix; or, $ai, j-1$, $ai+1, j$, $ai, j+1$, and $ai-1, j$. If the algorithm detects any inconsistency, it will update the causing matrix entry and add it to the queue for further propagation.

A faster, but not as simple, algorithm has been devised to convert a matrix into canonical form. All

```
Procedure Canonical_Conv(A_{n×m})
/* A is a matrix relation to be converted
   into its canonical form */

begin
   Place each pair < i,j >, 1 ≤ i ≤ n , 1 ≤ j ≤ m,
     on a fifo queue Q;
   While Q not empty
     remove < i,j > from Q;
     if (j > 1) check_left(A,i,j);
     if (i < n) check_below(A,i,j);
     if (j < m) check_right(A,i,j);
     if (i > 1) check_above(A,i,j);
   end;
end;
```

Figure 1: Algorithm to convert a matrix relation into canonical form

the algorithms, definitions, proofs, and further details can be found in [2].

3 CONSTRAINT PROPAGATION

In this section we concentrate on the problem of performing constraint-propagation (determining path-consistency) on a network of interval sequences and their (reduced) matrix relations.

3.1 Convex Interval Network

In the traditional approach, constraint propagation is performed on a network of convex intervals. The nodes are the intervals and the the arcs are labeled with sets of binary relations from the 13 relations. The labels represent constraints on the relationships among the nodes.

The original constraint propagation algorithm, checks the consistency of every triplet in the network (3-consistency). It updates the labels by removing basic relations which cause inconsistency and propagates the updates over the network. It uses the operations defined for the relation algebra such as intersection, composition, and inversion [1].

3.2 Non-convex Interval Network

In our extension, constraint propagation is performed on a network of Non-convex intervals. The nodes are sequences of intervals and the arcs are labeled with matrix relations. The meaning of the labels is the same as in the traditional approach.

The constraint propagation algorithm, checks the consistency of every triplet in the network (3-consistency). It updates the labels, which are matrices, by removing basic relations from their entries that cause inconsistency. It propagates every update over the network. The constraint-propagation uses operations defined for matrix relation algebra for the updates.

First, notice that the basic operations that are required to perform path-consistency (intersection and compositions of relations) can be easily generalized to matrices. The intersection of two matrix relations $A_{n\times m}$ and $B_{n\times m}$, $A \otimes B$, is a matrix relation $C_{n\times m}$ where

$$C_{i,j} = A_{i,j} \times B_{i,j}$$

The composition of two matrix relations $A_{n\times r}$ and $B_{r\times m}$, $A \odot B$, is a matrix relation $C_{n\times m}$ where

$$C_{i,j} = \prod_{k=1}^{r} (A_{i,k} \circ B_{k,j})$$

Or,

$$C_{i,j} = A_{i,1} \circ B_{1,j} \times A_{i,2} \circ B_{2,j} \times \cdots \times A_{i,r} \circ B_{r,j}$$

The operators $\times$ and $\circ$ are the intersection and composition defined for convex interval relations [1]. Also, in calculating C, we need to consider only the subscripts of the mixed regions of A and B.

The resulting constraint propagation algorithm is presented in Figure 2. The procedure *Propagate* is called for every pair of NCIs, or sequences of intervals, I and J in the network. It checks the consistency of every triplet containing I and J (I,K,J). If inconsistency is detected, it updates the causing matrix entries and put the corresponding pair of NCIs on the Queue for further propagation.

Embedded in the algorithm is a procedure (Canonical_Conv), given in Figure 1, for generating the canonical form of a matrix relation. The purpose of such procedure in the propagation process is to keep all the matrices in the network in canonical form. This implies the matrices are consistent and reduced which results in more efficient operations. For more details see [2].

4 ANALYSIS

It can be demonstrated that constraint propagation with sequences performs better than traditional interval propagation in both space and time. The reason for this improvement is that when dealing with sequences of intervals the assumptions which went

```
Procedure Propagate(I, J)
/* Called to propagate the change to the relation
   between two sequences I and J to all other
   sequences. */

For each sequence K in Ncintervals do
  begin
    Temp := Table[I, K] ⊗ (Table[I, J] ⊙ Table[J, K]);
    If Temp = E_M then Signal Contradiction;
    If Table[I, K] ≠ Temp then
        Place pair < I, K > on Queue;
        Canonical_Conv(Temp);
        Table[I, K] := Temp;
    Temp := Table[K, J] ⊗ (Table[K, I] ⊙ Table[I, J]);
    If Temp = E_M then Signal Contradiction;
    If Table[K, J] ≠ Temp then
        Place pair < K, J > on Queue;
        Canonical_Conv(Temp);
        Table[K,J] := Temp;
  end;
```

Figure 2: Constraint Propagation Algorithms for Sequences of Intervals

into forming the ordered sequences can be represented *implicitly*, rather than explicitly, which results in a smaller knowledge base. Furthermore, the temporal reasoner does not have to propagate information about the temporal relations between elements within the same sequence during the reasoning process, which results in demonstratively fewer operations to be performed.

4.1 Complexity

The basic algorithms of Figure 2 were enhanced and supplemented by other algorithms that exploit the valuable properties of canonical matrix relations. The algorithms and full analysis can be found in [2]. Analyzing the run time of our constraint propagation algorithm and the traditional constraint propagation algorithm, we find them to be of the same asymptotic order but our algorithm has smaller coefficients[1]. The order of each of the algorithms is $O(k^3.n^3)$ where k is the number of sequences (or non-convex intervals) and n is the size of each sequence (number of convex intervals in a sequence).

4.2 Example

We present a simple example to show a sample of the input data we use for our testing and the measurements we take into consideration.

Assume 4 sequences of intervals: I_1 of size 3, I_2 of size 3, I_3 of size 2, and I_4 of size 4. The initial relations are given as follows:

$$R(I_1, I_2) = \begin{vmatrix} ds & p & p \\ \tilde{p}\tilde{o} & e & pm \\ \tilde{p}\tilde{m} & \tilde{p} & o \end{vmatrix}$$

$$R(I_1, I_3) = \begin{vmatrix} s & p \\ f & p\tilde{f} \\ \tilde{p} & \tilde{p} \end{vmatrix} \qquad R(I_2, I_3) = \begin{vmatrix} \tilde{f}o & p \\ \tilde{m}f & op \\ \tilde{p} & \tilde{p} \end{vmatrix}$$

$$R(I_2, I_4) = \begin{vmatrix} o & pm & p & pm \\ f & p & p & p \\ \tilde{p} & \tilde{p} & p & p \end{vmatrix}$$

$$R(I_3, I_4) = \begin{vmatrix} e & pm & p & p \\ \tilde{p}\tilde{o} & e & p & pm \end{vmatrix}$$

We use the number of interval compositions, which are expensive operations, as a measurement for time performance. Manually, we find that our approach requires **937** operations while the traditional approach requires **more than 1320** operations. The implementation assured these numbers and added that the exact number of operations required by the traditional method is **1360.**

For space complexity, we use the number of Allen relations that require memory storage as a measurement. Manually, we find our approach to require storages for **54** relation while the traditional approach requires storages for **66** relation.

Also, the CPU time measurement proved the advantages of our approach. For this example, our algorithm required **4.58** seconds while the other required **9.13** second.

4.3 Experimental Results

An implementation of both constraint propagation algorithms, the traditional one and our extended one, was conducted[2]. The algorithms were compared by testing them on 32 different sample inputs. The results showed that our algorithm had a better performance in term of CPU time and number of composition operations (which is the most expensive operation). We choose the CPU time measurement results to present in this paper.

The following table summarizes some of the experimental results we obtained when running the program on different sample data. It illustrates the improvement of our constraint propagation algorithm over the traditional one (except when each of the sequences consists of one interval).

[1] Assuming a sequence consists of more than one interval.

[2] The algorithms are implemented and tested by Hsin-yu Wang as a part of her Master thesis work.

Comparison of CPU time (in seconds)			
#CC	CI	NCI	
	time	time	# sequences
3	0.33	0.47	3
9	5.03	1.08	3
		5.90	9
18	39.92	3.20	3
		16.78	9
27	175.13	7.82	3
		38.23	9
36	Fail	16.60	3
		73.41	9

#CC: total number of convex intervals (convex components)
CI: convex interval approach
NCI: non-convex or sequence of intervals approach
#sequences: the number of sequences formed from the convex intervals
Note: #CC divided by #sequences gives the size of each sequence (assuming all sequences are of the same size)

5 SUMMARY

We have outlined an approach to interval temporal reasoning which exploits additional domain knowledge in the representation of the events and their relations. Specifically, collections of intervals can be formed from knowledge about the part-whole compositional structure of the events in the domain. This part-whole structure can represent things like different stages of the same event, or different segments of a single, interrupted event, thus allowing for greater granularity in the representation of temporal knowledge, which normally requires that events be uninterrupted. An assumption is made that the intervals can be sequentially ordered, thus do not overlap. Temporal relations between such sequences can be mapped to matrices. Additional temporal knowledge can be brought to bear in the recognition that all matrices corresponding to consistent binary temporal relations between sequences consist of 3 distinct regions. Consequently, only the information in one of the regions needs to be explicitly stored, which results in a more efficient technique for constraint propagation.

Finally, the success of our approach in extending the interval model has led us to apply it for the purpose of extending the point model [6].

References

[1] J. Allen. (1983) Maintaining Knowledge About Temporal Intervals. In Brachman, R., and Levesque, H., (eds.) *Readings in Knowledge representation*, (San Mateo:Morgan Kaufman), 510- 521.

[2] L. al-Khatib. *Reasoning with Non-convex Time intervals*. Ph.D. Dissertation, Computer Science Program, Florida Institute of Technology (1994).

[3] P. Ladkin. *Time Representation: A Taxonomy of Interval Relations*. In Proceedings of AAAI-86, pp. 360-366.

[4] G. Ligozat. *Weak Representation of Interval Algebras*. In Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90), Vol. 2, pp. 715-720.

[5] R. Morris, and L. al-Khatib. *An Interval-Based Temporal Relational Calculus For Events With Gaps*. The Journal of Experimental and Theoretical Artificial Intelligence, 3, (1991), 87-107.

[6] R. Wetprasit, A. Sattar, and L. Khatib. *Reasoning with Multi-Point Events*. AI-96, the eleventh biennial conference on Artificial Intelligence sponsored by the Canadian Society for Computational Studies of Intelligence, (forthcoming).

[7] M. Williamson, and S. Hanks. *Exploiting Domain Structure to Achieve Efficient Temporal Reasoning*. AAAI-92 Workshop Program, Implementing Temporal Reasoning. (1992), 178-190.

An intelligent approach to flexible resource allocation in multiagent systems*

Will Briggs Diane Cook
Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX 76019 U. S. A.
wbriggs@sirius.uta.edu, cook@centauri.uta.edu

Abstract

Although communication is generally considered to dominate over processing cost in distributed systems, the problem of communication cost in multiagent planning has not been sufficiently addressed. Exclusive access to resources can prevent communication due to conflict, which dominates communication costs; but this can prevent agents from finding solutions. We propose a clustering method for allowing resources to those agents most likely to need them, and show that it significantly reduces communication costs without sacrificing soundness, when allowed gradual relaxation of the exclusivity requirement.

1 INTRODUCTION

As automation of intelligent tasks increases, the need will arise for heterogeneous agents working in a common environment. As a result, agents will need to plan and coordinate plans in real time.

Cook [Coo94] has described three methods of handling multiple agents: central control, distributed control, and local control (no communication). She notes that central control is effective when communication is reliable, and local control is effective if no communication is needed; otherwise, distributed control is necessary. This is the protocol generally used with multiagent systems [DM91, vM92, GD94].

Communication is generally more time-consuming and less reliable than computation. In the case of planning, both are intractable, but communication even more so. Consider two agents searching the space of possible plans. Each plan must be reconciled with the other in to avoid harmful interactions (and, possibly, to facilitate helpful interactions); since the size of the search space grows exponentially with maximum plan length, we find that in the worst case, communication cost grows with an exponent equal to twice the maximum plan length [BC95b].

Multiagent planning research so far has mostly provided mechanisms for interaction [Geo86, vM92] without attention to this problem, although Gmytrasiewicz and Durfee [GD94] show a model in which agents use decision theory to decide when to communicate, and Durfee and Montgomery [DM91] show a method of reducing the cost of communication by sending abstract messages. Previous work on *social laws* [MT90, MT91], that is, constraints on what actions an agent may take, has shown a method for reducing communication and planning time by reducing the options an agent has at any point. Briggs and Cook [BC95a] show how to use successive relaxation of social laws to ensure that the social laws do not prevent a solution, while preserving the reduced costs in most cases.

Agents may avoid the exponential costs of communication during plan generation by partitioning the available resources and having each use only resources from its own partition during the entire plan. The clustering algorithm below performs such exclusive resource allocation in an intelligent manner, preferentially allocating resources that are likely to be used in achieving the agents' current goals. The communication cost is polynomial, requiring only that each agent report its goals and its abilities to whatever machine runs the algorithm, and that it be told what resources it may use. (This echoes the work of Yang [Yan92], who partitions domains within a single-agent planning system to reduce the branching factor in planning.)

At times there is *no* exclusive resource allocation scheme which will allow each agent to achieve its goals, and planning will fail. Because of this, we require a method for relaxing the exclusivity requirement. The cost for communication is lower in a domain in which fewer operators are allowed, because

* This research was supported in part by the National Science Foundation, under grant IRI-9308308.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/127 ©1996 FLAIRS

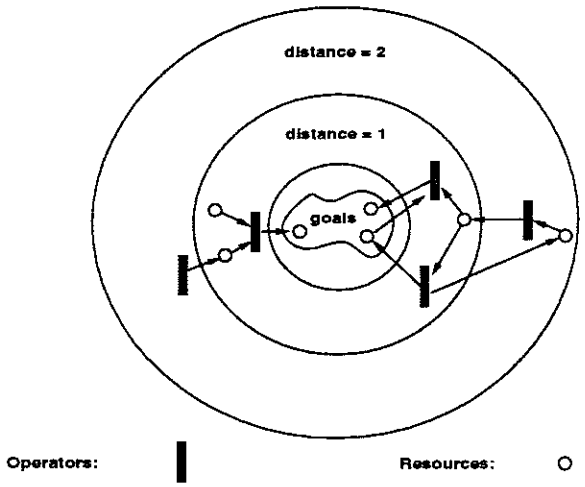


Figure 1: Distances of resources from the goal resources, defined as the shortest path via the links of the agent's operator set.

the search space is exponentially reduced; therefore we relax gradually, to avoid the cost of communicating without partitioning. (As with the initial allocation, overhead for the algorithm is polynomial and thus trivial compared to that of planning.) In the worst case, we will have to relax to the point of having no partitions. We will see that the overhead of failed planning attempts due to insufficient relaxation is justified.

For an example of both clustering and relaxation, consider robots working in a manufacturing domain. One robot is assigned the task of cutting a wooden plate and smoothing the edges, while another robot is to produce a metal plate with drilled holes. We would prefer to allow a sander or a rasp to the former, since these items are often used with wood; the other might be allowed metal-working tools. If the first robot requires a metal-working tool to complete its plan, or if some tools are used on both substances, we will need to relax the partition.

2 CLUSTERING

Here we define terms for the discussion to follow.

An *operator* has zero or more *preconditions*, which must be true if the operator is to execute, and one or more *preconditions* which it establishes. A *resource* is something an operator must have in some state in order to execute, for the duration of the operator.

An *agent* has a set of *goals* to establish, and a set of operators for establishing them. Planning is *deliberative*, that is, each agent develops a partially- or totally-ordered set of operators for achieving its goals,

Algorithm **Allocate-resources**
Given:
R, a set of resources;
A, a set of agents;
Return:
{ Resources and operators to be allowed: }
ResAllowed: array[agents] of set of resource;
OpsAllowed: array[agents] of set of operator;

{ Distances to resources and operators: }
ResDistance: array[agents, resources] of number;
OpDistance: array[agents, operators] of number;

{ Size of ResAllowed[agent]: }
ResNumber: array[agents] of number;

for $a \in A, r \in R$ do
run a shortest-path algorithm to find distances
and store into array *ResDistance*;

{ The operator's distance is the minimum
distance of all its resources: }

for $a \in A, op \in a$ do
$OpDistance[a, op] = \min_{r \in op} ResDistance[a, r]$;

{ The maximum distance of any operator,
as seen by any agent: }

$max\_distance = \max_{a \in A, op \in a} OpDistance[a, op]$;

for $d = 0$ to *max_distance* do
repeat
order A so that agents with fewer allowed
operators will come first;

for $a \in A$ in order do begin
select some operator from a with distance d,
so that no resource of the operator
is allowed to any other agent;

if such an operator exists then begin
append each resource of the operator
to *ResAllowed*[a];
append the operator to *OpsAllowed*[a];
end if
end for
until current iteration appended no operators;

for $a \in A$ do $ResNumber[a] = |ResAllowed[a]|$;

Figure 2: Algorithm **Allocate-resources**.

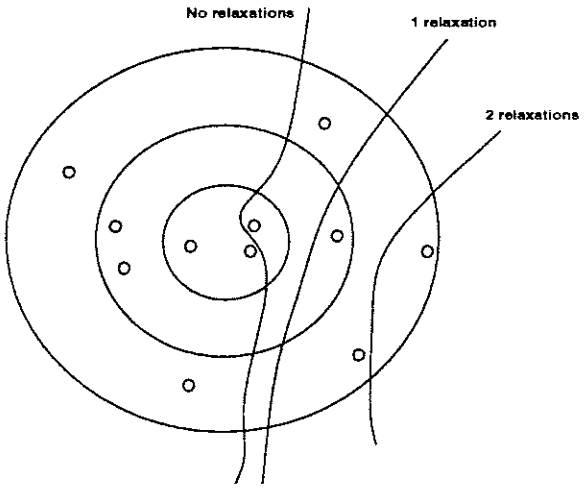


Figure 3: Subsequent relaxations in the allocation scheme allowing more resources to an agent. Resources to the left of a wavy line may be used by the agent after the given number of relaxations.

given the initial state of the world.

If operator A may establish a pre-condition for operator B, we say there is a *link* from A to B. The *distance* between any 2 resources, as seen by an agent, is defined as the length of the shortest path of links between the 2 resources, using only the agent's operators. (Obviously, distance is defined differently for different agents.) The distance between two operators is the maximum distance between any resource in one and any resource in the other. *Near* and *far* are defined accordingly. If there is no path between resources (or operators), the distance is infinity.

The distance of a resource, unless it is explicitly stated as distance from another, is assumed to be the distance to the nearest *goal resource*, i.e., the nearest resource which is required by some goal. The distance of an operator is the distance of its farthest resource. Figure 1 shows a set of goal resources, surrounded by operators which can affect their states, which in turn are connected to the resources they require. (Arrows connect preconditions to operators and operators to their postconditions.) There may exist multiple paths from a resource to the goal resource set; a shortest-path algorithm can provide us the resource's distance.

Operators (and the resources they use) are more likely to be useful if they are near to the goal resources, as they are more likely to be able to establish either the goals or nearby subgoals; therefore we should select first those resources which are near the goals. This principle forms the basis of our clustering method, outlined in Figure 2.

Algorithm **Relax-allocation**
Given: R, a set of resources;
A, a set of agents;
X, the maximum number of relaxations;
ResAllowed: array[agents] of set of resource;
ResDistance: array[agents, resources] of number;
ResNumber: array[agents] of number;
Return: ResAllowed;

for $a \in A$ do
{ Select $1/X$ of the resources not initially allowed to this agent }
for iteration = 1 to $(|R| - ResNumber[a]/X)$ do
begin
select r from $R - ResAllowed[a]$;
append r to $ResAllowed[a]$;
$ResNumber[a]+ = 1$;
end for;

Figure 4: Algorithm **Relax-allocation**.

3 RELAXATION

If the exclusivity requirement does not afford a solution, we may opt for a more lenient allocation, that is, one which allows more resources to each agent; this is called *relaxation*. Figure 3 shows how each agent is allowed to plan with more resources after the allocation scheme is relaxed. Relaxation is gradual, such that after X relaxations the agent may use any resource; if we relax this far, we have degraded to planning with no partitioning of resources at all. The algorithm in Figure 4 describes this process.[1]

4 ANALYSIS

Given the maximum number of relaxations X and a maximum plan length L, we can derive the expected communication cost using this formula:

$$\widehat{Cost} = \sum_{\theta=0}^{X} SearchSpace(L, \theta) * P(\theta)$$

where $SearchSpace(L, \theta)$ is the average number of operators considered, in searching for a plan of length L or less, when there have been θ relaxations of the allocation scheme; and $P(\theta)$ is the probability that no solution was found before the θ^{th} relaxation.

[1] In the future, we would like to qualify the algorithm so that it preferentially selects resources with minimal distance, thus continuing to grow clusters from the goal resources, rather than merely allowing previously disallowed resources at random.

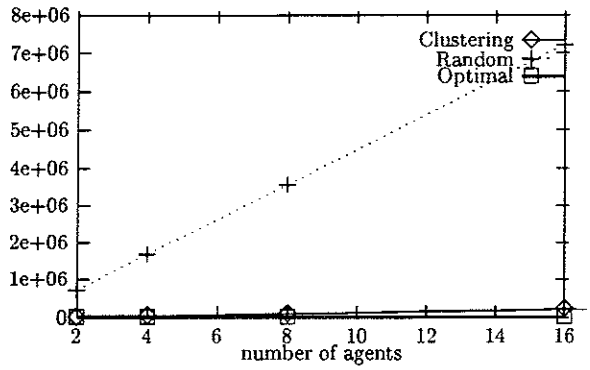


Figure 5: Changing expected communication costs using the clustering and relaxation method, compared to those of random and optimal resource allocation schemes. For these results, we assumed 32 resources, 2 goal resources and 16 operators per agent, 3 resources required per operator, and a plan length of 4; and we set X, the number of relaxations needed to allow each resource to each operator, to be 4. Communication cost is measured in terms of possible harmful interactions between operators considered by the agents as they search their planning spaces.

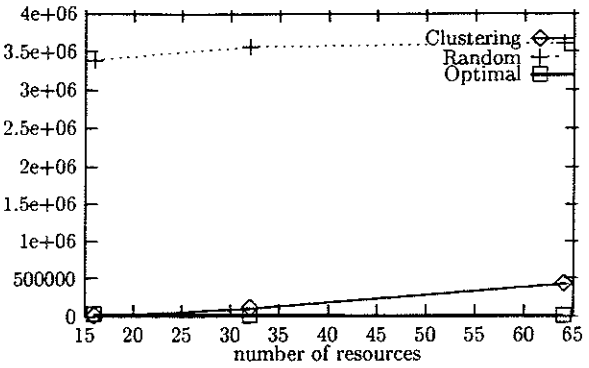


Figure 6: Communication cost, changing with the number of available resources.All parameters are the same as in Figure 5, except that we assume 8 agents.

The proofs involved in determining the value of $\widehat{Cost}$ are beyond the scope of this paper, and are provided in another publication [Bri96].

5 RESULTS

To measure the value of our approach, we considered a completely random domain, specified only in terms of number of operators, number of agents, number of resources, goal resources and operators per agent, and resources required per operator. (A random domain should have less inherent clustering than many existing domains, thus degrading the performance of the clustering method relative to more structured domains; further, it gives us domain-independent results.) We assumed a final plan length, and decided how many relaxations X should be required to allow each resource to each operator.

Figures 5 and 6 shows how the expected cost of communication per agent, measured in possible conflicts between one's own operators and those of another, changes with the number of agents and with the number of resources, respectively. *Clustering* denotes the method presented; the *random* allocation scheme allows resources to agents without regard to their goals; the *optimal* allocation is assumed to allocate precisely the needed resources before any others.

In both cases our algorithm does more poorly than optimal, of course, because there are attempts at planning with insufficient resources, and many resources which are not useful are nonetheless allocated and cause contention between agents who are allowed to use them (after some relaxation of the original allocation). It does much better than random in both cases. In Figure 5, the cost seems to vary linearly with the number of agents, at least within the region tested; for this test, the cost from the clustering method is a little under an order of magnitude that of the random method. Note that the large communication costs significantly outweigh the cost the allocation and relaxation algorithms (which require only a sharing of knowledge about operators, goal resources, and resource allocations).

The method, therefore, can be expected to provide effective resource allocations automatically and thus reduce the cost of multiagent communication for planning, without the need for human intervention.

6 FUTURE WORK

In coming months, we would like to make the relaxation algorithm use distance as a criterion, as the allocation algorithm does, so as to bring the benefits of clustering into the relaxation process, and to require both allocation and relaxation to allocate disputed resources not merely fairly, but in such a way as to maximize the number of allowed operators over all agents. Testing the behavior of the method as we vary the number of relaxations possible is needed, to find which values are best. Also useful will be testing of the method on real-world domains such as manufacturing, and information gathering on the Internet.

References

[BC95a] Will Briggs and Diane J. Cook. Flexible social laws. In *Proceedings of IJCAI-95*. IJCAI, 1995.

[BC95b] Will Briggs and Diane J. Cook. Local planning and teamwork: minimizing communication in multiagent domains. Technical Report TR-CSE-95-001, University of Texas at Arlington, 1995.

[Bri96] Will Briggs. *Modularity and Communication in Multi-Agent Planning*. PhD thesis, University of Texas at Arlington, 1996.

[Coo94] Diane J. Cook. Reconfiguration of multiagent planning systems. In *Proceedings of AIPS-94*, pages 225–230. AIPS, 1994.

[DM91] Edmund H. Durfee and Thomas A. Montgomery. Coordination as distributed search in a hierarchical behavior space. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1363–1378, Nov/Dec 1991.

[GD94] Piotr J. Gmytrasiewicz and Edmund H. Durfee. Rational coordination and communication in multiagent environments through recursive modeling. Submitted to *Journal of AI Research*, 1994.

[Geo86] Michael P. Georgeff. Representation of events in multiagent domains. In *Proceedings of AAAI-86*, pages 70–75. AAAI, 1986.

[MT90] Yoram Moses and M. Tennenholtz. Artificial social systems part i: basic principles. Technical Report CS90-12, Weizmann Institute, 1990.

[MT91] Yoram Moses and M. Tennenholtz. On formal aspects of artificial social systems. Technical Report CS91-01, Weizmann Institute, 1991.

[vM92] F. von Martial. Coordinating Plans of Autonomous Agents. Springer-Verlag, 1992.

[Yan92] Qiang Yang. A theory of conflict resolution in planning. *Artificial Intelligence*, 58:361–392, 1992.

PHYSICAL IMPLEMENTATION OF PERFORMING AGENTS

Karl R. Wurst and Robert McCartney
Department of Computer Science and Engineering, U-155
University of Connecticut
Storrs, CT 06269-3155
E-mail: wurst@cse.uconn.edu

Abstract

Performing agents are agents designed to entertain an audience through performance. Most computer-based performing agents have been virtual screen-based actors. We are building physical, computer-based performing agents by combining techniques from artificial intelligence, robotics and puppetry. Our agents are autonomous individuals, each with its own processor on board, that interact with each other, communicating on a low-bandwidth infrared system. The issues we address are those of communication between the agents, between the agents and the audience, and between the agents and an external director.

1 INTRODUCTION

Performing agents are agents designed to entertain an audience through performance. Traditional physical performing agents are human actors and puppets, but performing agents may be realized in many media – video, audio, text and others, singly or in combination. In most current AI research, these performing agents are on screen or video with no physical presence. Our work is closer to the traditional view of performing agents, having physical puppet actors on stage, and uses techniques and materials from the craft of puppetry as well as the field of autonomous robotics, a mixture that we term *puppotics*. While this research is being done in an engineering school, it draws heavily on the resources and expertise of the University of Connecticut's world-renowned Puppet Arts Program.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/132

2 WHY PHYSICAL PERFORMING AGENTS?

We are building physical performing agents so that we can examine the interaction between agents. We wish to examine the different forms that this interaction may take, and the difficulties involved in implementing this interaction in physical form. By working with physical implementations, we are making explicit many of the assumptions inherent in the software-only research in this area.

Also, our agents are physical because we believe that they will be more entertaining and engaging than virtual agents. Audiences have a more immediate rapport with a puppet than they do with an animation because of the physical presence of the puppet.

3 THE WOGGLES

Our agents are based on the Woggles of the Oz Project at CMU[Loyall and Bates, 1993]. The Woggles are video actors, depicted as spheres, that exist in a video landscape. They move through this landscape and interact with each other and with a human-controlled woggle. They communicate with each other by gesturing. The Woggles are also in use at Stanford in the Virtual Theater project[Hayes-Roth, 1995].

The Woggles choose their actions based on their internal emotional and physical states, their personalities, and interaction with each other. Each woggle has a distinct personality that affects how it will go about attempting to satisfying its goals. In the CMU Woggle world, *The Edge of Intention,* there are three woggles – Wolf, Shrimp and Bear. Wolf and Shrimp have aggressive and shy personalities respectively. Each may have the goal off amusing himself, but the differences in their personalities will cause them to satisfy this goal differently. Shrimp may decide to go dance around by himself to amuse himself, while Wolf may

decide to go pick on Shrimp.

Their performances are unstructured, that is, there is no script or plot that they follow in their performance, and are entirely interaction-driven. They communicate entirely by making physical gestures and visually observing the gestures of others. Their repertoire of gestures includes "Say Hey" – a greeting gesture, squash down, puff up, threaten and spin. This limited number of gestures leads to simple interactions – woggles can gesture at each other, and move toward or away from other woggles. There is no external direction of their actions at this time.

Our goal for our physical agents, the RoboWoggles, is to reproduce the video version of the Woggles as closely as possible with physical devices. We wish to duplicate the behaviors as well as mimic the visual aspects of the original Woggles as much as possible.

The fact that there is a behavioral model already developed for the Woggles, with appropriate gestures for communication makes them attractive as a starting point for our research. Their simplicity, both in processing and in physical design and range of motion, makes them suitable for physical agents.

4 COMMUNICATIONS ISSUES

There are three types of communications that can take place with performing agents – agent-to-agent, director-to-agent, and agent-to-audience. These communication issues are not limited to performing agents, but will manifest themselves in any multi-agent setting (except for possibly the last – agent-to-audience communication.). A team of robots cleaning up an oil spill, for example, would have to communicate to coordinate their efforts, and may receive instructions from a supervisory agent that can see the overall picture.

4.1 Agent to Agent

Communication between the agents themselves can take the form of point-to-point dialog between two agents, or a broadcast from one agent to a group. In our case, the difference is largely in the intent of the agent. Since all communication is performed by gesturing, all messages are broadcast to any agent that is within visual distance. Some messages may be intended for anyone who can see it, but more often it is intended for the agent that the sender is facing. It is up to the receiver to determine if the message was intended for it or not, and what to do with the information received. One could imagine an agent "eavesdropping" on a conversation between other agents and using the information to change its own internal state in some way – becoming angry perhaps.

4.2 Director to Agent

In addition, there can be communication between an external director and the performer. An example of this is what Hayes-Roth calls *directed improvisation*[Hayes-Roth *et al.*, 1994]. The agents are given high-level, abstract instructions by the director, and improvise the specifics. This direction can come at the beginning, in the form of a plot to be followed, or can be interjected by the director throughout the performance, as a way of keeping the story on track[Kelso *et al.*, 1992] These directions from the external director would likely add to or change the agent's goals, to force the agent's performance to come into line with the director's overall plan. Other forms of director to agent communication can be imagined, such as a supervisor directing working robots to concentrate their efforts on a particular area of a problem.

There also exists the possibility of agent to director communication. This would allow the agents to inform the director of their internal state (possibly as a response to a query from the director,) or to request new instructions if they get stuck.

4.3 Agent to Audience

Finally, there is the communication between the performer and the audience to consider, and how the performing agent can get his message across. There must be a common language that the audience and the performer both understand with conventions for certain concepts. This is one of the areas where we are drawing on the expertise of the puppetry field, where the issues of communication with the audience, often with non-verbal agents with limited physical expression, is explored.

It may also be useful to have audience to agent communication be possible. Possible uses for this would be to allow the agents to take a bow to applause, respond to a heckler, or even to adjust its timing to audience response.

5 IMPLEMENTING WOGGLES

Our physical implementation matches the ideal quite well as far as gestures are concerned, and adds individual, internal control to each woggle. The body is a ten inch diameter flexible sphere that is mechanically deformable. This allows the characteristic woggle gestures of puff up, squash down and so on. This

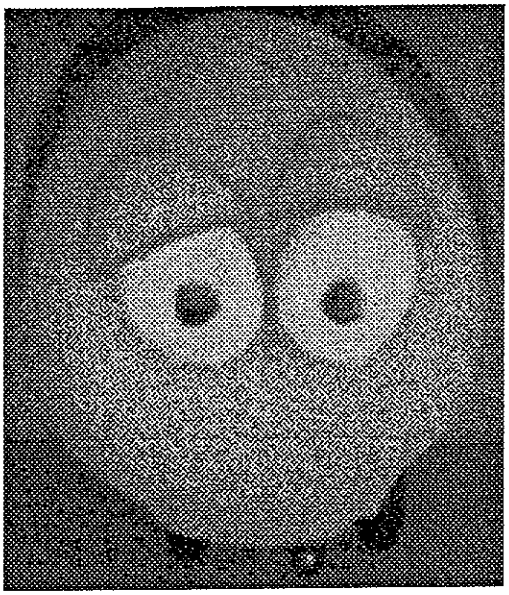


Figure 1: Prototype RoboWoggle

is accomplished with two servomotors connected to an armature that can deform the body at the top and sides (see Figure 2.)

The woggle is controlled by a system based on the Motorola 68HC11 microcontroller. The system is based on Marvin Green's BOTBoard, modified to provide an additional 32K of memory. It can control 4 servomotors, and has a number of analog and digital inputs, and digital outputs.

The most notable departure from the video woggles is the RoboWoggles' method of propulsion. Rather than attempt the potentially very difficult mechanical engineering problem of producing jumping behavior, we opted instead for having the RoboWoggles roll about on two wheels. The two wheels are differentially driven, allowing the woggles to spin in place as well as move around on the floor.

The woggles' communication is visual: one agent makes a physical gesture, the other observes it. In a "traditional" vision system a camera or similar sensor would capture the complete scene before the agent. A vision processing system would then determine which elements of the scene should be grouped together to form individual objects or agents. Finally, the system must watch the changes in pose of an agent over time to determine what gesture it is performing.

We propose a simpler, but functionally equivalent vision system that eliminates the need for most of the visual input processing. In our "vision-less" vision system, each agent simultaneously gestures and transmits information about the gesture to all agents around him. This vision system should be indistinguishable from a "traditional" vision system from the audience's point of view. It still involves active sensing and processing of visual information to determine what another agent is doing.

Communication between woggles is performed by a line-of-sight, low-bandwidth infrared communication system. Transmission of gestures is broadcast 360° by four transmitters (each covering 90°.) Each transmitter broadcasts the identity of the woggle performing the gesture and the position of the transmitter itself (front, back, right or left) as well as the gesture. This allows a receiving woggle to determine the identity and facing of the transmitting woggle, as well as turning or spinning behavior (by observing the transmitters' signals over time.)

Reception is from the front of the receiver only. In other words, a woggle can only see what another woggle is doing if it is looking at (facing) it, but a woggle can see what other woggles are doing, even from behind them.

6 STATE OF IMPLEMENTATION

At this time, the prototype microcontroller system is working. The base mechanics with the drive and steering mechanism are completed as well as the body deformation mechanism. A prototype body was made from reticulated foam (a material commonly used for puppet bodies – including the Muppets™.) Unfortunately, it did not have quite the flexing characteristics hoped for, and the amount of sewing necessary to produce it made it prohibitive for making multiple bodies.

7 FUTURE WORK

The communication system is still in the design stages. A simple infrared scheme (as in consumer electronics remote control systems) is sufficient to transmit the gesture "vocabulary" needed to mimic vision, but we need to develop communication protocols that can be extended to support broadcast communications and reception of external direction. This extended protocol will need to be able to communicate new goals, change the priorities of goals for an agent, or give explicit instructions, and may implemented using different communication hardware.

Another group in our laboratory is developing a general, expandable, modular robot controller system that will eventually replace the prototype system now in use.

A new body will be cast from neoprene, and the casting process will allow multiple bodies to be produced easily.

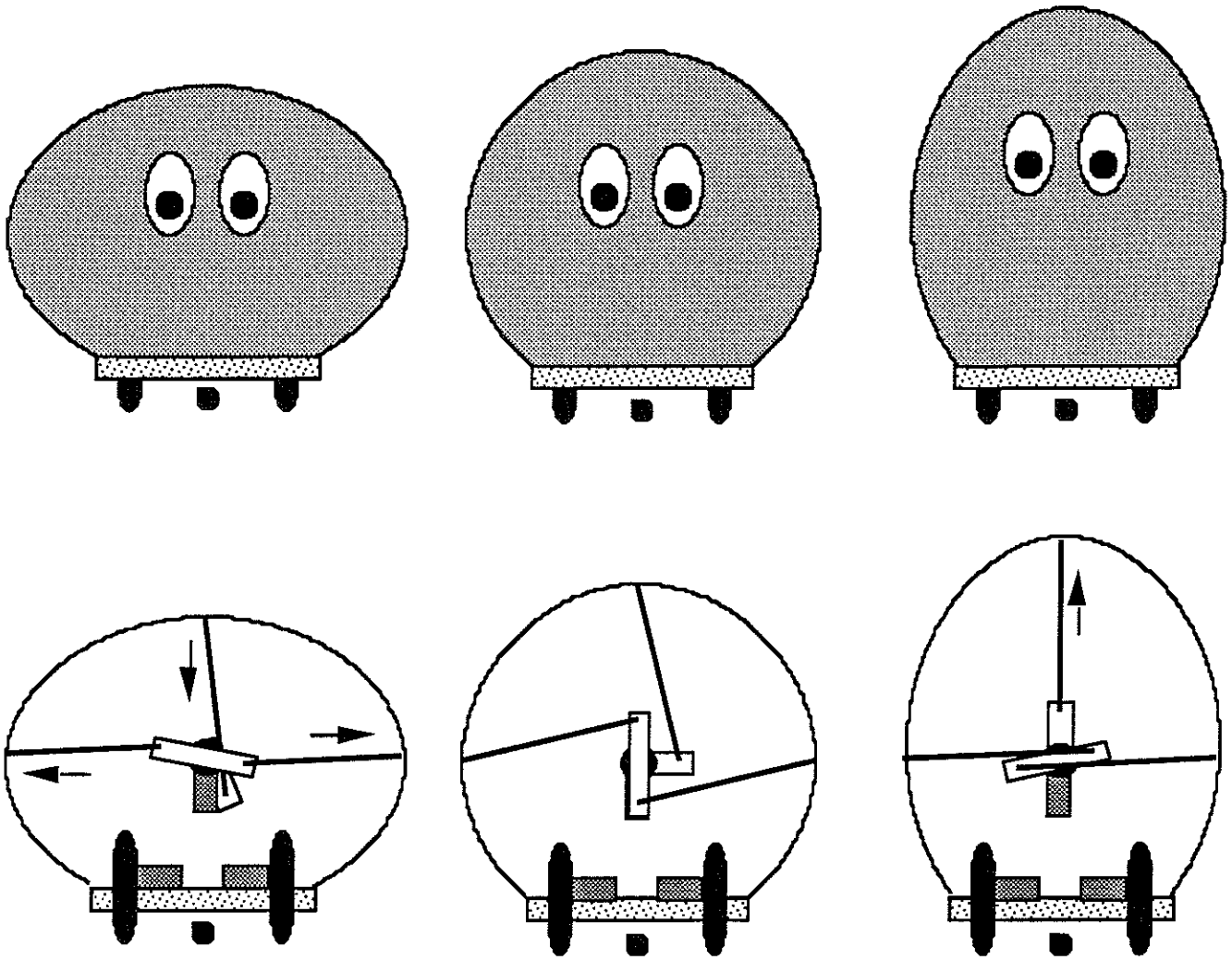


Figure 2: Deformation and locomotion mechanics (squash, normal, and "Say hey")

We plan to look into structuring the performances by providing some type of plot mechanism. This will require either some sort of script to follow, or the implementation of external direction to keep the improvisations from getting out of control and keep the overall performance on track

8 RELATED WORK

The Oz Project at CMU[Bates, 1992] is developing tools to support virtual worlds with believable agents that humans can interact with. The Woggles and their world, *The Edge of Intention* are just one of the systems that they are building with these tools. They have developed an integrated agent architecture called Tok[Bates *et al.*, 1992b] which includes a reactive component, Hap[Loyall and Bates, 1991], an emotion system, Em[Reilly and Bates, 1992], and a natural language system, Glinda. A major thrust of their work is external direction with the goal of being able to immerse a human in a complete dramatic work. They want the system to be able to improvise responses to the human's actions, while still maintaining the overall structure and goals of the plot.

The Virtual Theater project at Stanford[Hayes-Roth *et al.*, 1995] is also concerned with *directed improvisation,* but as a more general paradigm for human-computer interaction. Their system, the Virtual Theater, uses the Woggles as an environment for children to write and perform plays. The children may write out detailed scripts for the characters to perform, create abstract stories for the characters to improvise, or directly control the characters as video puppets. The children can mix these modes, and in fact, switch between them at any time.

The ALIVE System at the MIT Media Lab[Maes

et al., 1996] presents a virtual world populated by autonomous agents with their own goals and intentions, modeled with their Hamsterdam[Blumberg, 1994] agent toolkit. It has a novel interface that allows use of the user's entire body to interact with the agents. It does this by projecting a video image of the user onto a large projection screen, placing the user into the virtual world. The user sees himself interacting with the agents and objects he sees. The system interprets the user's actions through video processing that identifies the user's gestures.

9 CONCLUSIONS

We are building physical implementations of performing agents whose main purpose is to provide entertainment for an audience. The ROBOWOGGLES are an example of *puppotics:* a combination of artificial intelligence, robotics and puppetry technologies. They communicate with each other by physical gestures, and communicate those gestures over an infrared communication system, while performing the actual gesture for the audience. These simple performing agents provide a good starting point for exploring performances with more complex plots and interactions and the possibility of external direction of the agents. In addition, we are exploring the issues involved in the communication among the agents and between the agents and an external director. The communication issues that must be addressed are not limited to performing agents, but will manifest themselves in any setting where multiple agents are working together.

References

[Bates *et al.*, 1991] Joseph Bates, A. Bryan Loyall, and W. Scott Reilly. Broad agents. In *Proceedings of the AAAI Spring Symposium on Integrated Intelligent Architectures*, Stanford, CA, 1991. Available in *SIGART Bulletin*, Volume 2, Number 4, August 1991, pp. 38-40.

[Bates *et al.*, 1992a] Joseph Bates, A. Bryan Loyall, and W. Scott Reilly. An architecture for action, emotion, and social behavior. In *Proceedings of the Fourth European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, S. Martina al Cimino, Italy, 1992.

[Bates *et al.*, 1992b] Joseph Bates, A. Bryan Loyall, and W. Scott Reilly. Integrating reactivity, goals, and emotion in a broad agent. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, Bloomington, IN, 1992.

[Bates, 1992] Joseph Bates. The nature of characters in interactive worlds and the Oz Project. Technical Report CMU-CS-92-200, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, October 1992.

[Blumberg, 1994] Bruce Blumberg. Action-selection in Hamsterdam: lessons from ethology. In *Proceedings of the Third International Conference on the Simulation of Adaptive Behavior*, Brighton, August 1994.

[Hayes-Roth *et al.*, 1994] Barbara Hayes-Roth, Erik Sincoff, Lee Brownston, Ruth Huard, and Brian Lent. Directed improvisation. Technical Report KSL-94-61, Knowledge Systems Laboratory, Stanford University, Palo Alto, CA, September 1994.

[Hayes-Roth *et al.*, 1995] Barbara Hayes-Roth, Lee Brownston, and Erik Sincoff. Directed improvisation by computer characters. Technical Report KSL-95-04, Knowledge Systems Laboratory, Stanford University, Palo Alto, CA, January 1995.

[Hayes-Roth, 1995] Barbara Hayes-Roth. Agents on stage: Advancing the state of the art of AI. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, August 1995.

[Kelso *et al.*, 1992] Margaret Thomas Kelso, Peter Weyhrauch, and Joseph Bates. Dramatic presence. Technical Report CMU-CS-92-195, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, December 1992.

[Loyall and Bates, 1991] A. Bryan Loyall and Joseph Bates. Hap: A reactive, adaptive architecture for agents. Technical Report CMU-CS-91-147, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, June 1991.

[Loyall and Bates, 1993] A. Bryan Loyall and Joseph Bates. Real-time control of animated broad agents. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, Boulder, CO, June 1993.

[Maes *et al.*, 1996] Pattie Maes, Trevor Darrell, Bruce Blumberg, and Alex Pentland. The ALIVE System: wireless, full-body interaction with autonomous agents. *ACM Multimedia Systems*, Spring 1996. To be published.

[Reilly and Bates, 1992] W. Scott Reilly and Joseph Bates. Building emotional agents. Technical Report CMU-CS-92-143, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, May 1992.

A SYSTEM FOR AUTOMATIC VISUAL INSPECTION OF VLSI MICROCIRCUITS

Dan E. Tamir and Herold T. Tong

Department of Computer Science
Florida Institute of Technology
150 W. University Blvd.
Melbourne Florida 32901

Abstract

This paper summarizes a research project to explore the possibility of automating the process of visual inspection of electronic microcircuits. The inspection system is implemented through integration of digital image-processing algorithms with expert system techniques. It contains three major components organized in three layers. The upper layer is a rule-based expert system, the intermediate layer is a control unit, and the lower layer is an image processing unit. A prototype of this system has been tested with respect to detection of metallization defects in the bonding pads of VLSI microcircuits. The results of these tests show that the system has high potential for commercial applications.

1 INTRODUCTION

Industrial inspection is an area in which the application of machine vision technology shows exceptional promise. Intense competition and stringent product reliability requirements often make inspection the single most important and sometimes the most expensive stage in a production process. Much of this expense is due to the labor intensive nature of most inspection tasks.

However, currently there are no commercial or experimental systems capable of reliable and cost-effective automatic visual inspection of microcircuits.

The research project reported in this paper is supported by Chip Supply Inc., Orlando, Florida. The goal is to explore the possibility of automating the process of visual inspection. Currently, the process of manual visual inspection of electronic microcircuits requires many well trained human inspectors trying to identify mechanical defects in the microcircuits using sophisticated and expensive optical equipment. It includes checking incoming and outgoing materials for construction and workmanship compliance with the requirements of an applicable acquisition document.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/137 ©1996 FLAIRS

1.1 Automatic Inspection of Electronic Microcircuits

Several mechanical defects can occur in electronic microcircuits during the process of production, storage, handling, and testing of the circuits. Many of the defects exhibit changes in the shape or color of the active circuit areas in the microcircuit. Most of these defects can be detected through visual inspection with high-magnification devices. Automating the inspection operation or even a part of the operation can be cost effective. It can reduce the load imposed on the human inspector and increase the reliability of the inspection process while saving expensive labor cost.

Many active circuit areas need to be checked during the inspection process. The bonding pad, used to connect the microcircuits of the chip to inputs and outputs from external devices, has been selected as the subject of this research.

One of the steps of chip production is functional examination. During the functional examination, wires are attached to the bonding pads and later removed from the bonding pads. This process can cause metallization scratches and voids.

The research reported in this paper focuses on defects of metallization scratches, voids, and corrosion, on the bonding pads. Currently, the automatic visual inspection system performs the above inspection operations on the integrated circuit chip FairChild-6792. The inspection operations examine the following metallization defects:

(1) Metallization scratches, that is, any tearing including probe marks in the surface of the metallization [1].

(2) Metallization voids, that is, any region in the metallization where the underlying metal or passivation is visible [1].

(3) Metallization corrosion, that is, corrosion located on the surface of the metallization [1].

The automatic inspection process is accomplished through the use of an intelligent computer system integrating an expert system with digital image processing operations.

1.2 System Overview

The system is modular and implemented through integration of digital image processing algorithms with a control unit and an expert system. The inspection system contains three major components organized in three layers. The lower layer includes the image processing unit. The next layer is the control unit and the upper layer includes a rule-based expert system.

The image-processing unit performs several basic image processing operations such as image acquisition, image segmentation, and image registration.

The control unit serves as a coordinator between the image-processing unit and the expert system. It contains information about the specific circuit under inspection. It determines the sequence of image processing operations and passes the results of the image processing unit to the expert system to determine the status of the chip under inspection.

The rule-based expert system contains the specifications of the manual visual inspection (the acquisition document). It emulates knowledge and decision making procedures used by the human inspector and makes a decision about the status of a microcircuit on the basis of the data supplied by the image-processing unit through the control unit.

The image processing unit and the expert system are "chip independent" and highly modular. The image processing unit performs basic image processing operations and the expert system encodes general human knowledge obtained from the inspection manual. On the other hand, the control unit is "chip dependent". It contains information specific for the chip under inspection. Hence, the control unit is the only module of the system that has to be updated with every new type of circuit for inspection.

Due to space shortage, this paper concentrates on the image processing unit and the control unit. Details concerning the expert system and the set of experiments can be found in [2].

2 IMAGE PROCESSING FOR VISUAL INSPECTION

Generally, mechanical defects present a change in the color and/or texture of an area in the chip. A chip verified by manual inspection to be free of defects is serving as reference. The strategy of the inspection process is to compare the inspected chip to the reference chip, detect areas of difference, and check if these areas present an evidence of a defect. The image processing unit is responsible for performing the following image processing operations:

(1) Image Acquisition

(2) Image Segmentation

(3) Color Separation

(4) Segment Features Extraction

(5) Image Registration

(6) Difference Image Acquisition

The above operations are performed on the reference chip and the inspected chip.

2.1 Image Acquisition

The goal of the image acquisition stage is to obtain a digitized image of the chip. It is implemented using a CCD camera hooked to a microscope. A digitalization board provides 256 gray levels. Three sets of images of each chip (red, green, and blue) are obtained by using appropriate filters on the camera.

2.2 Texture Segmentation

Image segmentation is the partition of an image into a set of contiguous non-overlapping homogeneous regions. In this work, texture segmentation is implemented through clustering and connected component labeling [3,4]. Clustering of texture features of the image pixels is used as a mean of identifying homogenous regions. These features include brightness, 3x3 mode of brightness, 5x5 mode of brightens, and 7x7 mode of brightness. A connected component labeling (CCL) algorithm follows the clustering. The CCL assigns a unique label to each region. After texture segmentation, each region is represented by the average gray level and the set of pixels of that region. According to our notation, an image segment, an image region, and an image object are synonyms.

2.3 Color Separation

Color separation is implemented through image segmentation using color features such as brightens of the red, green, and blue (RGB) components, and mode filters of the RGB components. The results of the color separation

are represented in a similar way to the results of the texture segmentation. The color (RGB) of a color homogeneous region and its pixels are recorded. The texture homogeneous regions are different from color homogeneous regions. Generally, a texture region includes one or more color regions. The results of the texture segmentation and color separation are used in the following stages of the image processing unit. In addition, they are used by the control unit to generate the input data for the expert system

2.4 Segment Features Extraction

The segment features extraction procedure extracts features from each texture and color segment in the image of the microcircuit and uses these features for the representation of the segment.

In the first step of feature extraction, a boundary following algorithm extracts the coordinates of the boundary points of each region [5]. Next, the following features are derived: 1) the chain code of the segment boundary pixels [6], 2) the average gray level or color of the segment, 3) the center of segment, 4) the area of the segment, 5) the length code representation of the segment [7], and 6) the variance of the length code of a segment (surface "roughness" measure).

Let S_k be the *k'th* segment, and let $\vec{P}$ be the vector representing the *(i,j)* coordinates of a pixel of that region, then, $\vec{C}_k$, the vector representing the center of the segment, is obtained using the formula:

$$\vec{C}_k = \sum_{\vec{P} \in S_k} \vec{P}$$

The area of a region is approximated by the number of pixels in the region. The length code representation of a region is the set of distances measured from the region center to each of the boundary pixels, starting from the upper-left-most boundary pixel. This representation is translational invariant.

The measure of surface "roughness" is used to distinguish between voids and scratches. Generally, a void in a functional unit leaves a well shaped boundary such as straight lines in neighbor units. A scratch, however, produces regions with "rough" boundaries. The variance of the length code of a region is used to determine the roughness of the region. A region with high variance in the length code is considered to be rough.

2.5 Image Registration

The image registration stage aligns the inspected image with the reference image. The accuracy of the image registration operation is vary important for matching and locating the bonding pads on the inspected chip. In this work, registration is performed using length code of regions[7].

The length code of a region is translational invariant. The cyclic auto-correlation of a length code set is both translational and rotational invariant. Our algorithm uses features obtained from the cyclic auto-correlation of segments to match segments from the reference image with segments from the inspected image. The centers of the best matches are used to calculate the rotation and translation that has to be applied to the inspected image in order to rectify this image with the reference image.

2.6 Difference Image Acquisition

After alignment, a difference image is created from the reference image and the inspected image. Let $R_{i,j}$ be the *(i,j)* pixel from the reference image, and $I_{i,j}$ be the *(i,j)* pixel from the inspected image, then $D_{i,j}$ the *(i,j)* pixel of the difference image is obtained using the formula:

$$D_{i,j} = |R_{i,j} - I_{i,j}|$$

In addition, the image processing unit obtains the mean square error (MSE) for the entire image and for each segment. The MSE is obtained through the integration of gray values of pixels in the difference image. For example, let S_j denote the *j'th* segment of the reference image. The MSE of S_j is obtained using the formula:

$$MSE_j = \sum_{D_{i,j} \in S_{i,j}} (D_{i,j})^2$$

The MSE is used by the control unit for rough yet fast assessment of defects. if the MSE of an image region is below a given threshold then the control unit declares the region as free of defects and terminates the inspection of that region. Otherwise, the detailed inspection process is performed on the each individual region.

3 THE CONTROL UNIT

The control unit serves as a medium between the image processing unit and the expert system. It has circuit specific knowledge, referred to as the model, which is used to direct the image processing unit. The results of the image processing are filtered, formatted by the control unit and transmitted to the expert system.

The inspection process includes two main stages; an off-line learning stage, and an on-line inspection stage. The input for the off-line stage is internal representation of the circuit (the model) and an image of a defect free chip. This image is referred to as the reference image. The out-

put from the learning stage is saved in the control unit using internal representation and in the fact-base of the expert system.

The input for the on-line stage is the image of an inspected chip (the inspected image), the reference image, and the internal representation of the data obtained from the reference image during the learning stage. The output of this stage is formatted and transferred to the expert system. Both stages involve the following operations:

(1) Application of the Image Processing unit,

(2) Unit and Sub-unit Identification,

(3) Data Formatting and Transmission

3.1 Application of the Image Processing Unit,

This phase involves obtaining the reference/ inspected image, texture segmentation of the image, color separation of the image, extraction of the features of regions identified in the image, registration of model coordinates with the reference image coordinates (off-line) or of the reference image with the inspected image (on-line), and obtaining the difference image (on-line).

Before proceeding with the on-line stage, the control unit compares the minimum square error (MSE) obtained from the difference image with a given threshold. If the MSE is below that threshold, then the inspected chip is declared as acceptable and the process of inspection is terminated. Otherwise, the control unit proceeds with the on-line stage.

The results of the image processing phase include the set of objects identified in each image. Each object is represented by its average gray level or color, the area of the object, the surface roughness measure of the object (variance of the length code), and the coordinates of the object's center.

3.2 Unit and Sub-unit Identification Procedure

In the off-line stage, the unit identification procedure matches functional units in the model (the circuit specific knowledge) with objects in the reference image. In the on-line stage, this procedure is applied to the image coordinates of the reference and the inspected images. In Both cases a nearest neighbor procedure is used. Sub-units, such as the passivation layer of the bonding pads, are identified in the same way.

For example, in the on-line stage, the set of coordinates of the centers of every region in the reference image and in the inspected image are obtained. Next, the distance between every pair of centers is computed. Let $d_{i,j}$ denote the distance between the center of region i in the reference image and the center of region j in the inspected image, then region l in the inspected image is mapped into region k in the reference image if $d_{k,l} \leq d_{k,m}$ for every region S_m ($m \neq l$) in the inspected image.

This research concentrates on the bonding pads. Hence, at the end of the unit/sub-unit identification stage, each bonding pad on the reference/inspected image is represented by the average gray level of the top layer of the pad, the colors of the top layer, the colors of the unglassivation layer, the colors of the underlying passivation layer of the pad, the coordinates of the pad's center, and the chain code representation of the pixels of the pad's boundary.

In the on-line stage, the next phase (data formatting and transmission) depends on data obtained from the difference image. For each bonding pad in the inspected image, the control unit compares the minimum square error (MSE) of that pad, obtained from the difference image, with a given threshold. If the MSE is below that threshold, then this pad is discarded from further inspection.

3.3 Data Formatting and Transmission

The input data produced for the expert system has to have a specific format. Formatting the data obtained from the image processing unit and transmission of the data to the fact -base or the black-board of the expert system is the responsibility of the control unit. This is done in the data formatting and transmission phase.

Most defects caused by metallization scratches and voids change the color of the top layer and the shape of the bonding pad. Using the results of the image processing stage, the control unit produces data to denote whether there is a significant difference between the color or the shape of a region in the reference image to the color or shape of that region in the inspected image. For example, the area of respective regions is used to produce input data for the expert system. This data can have the form: "the area of the void is less_than 25% of the area of the pad".

Metal damages caused by scratches and voids have very similar appearances on the image. Each type of damage, however, requires different inspection criteria. The method used to differentiate between the two is to compute the variance of the length code of the regions. Since the shape of the damage caused by the metallization scratches is rough, the value of the variance is higher than the value computed in cases of metallization voids. The control unit compares the variance to a threshold and produces input such as "the surface is rough", for use by the expert system.

The data formatted by the control unit is passed to the expert system. Data concerning the reference chip is transmitted to the fact-base. Data concerning the inspected chip, and differences between the reference and inspected images is transmitted to the black-board

4 THE EXPERT SYSTEM

Today, excellent and cost-effective tools for development of expert systems are available in the market and in the public domain. However, at the time we have started this project, cost-effective expert system shells and development tools were not available yet. Furthermore, commonly available expert system shells do not have good explanation mechanism. Moreover, these tools do not supply complete control over the implementation of the system. For these reasons, we have used a model for expert system shell, developed in our University, to implement the expert system shell for the inspection process [2].

4.1 Knowledge Acquisition

In the process of knowledge acquisition, the knowledge engineer is trying to determine from documents and directly from the experts the manner in which a specific problem is solved. The approach to the knowledge acquisition adopted in this research is attending lectures given by the inspection experts and interviewing the inspectors. The lectures are designed to train new inspectors. Through the lecture, we have learned the basic concepts of microcircuits development, the manual inspection process, the inspection criteria, and the symptoms of defects. The task of knowledge acquisition is often difficult. Most inspectors can not easily describe their expertise or the way they make decisions. Therefore, during the interviews we have presented different situations to the inspectors and have attempted to elicit the heuristic used to make a decision in that situation. Since the problem domain is to inspect the defects caused by metallization scratches, corrosion, and voids, the interviews concentrated on the issues related to these defects.

5 EXPERIMENTS AND RESULTS

The prototype visual inspection system has been examined several bonding pads of different FairChild-6792 chips. A defect free FairChild 6792 chip serves as the reference chip during the process.

The bonding pads have been inspected by a human inspector and by the system. We denote the case where the system classification agree with the human classification as correct classification. Let H_0, the null hypothesis of the inspection system, be: "The chip is defect free". From our experiments we conclude that the total error of the system is 20%, the error of type I (rejecting a defect free circuit) is 13%, and the error of type II (accepting a defective chip) is 7%. In addition, the confusion between scratches and corrosion is much larger than the confusion between voids and scratches.

The tests results give a relatively high total error, but almost acceptable error of type II. Taking into account that there are no commercial or experimental systems capable of reliable and cost-effective visual inspection of microcircuits, these test results are quite encouraging.

6 CONCLUSION

An automatic visual inspection system has been presented. The system includes an image processing unit, a circuit dependent control unit, and a rule-based expert system.

A prototype of this system, designed for the inspection of bonding pads, has been implemented and tested.

Analysis of the system classification errors reveals that most of them are due to problems in the segmentation and/or in the registration process. We conclude that an investigation of new algorithms for segmentation and registration is required in order to improve the performance of the system.

References

[1] Chip Supply, *Internal Visual Inspection Manual M-2010*, Chip Supply, Orlando, Florida, 1989

[2] Herold T. Tong, *A Rule-Based Expert System for Automatic Visual Inspection*, Florida Tech, Melbourne January 1995.

[3] Robert M. Haralick and Linda G. Shapiro, Computer and Robot Vision, Addison-Wesley, Reading, Mass., 1992

[4] Dan. E. Tamir, Chi-Yeon. Park and Wook-Sung Yoo, "Vector Quantization and Clustering: A Pyramid approach", *DCC Industrial Workshop*, Utah, March, 1995.

[5] Azriel Rosenfeld and Avinash. C. Kak, *Digital Picture Processing*, Academic Press, Orlando, Florida, 1982.

[6] H. Freeman, "Computer Processing in Line Drawing Images", *Computing Survey*, vol. 6, no. 1, March 1974: 57-98.

[7] Antony T. Lam, *Image Matching - A Comparative Study of Two Approaches*, Florida Tech, June 1992.

A Genetic Algorithm Approach to Identifying Roads in Satellite Images

Julian Eugene (Gene) Boggess
Computer Science Department, Mississippi State University
P. O. Box 9637, Mississippi State, MS 39762
(601) 325-2756 • gboggess@cs.msstate.edu

Abstract

Human investigators were observed while attempting to process satellite images for the purpose of extracting roads. Their techniques included devising a set of rules that helped them separate road from non-road pixels in the image. A Genetic Algorithm was used to simulate the process by which human informants constructed sets of rules and found better sets of such rules in a much shorter period of time.

1 THE PROBLEM

Data about much of the earth's surface features is now available in the form of satellite imagery. In particular, LandSat images, with 7 spectral bands (see Table 1) and a resolution of 30 meters per pixel (90 meters per pixel for the thermal band), have been available for public use for several years. Useful information may be extracted from this satellite data through the use of various automated techniques.

Band	Color
1	blue
2	green
3	red
4	near infrared
5	near/mid infrared
6	thermal infrared
7	mid infrared

Spectral bands of the LandSat Thematic Mapper sensor
Table 1

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/142 ©1996 FLAIRS

Many people, including commercial cartographers and the military, are interested in identifying roads in satellite images. Although human beings are generally quite proficient at extracting road networks from satellite images, this has proven to be a difficult task to automate. A number of different approaches to identifying roads in satellite images have been tried; good results have been obtained using backpropagation neural networks (Wolfer *et al.*, 1994; Boggess, 1994) and hybrid systems (Ko, 1995), but the process is time-consuming and tedious.

It was thought that it would be interesting to try to emulate the process whereby humans attempt to extract roads from satellite images, in order to see whether their techniques might be useful in automating the process. Therefore, the following exploratory procedure was carried out, with two goals: to emulate the technique used by two human investigators in their attempts to identify roads in a satellite image, and to automate and optimize the process as much as possible.

2 THE STUDY

To investigate how humans go about solving this problem, a study was conducted in which two human "experts" were observed while attempting to extract roads from a LandSat satellite image of Vicksburg, Mississippi. A single channel of the image with which they were working is displayed below in greyscale (256 possible values per pixel). (See Figure 1, which shows the area in the blue spectral band, and Figure 2, which shows the mid infrared .)

One method of identifying roads in LandSat images that both individuals were observed to employ involved examining a sample set of road and non-road pixels, using the multi-channel spectral information about the pixels to develop

a set of rules for determining whether a given pixel is a road pixel or not. In this technique, the spectral characteristics of known road pixels were examined, and a set of upper and lower bounds for these pixels in each of the spectral bands was obtained. A rule was established for each of the spectral bands, and the other pixels in the image were checked for compliance with these rules; pixels which obeyed all (or most) of the rules were classified as road pixels, while pixels which fell outside of the rule boundaries were classified as non-road pixels. The entire Vicksburg image was then processed using the generated set of rules. The investigators tended to use their intuitions to judge the resulting image as good or not, rather than compare actual numbers of correctly and incorrectly classified pixels, but their intuitions were surprisingly accurate.

The set of rules which an investigator settled on might be thought of as comprising an expert system of sorts, in which the rules embodied the knowledge of the investigator regarding which combination of upper and lower boundaries in each of the channels could most effectively serve to discriminate road and non-road pixels.

Because of variations in illumination, contrast, and intrinsic spectral reflectance characteristics, road pixels vary widely in their spectral values. Consequently, a simple set of rules will not be very useful, for if it is to include all road pixels, then it will also include in error many pixels which are not roads. Thus, the investigators found themselves varying first the upper bounds, then the lower bounds, of this rule, then that rule, separately and in combination, in hopes of discovering the optimum combinations of rules which would maximize the number of road pixels identified as roads, while minimizing the number of misidentified non-road pixels. This turned out to be rather frustrating and time consuming.

After establishing a set of rules in the manner described above, one of the investigators generated the image in Figure 3 below. In this image, the pixels following all of the rules are displayed in black, pixels which follow most of the rules are displayed in gray, and the rest of the pixels are displayed in white. It is evident that this technique, while not perfect, is quite useful in eliminating many of the non-road pixels, while preserving most of the road pixels. If this approach could be speeded up and regularized, it could be employed as a quick and dirty initial procedure, serving as a starting point for more sophisticated additional investigative techniques.

For this study, a set of 600 pixels was selected, half road and half non-road. Only 600 pixels were utilized due to the difficulty in manually selecting accurate and evenly-distributed samples of the various types of roads and different terrain types. In addition, one of the purposes of this study was to see what kind of results could be obtained with the investment of as little time as possible.

The spectral values for these pixels were obtained from six of the seven bands; the thermal band was eliminated from consideration due to its poor resolution (120 meters per pixel) compared to the other sensors. It was hypothesized that a lower bound and an upper bound could be established for each of these bands, below and above which no road pixels occurred in the data sample. A function or set of rules could be derived from these boundaries such that a pixel whose spectral reflectance values fell between the lower and upper bounds for all six bands would have a high probability of being a road pixel, while one whose spectral values fell outside of these boundaries would be less likely to be a road pixel.

However, due to variations in road composition, road width, spectral reflectivity, the surrounding context, and other factors, the upper and lower bounds derived from direct inspection of the 300 road pixels does not produce a usable set of rules; an unacceptable number of mistakes tend to be made in misclassifying non-road pixels. Since each pixel could have a spectral reflectance value between 0 and 255 in each of the 6 spectral bands, and since an upper and a lower bound must be established for each band, it would require on the order of $((2^7)^6)^2$, or 2^{84}, operations to examine every possible set of rules in order to find the best one. Clearly, this would be an impractical task.

3 THE GENETIC ALGORITHM APPROACH

The genetic algorithm (GA) approach is a machine learning technique which uses an evolutionary approach to find a solution to a problem. It employs operations of selection, crossover, and mutation to improve the performance of a

population of possible solutions. The GA uses an evaluation function to rate the members of the population of solutions in comparison to one another; this evaluation, or fitness, determines the probability of a member of the population being chosen to reproduce, and thus pass along its characteristics to the next generation of possible solutions. It is often used to find an optimum (or near-optimum) solution to a function which is very difficult to optimize otherwise, and derives its effectiveness from an implicit parallelism in its search through the problem space. Holland (1975) demonstrated that a standard GA with a population size of N searches on the order of N^3 possible solutions with each new generation. Thus a GA is often able to search through a huge search space in a reasonable amount of time and come up with a good, if not optimal, solution.

The testbed was a data sample of 600 pixels, half road and half non-road, taken from a 400 x 400 pixel section (1200 meters x 1200 meters) of a LandSat Thematic Mapper satellite image of Vicksburg, Mississippi. The GA employed for this study used a population size of 100, initially consisting of 100 randomly-generated sets of rules; each set of rules was represented in the form of six pairs of numbers between 0 and 255, for the upper and lower bounds of the road pixels in the six channels. The only constraint imposed was that the lower bound must always be less than or equal to the upper bound.

The evaluation function embodied the knowledge that a human "expert" had about the roads in the image; the spectral values of road pixels were expected to fall between the upper and lower bounds for each channel, while the spectral values for non-road pixels were expected to fall outside of these bounds. One point was awarded for each channel that a set of rules got correct, for all of the pixels in the data sample. Thus, a set of rules that classified most of the road pixels correctly but misclassified most of the non-road pixels (or vice versa) would not score as well as a set of rules that performed reasonably well on both road and non-road pixels.

The GA used uniform crossover, steady state population change with no duplicates, and a linear ranking selection technique. In uniform crossover, each of the six rules from each parent has an equal chance to end up in a child, so different combinations of rules can be explored easily. Only one new child was produced during each generation, and this child was inserted into the population only if its fitness value exceeded that of at least one member of the current population; the least-fit member of the population was then deleted. This process places a lot of pressure on the population, forcing it to evolve rapidly, albeit at the expense of some loss of diversity. To insert some diversity back into the population, a relatively high mutation rate was adopted; crossover was performed during 70% of the generations, with mutation occurring during 30%. Selection for mating was done through a linear ranking process, insuring that those members of the population which had only a slightly better fitness values than their competitors nevertheless were selected considerably more often to participate in the crossover process.

The GA was run for 20,000 generations (a few minutes on a Sun 690 server), and the top-rated set of rules which had evolved was used to process the entire image. The results were compared to those obtained by a set of rules devised by the human "expert".

The GA evolved a set of rules which performed substantially better than the human, scoring about 15% higher than the human expert on the pixels in the data set; most of the gains came from not misclassifying non-road pixels. The human apparently had difficulty in devising rules which classified some roads incorrectly in order to perform better on non-road pixels. The GA was able to avoid this problem and come up with a balanced set of rules. An example of the output of the GA may be seen in Figure 4 below.

4 CONCLUSION

In comparing the results, the images produced by the sets of rules generated by the human and the GA seem very similar. Evidently, genetic algorithms show promise in their ability to evolve reasonably good sets of rules for identifying roads. At the very least, it is possible to use GAs to quickly evolve a set of rules that very closely emulates the set which a human expert can generate only after expenditure of considerable amount of time and effort. The results also suggest that humans use something equivalent to the GA's evaluation function -- perhaps their intuition about how "good" a processed image looks.

Further investigation revealed that humans performing road identification in satellite images also take context into consideration. Consequently, we might expect that techniques which use spatial as well as spectral information would stand a better chance of successfully separating roads from non-roads. In fact, additional research conducted by the author indicates this to be the case. Nevertheless, the above results remain interesting because they show that a GA can be used to produce very rapid preliminary classification of pixels into road and non-road categories, which can serve as a starting point for further image processing. Moreover, these results are interesting for what they suggest about how humans go about extracting roads from these images.

References

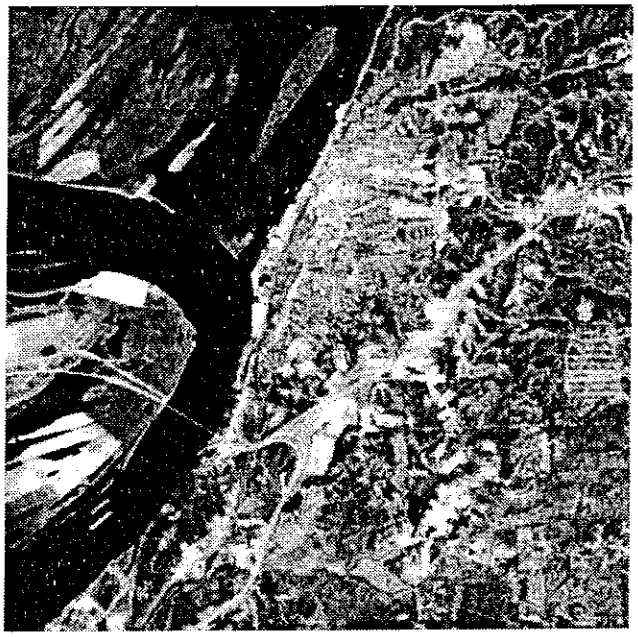
Boggess, J. E. 1994. *Using artificial neural networks to identify roads in satellite images*, **Proceedings of the World Congress on Neural Networks, San Diego, CA, 1994**, pp. I-410-I-415.

Holland, J. H. 1995. **Adaptation in natural and artificial systems**. Ann Arbor, MI: University of Michigan Press.

Ko, K. T. 1995. *A hybrid road identification system using image processing techniques and back-propagation neural networks*, Technical Report No. MSU-950607, Mississippi State University Department of Computer Science.

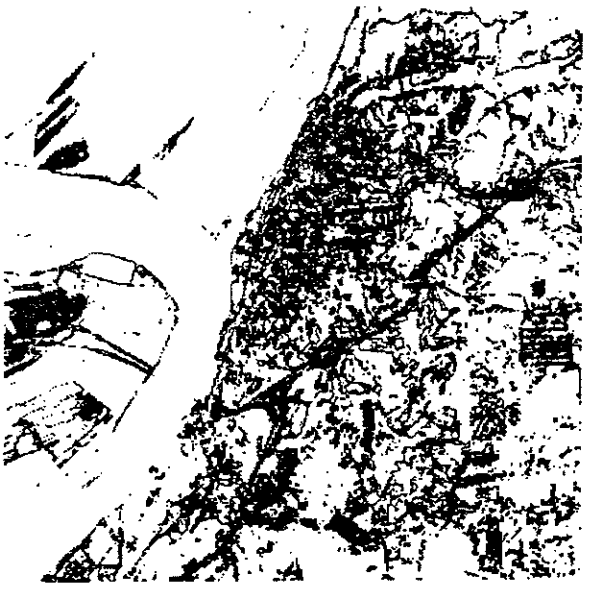
Wolfer, J, Roberge, J, and Grace, T. 1994. *Robust multispectral road classification in LandSat Thematic Mapper imagery*, **Proceedings of the World Congress on Neural Networks, San Diego, CA, 1994**, pp. I-260-I-268.

LandSat Thematic Mapper Satellite Image of Vicksburg, Mississippi: Channel 3, Histogram Equalized
Figure 1



LandSat Thematic Mapper Satellite Image of Vicksburg, Mississippi: Channel 7, Histogram Equalized
Figure 2

Output of the Human-generated Road Identification Expert System
Figure 3



Output of the Genetic Algorithm Road Identification Expert System
Figure 4

ADJUSTABLE GRAPHIC-BASED CLUSTERING METHOD

Liwu Chang
Naval Research Laboratory
Washington, DC USA
liwu@aic.nrl.navy.mil

Abstract

In this paper, we describe methods for a clustering system in the context of a graphic model from two aspects: The first aspect is about the properties of the proposed clustering method and the second aspect is on graph structure-based interpretation. We present methods for dynamically adjusting contents of clusters in the appearance of new samples. The result of clustering is regarded as a latent variable in the existing graph model. Clustering criteria are evaluated based on statistical properties of the graph model. To enhance the clarity in interpretation, we organize instances according to highly correlated features. We use an example to show the merits of using a graph model.

INTRODUCTION

Clustering is basically an inductive method that constructs a graph representation from a given data set with nodes standing for features, the unknown (referred to here as the hidden node) denoting the set of clusters and links indicating probabilistic relationships among features and the unknown [Pearl, 1988]. The hidden node is used to interpret unusual correlation embedded in a set of samples. In this paper, we describe the clustering system we have developed for experimenting with data organization from two aspects: The first aspect is about the properties of the proposed clustering method and the second aspect is on graphic-based interpretation. In particular, we consider the following challenging issues: One issue that plagues existing clustering methods is the inefficiency of handling large data sets. This problem exists in both top-down and bottom-up clustering search. The proposed method combinatively uses different modes of clustering. The second issue is the evaluation of clustering criteria for the generated cluster hierarchy. The evaluation is carried out on the basis of statistical properties of the graphic model. Another issue concerns inferencing multiple hierarchical clustering structures. Each hierarchical cluster structure is created according to a particular perspective. A subset of features is chosen to represent that perspective. In our experiments, we account for the perspective information as latent factors, each corresponding to a subset of highly correlated features. As a result, each sample can be organized from different perspectives.

In the next section, we describe the proposed clustering method. Its statistical properties are discussed in section 3. A simple example is shown in section 4. Clustering under different perspectives is discussed in section 5. A summary is given in the last section.

METHODS FOR HANDLING ITERATIVE ADJUSTMENT

Many statistical clustering algorithms use the agglomerative process that deals with all data simultaneously (i.e., the batch mode). Due to the need for exhaustive comparisons among entire clusters, the batch clustering approach is computationally inefficient at early stages of the clustering process. Moreover, it is difficult to update cluster distributions in the presence of new data. Alternatively, clustering can proceed in a sequential (or incremental) mode where one data instance is evaluated at a time. Because new data inevitably violate the global and local clustering constraints, a revision procedure is essential. However, revision may make the sequential mode of clustering costly as more data arrive. The proposed approach combines the use of both batch and sequential clustering modes. We take advantage of sequential clustering to quickly decide a cluster label for each new incoming sample. Then we adjust certain existing clusters. We impose a strict criterion for assigning new data in the sequential clustering phase and compensate for this by merging in the second phase.

The outcome from the proposed clustering method is a cluster hierarchy with each node representing a sub-class of samples. For a given initial hierarchy, the proposed method determines the location to which the new sample, s, should be placed based on a cluster criterion g. g is locally evaluated in each node. Clustering consists of four steps:

1. *Search:* This step examines the impact of the new sample s. Given hierarchy h, for a node n_i, if incor-

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/147 ©1996 FLAIRS

porating s increases g value, visit n_i's child nodes and prepare for updating g; otherwise stop and mark this node (called *residual*). There can be several residuals generated for a given s. Only the residual that contains the smallest sub-class in a branch is retained.

2. *Updating:* Update the g of node n_i if s is incorporated. Updating occurs on those nodes that are affected by the new arrival, s. The criterion g is defined as a function of similarity measures of samples in n_i. Currently, g is the average of the nearest neighbor distance (NNd). This metric yields more conservative (i.e., smaller) values compared with the result obtained from weighted minimum spanning tree algorithms [Chaudhuri, et al., 1994]. The conservative value helps to detect errors at early stages. This metric also provides necessary local information of a data point for re-grouping. Choices for NNd include difference between distributions of two classes, posterior probability of a class given s, etc. For each sample in n_i, we adjust g by replacing this sample's new NNd if it is affected by the addition of s.
3. *Revision:* This step adjusts node relationships in the hierarchy. The new arrival s is added to a residual, denoted as n_o, in which the NNd is minimum. In fact, NNd can be viewed as a global parameter, since it is compared across nodes.

   For any other residual, remove samples in its subclass from all its parent nodes (i.e., super-sets). Residuals and their child nodes may be conjoined with n_o. The condition of mergence may be judged by their g values. Alternatively, samples in those residuals are dismissed and re-labeled. This reassignment may spawn re-classification of other clusters.
4. *Selection:* This step selects and ranks different cluster partitions, where a partition corresponds to a set of exclusive nodes that jointly includes all samples collected so far. It's clear that this evaluation of clustering optimality is exponential in complexity. Constrained search (e.g., greedy) strategies are needed. Evaluation starts with the top node. In case of greedy search, for a certain partition p, the algorithm evaluates the clustering optimality for child nodes of each node in p and records the node, say n_p, with the highest score. The next partition to be visited is the one that contains the child nodes of n_p. Values of the hidden node will be described by the best partitioning.

Adaptive k-means revision has been discussed in several places (e.g., [Fukunaga, 1990]) for batch clustering. Incremental clustering algorithms in COBWEB [Fisher, 1987] and Anderson's categorizing approach [Anderson, 1990] does not provide revision. CLASSIT [Gennari et al., 1989] uses operators such as split and merging for revision. However, the revision is restricted to local comparison where no structural changes are involved and only metrics g are updated; this is inadequate to correct erroneous assignments made at earlier steps. The proposed adjustment method is similar to the incremental approach of [Nevins, 1995] in that both methods only re-classify certain portions of existing sub-clusters. The difference is that our method does not search the best match for those sub-clusters with respect to the entire hierarchy. We evaluate individual removed data points. This significantly reduces the traffic of transporting sub-clusters from one position to another. In our experiments with several data sets (e.g., UCI repository), the time required for adjustment is susceptible to the order of samples. The proposed method uses more time on computing NNd, while saving time on structural adjustment. The result is comparable with a batch mode.

STATISTICAL PROPERTIES OF GRAPHIC MODEL

Storing all hierarchies causes unnecessary complexity. Let the number of clusters be denoted by ω. Reduction in storage can be carried out by selecting a few prominent ω's. Suppose the criterion for assessing the number of clusters is determined by the posterior probability of the number of clusters, ω, given the set of samples, S, i.e., $Pr(\omega|S)$. In Bayesian formalism, the posterior distribution is the average of all models B_i,

$$Pr(\omega|S) = \sum_i Pr(\omega|B_i, S)Pr(B_i|S) \qquad (1)$$

As shown in (Figure 1), since the hidden node, c, is also an element of the graphic model, the feature set of each sample s should be augmented to include this variable. Hence, after rearrangement, $Pr(\omega \mid S)$ can be computed as

$$\alpha \int \sum_c \sum_i Pr(S, c|\theta, B_i, \omega)Pr(\theta|B_i, \omega)\, d\theta \qquad (2)$$

where θ stands for the parametric family. Factor α includes prior information about B_i and ω and the probability distribution of samples. The computation of (Equation 2) can be simplified under conditions where the structure of the graphic model, B_n, is known a priori and there is no preference on choosing the number of clusters.

For each ω, we compute $Pr(\omega \mid S)$ by instantiating the value of c (i.e., assigning a cluster label) for each sample s. Recall that the label of each sample in a partition is assigned at each level of evaluation of the Selection step. Hence, the summation in (Equation 2) will not be calculated. Note that c is considered a deterministic node. Suppose the likelihood of samples given θ obey a multinominal distribution. Under the assumption of a uniform distribution of θ [Berger, 1985], the score (i.e., the posterior probability) of a

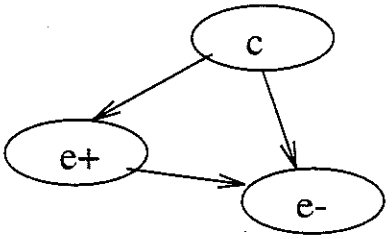


Figure 1: A Simple Graphic Model

particular value of ω is in proportion to

$$(\frac{(n_f-1)!}{(n_f+n_c-1)!}) \times \prod n_{fvc}! \qquad (3)$$

where n_{fvc} is the number of samples of cluster c at attribute f with value v, n_c stands for the number of data in the cluster c and n_f stands for the total possible values that attribute f can take (e.g., [Buntine, 1994][Cooper & Herskovitz, 1992]). This expression will be evaluated over all features. By ranking scores, several different ω's can then be selected for a given set of samples. Parameters associated with the graphic model need to be re-calculated whenever a new sample arrives. Note that Equation 3 can also be applied to the batch mode of clustering. Graphic model has recently been used in several research projects (e.g., AUTOCLASS [Cheeseman & Stutz, 1995], Ghahramani, (1994)).

EXAMPLE

Consider the data pattern given in table 1. It is assumed that symptoms developed by a patient are caused by exposure to some agents. Features of $x1$ to $x3$ $(e+)$ are the agents and $x4$ to $x10$ $(e-)$ are the manifestations observed. Hence, the two sets of features are causally related. Note features such as $x4$, 5, $x6$ and $x7$ appear frequently in the samples, across all agents. In the corresponding graphic model, since the goal of clustering is to find sub-classes within which samples possess strong similarity, the hidden node c is posited as the top node shown in (Figure 1).

In this example, samples are selected from three clusters. Because a large portion of features are shared by samples, failure to make distinctions between agent and manifestation features will result in samples grouped together regardless of their causes and hence, will yield unsatisfactory results. The proposed method, however predicts three clusters with correct labels.

Table 1: Samples

x1	x2	x3	x4	x5	x6	x7	x8	x9	x10
1	0	0	1	0	1	1	1	0	0
1	0	0	1	1	0	1	1	0	0
1	0	0	0	0	1	1	1	0	0
0	1	0	1	0	1	1	0	0	0
0	1	0	0	1	1	0	0	1	0
0	1	0	1	1	0	1	0	1	0
0	0	1	0	1	1	0	0	0	1
0	0	1	0	0	1	1	0	0	0
0	0	1	1	1	0	1	0	0	1

ORGANIZATION WITH DIFFERENT PERSPECTIVES

The need for perspectives by knowledge-based programs is prevalent. The goal is to gain clarity in representation and better interpretation of correlation among features.

The set of features may reflect various perspectives from which information was obtained. In clustering, using too many dependent features may cause interpretation problems. There are two reasons: First, the measure of similarity is perspective-sensitive. For instance, Guinea Pig is closer to Leopard than Macaw from the animal taxonomic perspective whereas it is closer to Macaw than Leopard if our perspective is pet. Clustering should be carried out with respect to a particular perspective. Second, if the number of features with one perspective significantly exceeds that of another one, the result of clustering will likely be dominated by the one with more features. Hence, we need to separate features and organize samples with respect to different perspectives.

In our approach, the property of perspective is regarded as the notion of a factor or a hidden structure - a subset of closely related features can be treated as statistically independent once the hidden structure is known. We apply clustering analysis to the derivation of properties of hidden structures and the feature selection method [Langley, 1994] to the evaluation of relevant features. The proposed approach derives the relevance relationship by means of a *sifting* process. For a cluster hierarchy h_i, we select salient features with respect to this hierarchy. Next, we store away those features and execute a clustering step to create the second hierarchy with the rest of features and the same set of samples. This sifting procedure is repeated until no significant outcome is recorded. Feature selection is carried out on the basis of improved predictive accuracy of test results. It uses the cross-validation procedure. The label of each sample is determined by its current cluster hierarchy. Start with a null set of fea-

tures. The improvement is measured before and after features are added to the original set. This approach iteratively examines different features and selects the combination with the highest predicted accuracy. The outcome may contain several subsets of features.

To show the generation of hidden structures, we borrow the example of animal categorization from [Martin & Billman, 1994]. In this data set, each animal is described by a list of features, i.e., "found", "food", "covering", "legs", "mobility", "reproduction" and "appearance". After the first clustering step, three clusters are generated from the animal data set and features "covering", "legs" and "mobility" are selected in the context of this set of clusters. On the next round, features "found"and "food" are selected with two generated clusters. With externally provided information, the first subset can be identified with the notion of "species" with three values $< bird, fish, mammal >$ and the second one corresponds to "affinity" with two values $< pet, wild >$. The notions of affinity and species correspond to two different perspectives. From the perspective of "species", features "covering", "legs" and "mobility" are thematic while other features are contextual. In a DAG model [Pearl, 1988], a perspective is represented as a top node. In this example, the top nodes include "affinity", "species", "reproduction" and "appearance". "Covering", "legs" and "mobility" are the child nodes of "species" and "found" and "food" are attached to "affinity".

The issue of extracting perspectives has been discussed in [Martin and Billman, 1995] on learning overlapped concepts. Their approach is expected to decide whether a sample should be assigned to a new perspective in the incremental mode of clustering. However, it's unclear how robust that approach is. Ballas (1993) has applied the concept of perspective to clustering environmental sounds. Perspective can be defined in various ways [e.g., Forbus, 1984] other than on correlated features.

SUMMARY

In this paper, we presented a clustering method for handling new samples through iterative adjustment, and evaluated clustering criteria in the context of a graphic model. With the graphic model, the result obtained from clustering can interpret samples more effectively. The proposed method allows one to handle elongated as well as compact data distributions.

Many issues, such as extracting multi-hidden causes and analyzing stability of sequential clustering methods, require further studies.

Acknowledgments

The author would like to thank Laura Davis, Behrooz Kamgar-Parsi, and Alan Meyrowitz for thier comments and discussion.

References

Anderson, J. (1990). *The adaptive character of thought.* Hillsdale, NJ: Erlbaum.

Ballas, J. (1993) Common factor in the identification of an assortment of brief everyday sounds. *Journal of Experimental Psychology: Human Perception and Performance, 2* pp 250-267.

Berger, J. (1985). *Statistical decision theory and Bayesian analysis.* New York: Springer-Verlag.

Buntine, J. (1994). Operations with learning with graphic models. *Journal of Artificial Intelligence Research, 2,* 159-225.

Chaudhuri, D., Murthy, C. & Chaudhuri, B. (1994). Finding a subset of representative points in a data set. *IEEE Transaction on SMC, 24,* pp. 1416-1424.

Cheeseman, P., Stutz, J. (1995). Bayesian classification (AUTOCLASS): theory and results. *Fayyad, U. et al (eds): Advances in Knowledge Discovery and Data Mining.* AAAI Press, Menlo Park.

Cooper, G. & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from Data. *Machine Learning, 9,* 309-347.

Fisher, D. (1987). Conceptual clustering, learning from examples, and inferences. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 38-49). Irvine, CA: Morgan Kaufmann.

Forbus, K (1984). Qualitative process theory. *Artificial Intelligence, 51,* pp. 95-143.

Fukunaga, K. (1990). *Statistical Pattern Recognition.* Academic Press, INC.

Gennari, J., Langley, P. & Fisher, D. (1989). Models of incremental concept formation, *Artificial Intelligence, 40*(1-3), (pp. 11-61).

Ghahramani, Z. (1994). Factorial learning and the EM algorithm. *Tesauro, G., et al. (Eds.), Advances in Neural Information Processing Systems 7.* Morgan Kaufmann.

Langley, P. (1994). Selection of relevant features in machine learning (Technical Report 94-3). Stanford, CA: Institute for the Study of Learning and Expertise.

Martin, J. & Billman, D. (1994). Acquiring and combining overlapping concepts. *Machine Learning, 16,* pp. 121-155.

Nevins, A. (1995). A branch and bound incremental conceptual clustering. *Machine Learning, 18,* pp. 5-22.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference.* San Mateo, CA: Morgan Kaufmann.

Scaling Learning by Meta-Learning over Disjoint and Partially Replicated Data

Philip K. Chan
Computer Science
Florida Institute of Technology
Melbourne, FL 32901
pkc@cs.fit.edu

Salvatore J. Stolfo
Department of Computer Science
Columbia University
New York, NY 10027
sal@cs.columbia.edu

Abstract

Many existing learning algorithms assume that the entire data set fits into main memory, which is not feasible for massive amounts of inherently distributed data. One approach we explore to handling a large data set is to partition the data into disjoint subsets, run the learning algorithm on each of these subsets of data, and combine the results by some means. The results achieved to date show promising results, but in nearly all cases accuracy of the final combined classifier is not as great as a single classifier computed from the entire data set. In this paper we evaluate our approach, called *meta-learning*, to learning from partitioned data where we relax the restriction that each subset of training data is disjoint, i.e. some amount of replication of training data is allowed. We anticipated data replication could improve overall accuracy, however, our findings suggest the contrary.

1 Introduction

With the coming age of very large network computing, it is likely that orders of magnitude more data in databases will be available for various learning problems of real world importance. Many existing learning algorithms require all the data to be resident in main memory, which is clearly untenable in many realistic databases. In certain cases, data is inherently distributed and cannot be localized on any one machine (for competitive business reasons, for instance). In such situations, it may not be possible, nor feasible, to inspect all of the data at one processing site to compute one primary "global" classifier.

Popular incremental learning algorithms such as [7, 10], aim to solve the scaling problem by piecemeal processing of a large data set that is consumed in a sequential fashion. Others have studied approaches based upon direct parallelization of a learning algorithm run on a centralized multiprocessor with "scalable" main-memory resources [13]. A review of such approaches has appeared elsewhere [5]. An alternative approach we study here is to apply *data reduction* techniques, where one may partition the data into a number of smaller *disjoint* training subsets, apply some learning algorithm on each subset (perhaps all in parallel), followed by a phase that combines the learned results in some principled fashion.

In such schemes one may logically presume that accuracy will suffer; i.e., combining results from a large number of separately learned classifiers trained on disjoint data may not be as accurate as learning one global classifier trained on the entire data set. Some amount of important information may be lost when reducing the data. Our proposed *meta-learning* techniques, which were first presented in [4], involves applying a learning algorithm to sets of predictions (treated as training data) made by a set of base classifiers in order to *learn how to combine* their collective predictions. Comparative results from applying meta-learning and other voting based strategies to learning from disjoint subsets were discussed in [5].

In this paper we provide an evaluation of our techniques and another published method on combining classifiers learned from partitioned subsets where some amount of *replication* is allowed. We believe this situation might be common in certain real world

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/151

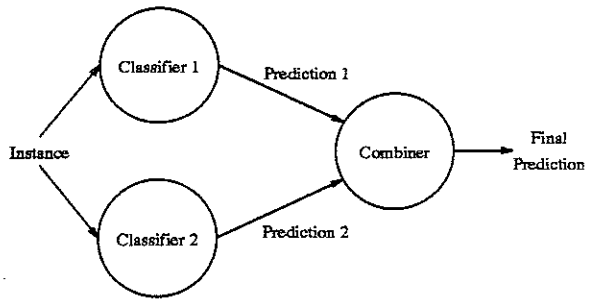


Figure 1: A combiner with two classifiers.

contexts. In situations where very large inherently distributed data exist, some amount of common information may be inevitable. Alternatively, in an attempt to boost predictive accuracy and to reduce inherent bias introduced by disjoint training data one may wish to purposefully replicate some amount of common training data. Howerer, this would incur higher computational cost. The question is, therefore, whether replication of training data buys anything?

2 Meta-learning Techniques

Our approach to combining multiple classifiers is to *meta-learn* a set of new classifiers, or *meta-classifiers*, whose training data are derived from a set of predictions generated by a set of base classifiers. Our techniques fall into two general categories: the *arbiter* and *combiner* strategies. Due to space limitations, only the combiner strategies are discussed here. Detailed descriptions of the *arbiter* strategies can be found in [3].

Combiner In the *combiner* [3] strategy, the predictions of the learned base classifiers on the training set form the basis of the meta-learner's training set. A *composition rule*, which varies in different schemes, determines the content of training examples for the meta-learner. From these examples, the meta-learner generates a meta-classifier, that we call a *combiner*. In classifying an instance, the base classifiers first generate their predictions, and then a combiner produces a final prediction (see Figure 1). The aim of this strategy is to combine the predictions from the base classifiers by learning a relationship or function between base predictions and the correct classification.

We experimented with two schemes for the composition rule. First, the predictions, $C_1(x)$, $C_2(x)$, ... $C_k(x)$, for each example x in the validation set of examples, E, are generated by the k base classifiers. These predicted classifications are used to form a new set of "meta-level training instances," T, which is used as input to a learning algorithm that computes a combiner. The manner in which T is computed varies as defined below. In the following definitions, $class(x)$ and $attribute\_vector(x)$ denote respectively the correct classification and the entire set of attribute values of tuple x, where x is a member of the validation set, E.

1. Return meta-level training instances with the correct classification and the predictions; i.e., $T = \{(class(x), C_1(x), C_2(x), ...C_k(x)) \mid x \in E\}$. This scheme was also used by Wolpert [11]. (For further reference, this scheme is denoted as *class-combiner*.)

2. Return meta-level training instances as in *class-combiner* with the addition of the attribute vectors; i.e., $T = \{(class(x), C_1(x), C_2(x), ...C_k(x), attribute\_vector(x)) \mid x \in E\}$. (This scheme is denoted as *class-attribute-combiner*.)

The next section discusses our findings from experiments where a controlled amount of "arbitrary replication" is permitted in each subset of data used to train the base classifiers.

3 Learning Tasks and Experimental Setup

Two decision-tree inductive learning algorithms were used in our experiments: ID3 [8] and CART [1], both obtained from NASA Ames Research Center in the IND package [2]. Two data sets were used in our studies. The DNA splice junction (SJ) data set[1] [9] contains 3,190 training instances of nucleotides and the type of splice junction, if any, at the center of each sequence. There are three possible junctions, and hence 3 categorical classes in this task. The protein coding region (PCR) data set[2] [6] contains DNA nucleotide sequences and their binary classifications (coding or non-coding). The PCR data set has 20,000 sequences. These two data sets represent two different kinds of learning tasks: one is difficult to learn (PCR at 70+% accuracy for over 20,000 training examples) and the other is easy to learn (SJ at 90+% accuracy for 3190 examples).

We varied the number of equi-sized subsets of training data from 2 to 64 ensuring each was disjoint but

[1] Courtesy of Towell, Shavlik and Noordewier.
[2] Courtesy of Craven and Shavlik.

with a distribution of examples of each class proportional to that of the entire database. We measure the overall predictive accuracy as the ratio of correct classifications over the total number of samples in a *test set* which is disjoint from the training data. Results presented in the following section are averages over 10-fold cross validation experiments.

Since the data we are dealing with here is relatively small, we have the opportunity to evaluate the various proposed schemes by comparing their accuracy against a "global classifier," i.e. a classifier that is learned from the entire data set. We call the accuracy of this global classifier produced by ID3 and CART the "baseline case," which is plotted in our graphs as the "one subset" case on the X-axis. In addition to our meta-learning strategies, a Bayesian statistical approach (*bayesian-belief*) as presented in [12] was also used in our experiments to combine classifiers.

4 Experimental Results on Replicated Data

In our experiments each partition of data allowed some amount of replication. We prepare each learning task by generating subsets of training data for the base classifiers according to the following generative scheme.

1. Starting with N disjoint subsets, randomly choose from any of these sets one example X, distinct from any other previously chosen in a prior iteration.

2. Randomly choose a number r from $1...(N-1)$, i.e. the number of times this example will be replicated.

3. Randomly choose r subsets (not including the subset from which X was drawn) and assign X to those r subsets.

4. Repeat this process until the size of the largest (replicated) subset is reached to some maximum (as a percentage, Δ, of the original training subset size).

In the experiments reported here, Δ ranged from 0% to 30%. Each set of incremental experimental runs, however, chooses an entirely new distribution of replicated values. No attempt was made to maintain a prior distribution of training data when incrementing the amount of replication. This "shot gun" approach provides us with some sense of a "random learning problem" that we may be faced with in real world scenarios where replication of information is likely inevitable or purposefully orchestrated.

The graphs in Figure 2 plot the results for only the *class-combiner* and *bayesian-belief* strategies. Results from other strategies are not shown. The results in all cases are conclusive: *replication essentially buys nothing!* In each case no measurable improvement in predictive accuracy is seen no matter which learning algorithm or combining scheme is used.

These negative results for replication are in fact positive from the perspective of computational performance! One may presume that applying a number of instances of a learning algorithm to disjoint training data results in a set of base classifiers each biased towards its own partition of data. Combining two or more such biased base classifiers by meta-learning attempts to share knowledge among the base classifiers and to reduce each individual's bias. Replication of training data is an alternative attempt to reduce this bias. Common or shared information replicated across subsets of training data at the onset of learning attempts to provide each learned base classifier with a "common view" of the learning task. The results here show that meta-learning from disjoint training data does an effective job of sharing knowledge among separate classifiers anyway. In fact, the overhead that may be attributed to replicated data (since the same data is being treated multiple times by separate learning processes) may be comfortably avoided, i.e. meta-learning on purely disjoint data seems to achieve good performance, at perhaps optimal speeds due to optimal data reduction.

These rather surprising results are of course limited to the learning algorithms and data sets used in this study. Additional experiments on other learning tasks are required to confirm this behavior. Even so, these two "arbitrarily chosen" tasks have provided impressive evidence of this interesting behavior.

5 Concluding Remarks

We have demonstrated that meta-learning over base classifiers trained on disjoint data is not measurably improved by schemes that attempt to replicate common information for initial training of the base classifiers. From a practical perspective this implies we can comfortably apply meta-learning techniques to disjoint partitions of data to maximize computational performance without a substantive negative impact on predictive accuracy.

We are exploring the use of meta-learning in improving the accuracy performance of local learned models by merging them with ones imported from

remote sites. That is, at each site, learned models from other sites are also available; however, raw data are not shared among sites. This happens when companies are willing to share "black-box" models, but not the proprietary raw data from which those models are derived for competitive reasons. Futhermore, we investigate the effects on local accuracy when the local underlying training data overlap with those at remote sites. Studies in measuring the independence among base classifiers and simplfying the final meta-learned structure by pruning related base classifiers are also underway.

Acknowledgment

This work was performed at Columbia University and has been partially supported by grants from New York State Science and Technology Foundation, Citicorp, and NSF grant IRI-94-13847. The first author would also like to thank Florida Tech for the usage of its resources.

References

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, Belmont, CA, 1984.

[2] W. Buntine and R. Caruana. *Introduction to IND and Recursive Partitioning.* NASA Ames Research Center, 1991.

[3] P. Chan and S. Stolfo. Experiments on multistrategy learning by meta-learning. In *Proc. Second Intl. Conf. Info. Know. Manag.*, pages 314–323, 1993.

[4] P. Chan and S. Stolfo. Meta-learning for multistrategy and parallel learning. In *Proc. Second Intl. Work. on Multistrategy Learning*, pages 150–165, 1993.

[5] P. Chan and S. Stolfo. A comparative evaluation of voting and meta-learning on partitioned data. In *Proc. Twelfth Intl. Conf. Machine Learning*, pages 90–98, 1995.

[6] M. Craven and J. Shavlik. Learning to represent codons: A challenge problem for constructive induction. In *Proc. IJCAI-93*, pages 1319–1324, 1993.

[7] J. R. Quinlan. Induction over large data bases. Technical Report STAN-CS-79-739, Comp. Sci. Dept., Stanford Univ., 1979.

[8] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[9] G. Towell, J. Shavlik, and M. Noordewier. Refinement of approximate domain theories by knowledge-based neural networks. In *Proc. AAAI-90*, pages 861–866, 1990.

[10] J. Wirth and J. Catlett. Experiments on the costs and benefits of windowing in ID3. In *Proc. Fifth Intl. Conf. Machine Learning*, pages 87–99, 1988.

[11] D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[12] L. Xu, A. Krzyzak, and C. Suen. Methods of combining multiple classifires and their applications to handwriting recognition. *IEEE Trans. Sys. Man. Cyb.*, 22:418–435, 1992.

[13] X. Zhang, M. Mckenna, J. Mesirov, and D. Waltz. An efficient implementation of the backpropagation algorithm on the connection machine CM-2. Technical Report RL89-1, Thinking Machines Corp., 1989.

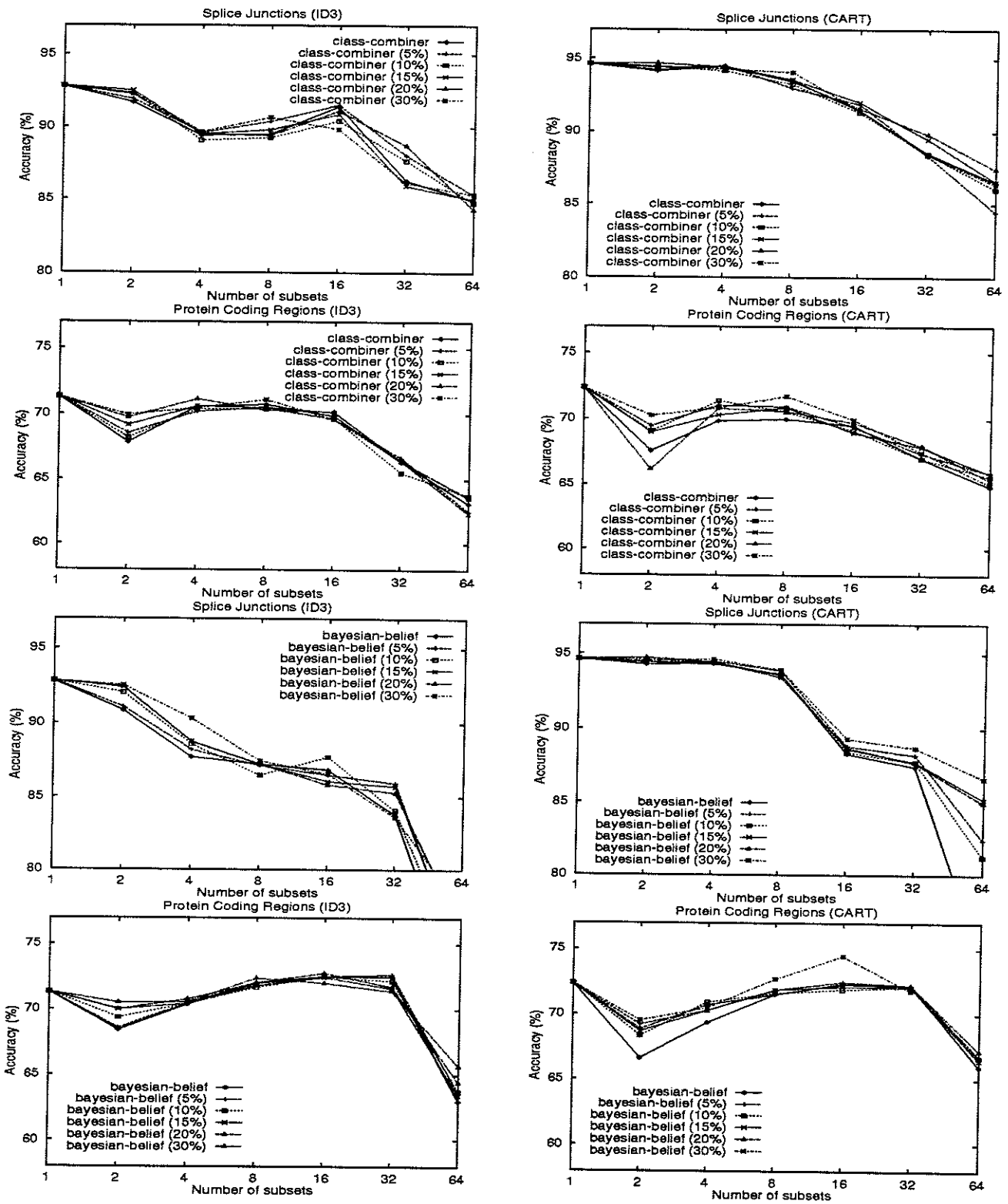


Figure 2: Accuracy for combining techniques trained over varying amounts of replicated data. Δ ranges from 0% to 30%.

Learning Decision Rules in Parallel

Ning Shan, Howard J. Hamilton and Nick Cercone
Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada S4S 0A2
e-mail: {ning, hamilton, nick}@cs.uregina.ca

Abstract

A parallel design for learning decision rules using a rough sets technique is presented. In our approach, knowledge is represented by a set of decision rules with a minimum number of condition attributes. To reduce the time complexity of algorithms for generating the set of decision rules, we propose new parallel algorithms based on associative memory (AM) processing.

1 Introduction

Acquiring decision rules (or production rules) is an important phase in developing a rule-based expert system. The inductive learning approach for inducing rules from data, in which instances are represented in terms of attributes and their values, is probably the oldest and most successful approach [3,4]. In general, given positive and negative instances of some concepts, the problem is, for each concept, to find the generalization from the data that best describes the concept. A *generalization* consists of a discriminating logical description that deterministically separates instances of one concept from those of other concepts. Hence, the problem is reduced formulating the simplest logical description of the target concept in terms of the specified attributes.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/156 ©1996 FLAIRS

The methodology of rough sets [1,2] has been applied to inducing decision rules [1,7]. Systems based on rough sets require a module for generating the best set of maximally general decision rules characterizing the dependency between condition and decision attributes. *Maximally general* decision rules are those with a minimum number of condition attributes. Two steps to generate a set of such rules are [1]: (1) eliminate redundant attributes by performing dependency analysis (*attribute reduction*) and (2) eliminate redundant values for the nonredundant attributes to generate decision rules with a minimum number of terms (*value reduction*).

Both procedures are time consuming. On a conventional serial computer, the time complexity of attribute reduction is $O(n^2 \times m)$, where n is the number of distinct instances, and m is the number of condition attributes, and the time complexity of value reduction is $O((n')^2 \times m')$, where n' is the number of distinct instances in the reduced decision table, and m' is the number of condition attributes in the reduced decision table. To reduce this time complexity, we propose a parallel architecture based on associative memory that reduces complexity for attribute and value reduction to $O(n \times m)$ and $O(n' \times m')$, respectively.

The remainder of the paper is organized as follows. In Section 2, we describe the input instances and preparation required to create an input decision table. In Sections 3 and 4, we present algorithms for attribute reduction and value reduction. In Section 5, we describe our associative memory (AM) architecture and show how it can be applied to the two algorithms. In Section 6, we draw conclusions.

U	C				D
	C_1	C_2	C_3	C_4	$Class$
x_1	0	0	0	0	1
x_2	1	1	1	1	0
x_3	0	1	0	0	1
x_4	1	1	1	0	1
x_5	1	0	1	1	1
x_6	0	1	1	1	1

Table 1: Example Decision Table

U	C			D
	C_1	C_2	C_4	$Class$
x_1	0	0	0	1
x_2	1	1	1	0
x_3	0	1	0	1
x_4	1	1	0	1
x_5	1	0	1	1
x_6	0	1	1	1

Table 2: After Attribute Reduction

```
Input: I, the set of input instances;
       C, the set of condition attributes
Output: C', the reduced set of attributes
C' := C
for a := 1 .. |C|
   C' := C' − {C_a}
   needed := false
   for i = 1 .. |I|
      for j = i + 1 .. |I|
         if Cond(C', x_i) = Cond(C', x_j)
           and Deci(i) ≠ Deci(j) then
             needed := true
             break from loop
         end if
      end for
      if needed then
         C' := C' ∪ {C_a}
         break from loop
      end if
   end for
end for
```

Figure 1: The MINATTR Algorithm

2 The Input Decision Table

Given a set I of input instances, the following preparation steps are required: (1) combine all instances with the same condition values but different decision values (the boundary approximation space in rough sets terminology [1]) by creating a new decision value for each unique disjunctions of decision values that appears; (2) convert all multivalued attributes in I into binary attributes; (3) create a separate decision table DT_d for each decision attribute d; (4) delete all duplicate instances (rows) from $DT_1, \ldots, DT_d$. After deletion, each remaining instance represents an equivalence class. Henceforth, we assume that the input is a decision table DT containing m binary condition attributes $C_1, \ldots, C_m$ and one decision attribute $Class$.

In the decision table shown in Table 1, the condition attributes are $C = \{C_1, C_2, C_3, C_4\}$ and the decision attributes are $D = \{Class\}$. The partition produced by classifying according to class is $\{Y_1, Y_2\}$, with $Y_1 = \{x_2\}$ and $Y_2 = \{x_1, x_3, x_4, x_5, x_6\}$.

3 Attribute Reduction

Attribute Reduction [1] is the process of obtaining a *reduct* (a minimal set of attributes). If an attribute in a decision table is not needed to classify the instances according to the decision attribute, it can be eliminated. Redundant attributes are likely to exist because of the preparations performed on the input. One possible reduct of C with respect to D is $C' = \{C_1, C_2, C_4\}$, as shown in Table 2.

To perform attribute reduction, we use the MINATTR algorithm (Figure 1). We consider each attribute $C_a \in C$. If any two instances x_i and x_j have identical values for the subset of attributes $C - \{C_a\}$, but have different decision values, then attribute C_a is needed; otherwise it is removed. $\text{Cond}(C', x_i)$ represents the values in instance x_i for the condition attributes in C', and $\text{Deci}(x_i)$ is the value of the decision attribute in instance x_i. The explicit breaks are shown to facilitate conversion into AM-based parallel procedures. Given n input instances and m attributes for each instance, this algorithm is $O(n^2 \times m)$.

No attribute in the reduct C' can be eliminated without affecting its classification decisions. By choosing the attributes from C in different orders, other reducts could be generated. Any such reducts can distinguish deci-

sion classes as well as C. Computing all reducts is NP-hard [8] and unnecessary here.

After the MINATTR algorithm has been applied, duplicates are removed from the instance set I, yielding I', because when an attribute is removed, instances that differ only in that attribute become identical. Once again, each remaining instance thus represents an equivalence class of instances. The result of this process is a reduced decision table DT'.

4 Simplification of Decision Rules

Decision rules could be obtained directly from the reduced decision table DT'. However, simpler rules can be created by dropping some of the terms specifying conditions between attributes and values. The process by which the maximum number of conditions (each specifying one value for an attribute) are removed without losing any essential information is called *Value Reduction* [1], and the resulting rule is called a *maximally general rule.* As with other machine learning algorithms, we attempt to ensure that (1) each rule is of minimum length and (2) the number of rules is minimum. Our algorithm is more successful at the former than the latter.

We attempt to identify unneeded conditions by examining the reduct, with reference to the attributes in C', for each instance. We create one reduct for each instance and call it a rule; the union of these reducts is the set of maximally general rules.

To perform value reduction, we use the MINRULE algorithm (Figure 2). The general idea is that for each instance, we try dropping each attribute value. If no inconsistency results, the value can be removed. The details are similar to those in the MINATTR algorithm, except that a set of rules is accumulated in $MinRuleSet$. Given n' instances and m' attributes for each instance, the algorithm is $O((n')^2 \times m)$.

From Table 2, we compute the reduct for each instance x_i and eliminate values for attributes not present in that reduct. The results are shown in Table 3, where '-' indicates *don't care.* The maximally general decision rules are:
$(C_4 = 0) \rightarrow (Class = 1)$
$(C_1 = 1) \wedge (C_2 = 1) \wedge (C_4 = 1) \rightarrow (Class = 0)$
$(C_2 = 0) \rightarrow (Class = 1)$
$(C_1 = 0) \rightarrow (Class = 1)$

```
Input: I', a set of input instances;
       C', the set of condition attributes
Output: MinRuleSet, the set of decision rules
MinRuleSet := φ
for i = 1 .. |I'|
   Rule := x_i
   for a := 1 .. |C'|
      Rule := Rule − {C'_a}
      needed := false
      for j = 1 .. |I'|1
         if Cond(Rule, x_i) = Cond(Rule, x_j)
           and Deci(i) ≠ Deci(j) then
            needed := true
            break from loop
         end if
      end for
      if needed then
         Rule := Rule ∪ {C'_a}
      end if
   end for
   MinRuleSet := MinRuleSet ∪ Rule
end for
```

Figure 2: The MINRULE Algorithm

U	C			D
	C_1	C_2	C_4	$Class$
x_1	-	-	0	1
x_2	1	1	1	0
x_3	-	-	0	1
x_4	-	-	0	1
x_5	-	0	-	1
x_6	0	-	-	1

Table 3: Simplified decision table

5 Parallel Approach

We apply parallelism to the innermost loop of MINATTR (or MINRULE), which performs *exact matching.* We can replace each of these exact matching loops by a constant-time operation on an associative memory (AM) [5] with n words (one for each instance) and $m+1$ bits per word (one for each attribute).

An AM consists of a memory, a control unit, and additional logic (such as tags and data-

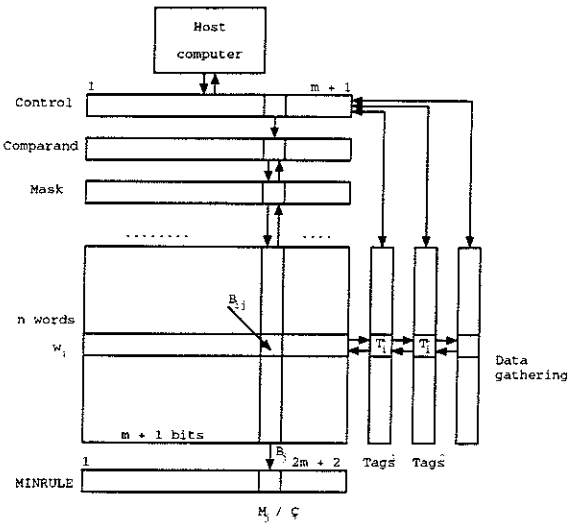


Figure 3: Structure of Associative Memory

gathering logic) and is controlled by a host computer. Each memory word consists of a flip-flop, comparison logic, read-write logic and additional logic. The ith word is denoted as $w_i = (B_{i,1}, B_{i,2}, ..., B_{i,m+1})$. Each *bit cell* $B_{i,j}$ can be written, read, or compared with external signals. In a time unit, all bit cells are supplied with the same external signals and execute the same operations.

The *control unit* consists of the actual control logic, the comparand, and the masking registers. The *comparand register* $C = (C_1, C_2, ..., C_{m+1})$ is used to store key operands being searched for or being compared with. The *masking register* $M = (M_1, M_2, ..., M_{m+1})$ is used to enable or disable the bit slices (vertical columns) involved in the operations. Each memory word has at least one tag bit T_i associated with it. The collection of tag bits is referred to as the *response* to a search. Using data-gathering logic, the control unit can read the status of the tag bits, which contain the results of the associative operations.

The structure of an AM for our task is shown in Figure 3. We associate two tag bits with each word in the memory: T_i^1 holds the response to a search and T_i^2 enables/disables word w_i for the current operation.

The instructions provided by our AM are described in [9]. Two features are (1) the **Some/None** signal, which reports to the control unit that some, or none, of the words have their tag bits set ($T_i^1 = 1$) in response to the previous search, and (2) the **Select First Responder** signal, which sets the tag bits T_j^1 to 0 for all cells j following the first cell i with its tag bit $T_i^1 = 1$.

Our implementation of the exact match algorithm (used in both MINATTR and MINRULE) simultaneously compares the contents of comparand register C activated by masking register M with all memory words. First, all tag bits T_i^2 are set to '1' ("enabled") and all tag bits T_i^1 are set to the default setting '1' ("match"). Masking bit M_j is activated only for the bits corresponding to the attributes of interest. The control unit issues a *search* instruction. Each bit $B_{i,j}$ of word w_i is compared to the corresponding bit C_j of the comparand. If any inequality is detected, a mismatch/disable signal is generated for word w_i. The tag bit T_i^1 is reset to '0' by this signal. The *some/none* output reports to the control unit that some, or none, of the words have their tag bits set. Thus, it tells whether any word responded to the previous search. Those words left in the "match" state at the end of processing exactly match the comparand register associated with the masking register. Also, if the T_i^2 bit of word w_i is set to '0', then the tag bit T_i^1 is also set to '0'. Detailed implementations are presented in Figures 4 and 5.

6 Conclusion

Significant opportunities for parallel processing exist in algorithms based on rough sets [6]. As a first step, a parallel algorithm to generate the set of maximally general decision rules was presented. A special purpose AM architecture, which includes a $(m+1) \times n$ bit memory, was presented. For induction problems with at most n input instances, m binary condition attributes, and one binary decision attribute, our approach decreases the time complexity from $O(n^2 \times m)$ for serial computation to $O(n \times m)$.

```
Write all instances into the AM
Set MM = 1...10
FOR a = 1 .. m
   Set MM_a = 0
   Set D    /* All instances */
   FOR i' = 1 .. n
      Set T^1, T^2 = D
      Select First Responder, call it i
      Set D_i = 0      /* Mark instance */
      Read w_i into variable W
      Load C = W
      Load M = 0...00      /* Ignore cond attrs */
      Set M_{m+1} = 1      /* Look at decision attr */
      Search      /* Search for Deci = Deci
      Set T^2 = T̄^1      /* Deci ≠ Deci*/
      Load M = MM
      Set T^1
      Search      /* Search for Cond = Cond */
      IF Some/None = 1 THEN    /* If needed */
         MM_a = 1      /* Restore attr */
         break
      ENDIF
   END FOR
END FOR
MM identifies attributes in reduct
```

Figure 4: MINATTR Procedure for AM

```
Write all instances into the AM
Set D      /* All instances */
FOR i' = 0 .. n
   Set T^1, T^2 = D      /* Unexamined instances */
   Select First Responder, call it i
   Set D_i = 0      /* Mark this instance */
   Read w_i into variable W
   Load C = W
   Load M = 0...00      /* Ignore cond attrs */
   Set M_{m+1} = 1      /* Look at decision attr */
   Search      /* Search for Deci = Deci */
   Set T^2 = T̄^1      /* Deci ≠ Deci */
   Load M = 1...11      /* All attributes */
   Set M_{m+1} = 0      /* Except decision attr */
   FOR a = 1 .. m      /* All cond attrs */
      Set T^1
      Set M_a = 0      /* Remove an attribute */
      Search      /* Search for cond = cond */
      IF Some/None = 1 THEN    /* If needed */
         MM_a = 1      /* Restore attribute */
      ENDIF
   END FOR
   Set M_{m+1} = 1
   Minrule      /* Collect results */
   Read MINRULE into variable Rule
   MINRULES := MINRULES ∪ {Rule}
END FOR
```

Figure 5: MINRULE Procedure for AM

References

[1] Pawlak, Z. *Rough Sets: Theoretical Aspects of Reasoning About Data,* Kluwer, 1991.

[2] Pawlak, Z. "Rough Sets," *International Journal of Information and Computer Science,* **11**(5):341-356, 1982.

[3] Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. (eds.) *Machine Learning: An Artificial Intelligence Approach,* Tioga, Palo Alto, CA, 1983.

[4] Shi, Z. *Principles of Machine Learning.* International Academic Publishers, 1992.

[5] Moldovan, D.I. *Parallel Processing: From Applications to Systems,* Morgan Kaufmann, San Mateo, CA, 1993.

[6] Muraszkiewicz, M. and Rybinski, H. "Towards a Parallel Rough Sets Computer," *Proc. International Workshop on Rough Sets and Knowledge Discovery,* Banff, Canada, 1993.

[7] Slowinski, R. (ed.) *Intelligent Decision Support: Handbook of Applications and Advances of Rough Sets Theory,* Kluwer, 1992.

[8] Wong, S.K.M. and Ziarko, W. "On Optimal Decision Rules in Decision Tables," *Bulletin of Polish Academy of Sciences,* **33**(11-12):693-191, 1985.

[9] Shan, N. and Zhang, C.N. "A Parallel Processing for Generating Maximally General Decision Rules Based on Rough Set Theory," *EXPERSYS-94 (Expert System Applications & Artificial Intelligence),* i.i.t.t. international, Gournat sur Marne, France. pp. 619-624.

An expert system for planning real-time distributed task allocation.

M. Alfano, A. Di Stefano, L. Lo Bello, O. Mirabella

Universita' di Catania, Facolta' di Ingegneria
Istituto di Informatica e Telecomunicazioni
Italy
email:ad@iit.unict.it, alfano@icsi.berkeley.edu,
llobello@iit.unict.it, omirabel@iit.unict

Abstract

In the current scenario, industrial process control is characterized by control, monitoring, and measurement tasks with a known dynamic which can, and often have to, be distributed over several nodes of a computer network. Response time and reliability are the main parameters a task allocator has to take into account in a distributed process control environment. Cost is a further parameter that can be computed to evaluate the choices made.
In this paper we propose an expert system for user-driven off-line process allocation which plans the distribution of the processes of a control task with a known dynamic over a network of nodes with heterogeneous hardware features. The aim of the allocator is to minimize the response time, or at least keep it within the time constraints indicated by the designer, at the same time guaranteeing a suitable degree of reliability. Finally, an example will be discussed and evaluated regarding the assignment of the modules of the known application to the nodes of a heterogeneous system.

1 INTRODUCTION

In the current scenario, industrial process control is characterized by control, monitoring, and measurement tasks with a known dynamic which can, and often have to, be distributed over several nodes of a computer network [1]. In the area of distributed process control the problem of task allocation is therefore widely felt.
Distributing the activities of a task over several processing nodes in a parallel system, whether it is closely or loosely connected, is a widely investigated approach nowadays in the attempt to balance the workload fairly between the computation resources and increase the average task execution speed. [2-7].
The scenario of distributed process control features tasks with a dynamic that is almost always totally known and strict time constraints. This means that providing adequate response time is not only useful to improve the efficiency of the system, but often necessary in order to meet the time constraints of the process being controlled.
On the other hand, the software of a process control system features not only predictable behaviour but also a certain stability: the task scenario varies slowly and a new control or monitoring task can be introduced after off-line assessment of its impact on the existing system.
No less important than task response times are problems of reliability [5]. Reliability is defined here as the probability that execution of a task will be correctly achieved. Some control tasks are so critical as to require replication on several nodes in order to ensure greater fault tolerance. Distributing the execution of a process over several machines ensures greater certainty of completion of the task, even on the occurrence of a partial or total fault in a machine executing one or more of the processes involved. This choice may affect the performance of the system as far as response time is concerned. Executing replicas of a process on different processing nodes increases the average workload on each of them and therefore leads to the need to schedule longer running queues locally, thus slowing down the processing of the single process. It is therefore necessary to reach a good trade-off between reliability and response time.
Response time and reliability are the main parameters a task allocator has to take into account in a distributed process control environment, and the choice of a strategy which can find the best trade-off, meeting the constraints imposed by the control scenario, is of fundamental importance.
To sum up, the features of a task allocator in a process control environment are:

- the dynamic of the task is known a priori;
- the scenario of the control software varies slowly;
- since allocation has to meet often urgent reliability and response time constraints, it has to be accurately evaluated and can be determined off-line.

In this paper we propose an Expert System (E.S.) for user-driven off-line process allocation which plans the

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/161

distribution of the processes of a control task with a known dynamic over a network of nodes with heterogeneous hardware features. The aim of the allocator is to minimize the response time, or at least keep it within the time constraints indicated by the designer, at the same time guaranteeing a suitable degree of reliability. The E.S. has to aid the designer in choosing the number and type of network nodes to be used and assessing the cost of the computing resources needed. The allocator is therefore guided by rules which establish the time constraints as well as the topological and hardware ones (a process cannot always be executed by any node whatsoever, but may require a specific machine), and suitable reliability and cost parameters.
The allocation algorithm proposed follows a backward-forward approach which, on the basis of a model of the task, tends to minimize the response time for an approximate configuration of the modules and their replicas. Evaluation of the response time and reliability by means of heuristic techniques we propose will allow us to check that the allocation meets the constraints imposed by the designer, or whether it is necessary to go back and find a suitable trade-off between efficiency and reliability.
In Sect. 2 below we deal with the model of the distributed application, in Sect. 3 we outline the structure and functioning of the Expert System, and finally in Sect. 4 we discuss, as an example, the results of the allocation of a known application on a heterogeneous system.

2 SOME REMARKS ON THE APPLICATION GRAPH MODEL

Here we consider an application task T comprising m processes or modules M_m and a distributed system made up of a set of n processing nodes $\Pi_n=\{P_1,...,P_n\}$ totally connected by a communication network. The problem of task allocation can be formulated as assigning the modules to the nodes in such a way as to minimize the overall response time and to satisfy the constraints imposed on the single module and all the tasks of the system in terms of time, reliability and hardware and software compatibility.
The task model is specified through its *application graph* $<M_m, R>$, the nodes of which represent the modules M_m making up the application and the arcs the relationships $r_{ij} \in R$ between them.
The nodes of the graph contain information on certain hardware and software parameters typical of the modules they represent, such as the execution time on a reference processor (*comp_t()*), a list of the nodes on which the module can be executed, etc.
The model proposed takes three types of relationship into consideration: *precedence*, *client/server* and *synchronization*. Figure 1 shows an example of an application graph which summarizes the relations described below.

1. *Precedence relationship* M_i -> M_j : M_i has to terminate its execution before M_j can start. It is represented by an arc from node M_i to node M_j.
2. *Client/server relationship* M_i - S_k: system servers are particular processes activated in the processing nodes according to specific strategies which are absolutely independent of process allocation. The servers in our model are therefore special modules which do not belong to any task, the allocation of which is assumed to be given when the application is scheduled.
   It is assumed that for each client-server relationship, the allocation algorithm possesses information about the server process, such as the name of the node it resides in and its service execution time, given by the sum of the server's computing time and its communication time.
3. *Synchronization relationship* M_i<->M_j: here we represented a single synchronization relationship which may indicate an activity made up of several communications so closely correlated with each other as to be seen as a single communication phase.

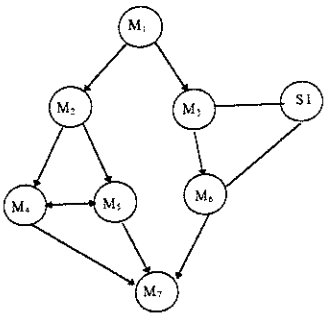


Figure 1: Application graph

3 STRUCTURE AND FUNCTIONING OF THE EXPERT SYSTEM

The Expert System presented is composed of blocks of rules, each of which solves a particular subproblem, which is allocating the modules of the application task, T, in such a way as to meet user requirements. One block chooses an allocation which meets the constraints the user requires for the response time of T, another focuses on the reliability of the application task, and another again calculates the global cost of execution, etc. We call these blocks Rule Blocks.
Since the three parameters - response time, reliability and cost of the application task T - are correlated, none

of these rule blocks is a separate entity, but contains links to one or more of the other blocks. These links represent the fact that choice of an allocation which meets user requirements for only one of these parameters may affect the performance of the system as far as the other ones are concerned, and that determination of the allocation which achieves the best trade-off between them will require backtracking and reconsideration of some of the choices already made.
Each block is made of a set of rules which are activated by the reasoning itself; some of them represent alternative solutions, whereas others cooperate to solve the specific problem.
The rules are structured as conditional phrases of the following form:
if<conditions>**then**<hypothesis>**do**<action>
where a condition can be found to be *true*, *false*, *not known* or *unknown*.
The reasoning session is started and supported by a mechanism according to which all the consequences of some change in the value of the conditions of the rules making up the knowledge base are examined. A condition can change its value according to the action specified in another inferred rule or following intervention on the part of the user.
The E. S. presented is, in fact, interactive, as the user does not only intervene in the system configuration phase, which is off-line, but takes an active part in the allocation process thanks to dialogue windows through which he supplies evaluation elements and answers which guide the inference engine along the various possible paths it could take.
All the rules in which changes occur in the value of the conditions are examined, to see whether all the conditions are true and therefore to conclude that the hypothesis of the rule is also true. The inference mechanism described above can be both a backward and a forward one. Reasoning stops when all the consequences of the change in value of the conditions have been examined.

3.1 Configuration

Configuration of the E. S. requires three phases, executed off-line:

- Acquisition of the hardware and software architecture of the system. The E.S. has to know the topology, the number of processors and their features such as processing speed, running costs, failure rate and the workload due to other applications already allocated or to daemon processes, etc. The E.S. also has to acquire information about the communication subsystem, in particular the running cost and failure rate.
- Acquisition of the model of the application provided in the form of an application graph. For each process the following are specified: the execution time on a reference processor, the communication time, the relationships with the other processes of the application task and the subset of processors on which it can be executed.
- Acquisition of the user-imposed constraints in terms of:
  1. the response time deadline;
  2. the reliability of the application as a whole and the number of replicas desired for certain processes (the most critical ones);
  3. the cost of the application as a whole in terms of both computational load and occupation of processing resources.

3.2 Mode of Operation

The E.S. proposed tries to allocate the processes making up the application task in such a way as to meet all the user's requirements. However, the time-critical features of a process control environment make response time a priority issue as compared with the other parameters. The E.S. will therefore find a possible trade-off between response time and reliability, always privileging the former. In addition, as guaranteeing reliability requirements in a process control environment always has priority over cost requirements, the latter will be penalized if it is not possible to satisfy both.

3.2.1 Response Time

When the initial configuration phase terminates, the rule block A is activated, which calculates the response time, reliability and cost of execution of the application task on a single processor. This block of rules gives an upper bound for the response time and a lower bound for reliability and cost. These values are used by the E.S. to evaluate the consistency with the specifications supplied by the user; if any inconsistency is found, the E.S. takes suitable steps to rectify it.
If, for example, the user has indicated a response time greater than the one obtained by executing the whole application task on a single processor, the E.S. sends a message to notify the user that, in view of the absence of significant time constraints, allocation will only take into account the constraints imposed on reliability and cost.
If, on the other hand, the user has asked for lower reliability and cost values than those obtainable with a monoprocessor system, the E.S. sends the user an error message.
In addition, on the basis of the difference between the response time and reliability values specified by the user and those referring to the monoprocessor system, the

E.S. can estimate the minimum number and type of processing nodes the user implicitly requires, and can thus guide the initial assignation in that direction. This estimate is based on the directions of the designer, who foresees the workload of the system as a whole, given a certain initial distribution of control applications. The designer, for instance, could choose to keep the system underloaded in view of immediate updating, or again he could decide to load it quite heavily, considering that the situation is already relatively complete and thus not immediately subject to modifications.
At the end of this phase, the rule block B is activated, which performs an initial allocation, aiming at minimizing the response time and based on the principle of parallel execution of independent processes, trying at the same time to assign processes linked by precedence relationships to the same processor. After having tried to assign a process Mi to each of the processors P_K eligible for the execution, the E.S. chooses the one providing the lowest response time by means of a heuristic formulation, a brief outline of which is given here in Figure 2.
If the resp_t(T) obtained with this allocation is greater than that required by the user, the E.S. backtracks to rule block B and re-assigns, using a greater number of processing nodes.

3.2.2 Reliability

Once the allocation which meets the user's time requirements has been determined, rule block C is activated, which calculates the reliability of the application task according to a heuristic formula which expresses the reliability of the application T, seen as the probability that it will be successfully executed, as the product of the reliability of the communication system and that of the processes it comprises (the probability that the modules are correctly executed) [9], i.e.

$$rel\_t(T) = rel\_c \cdot \prod_{i=1}^{m} rel\_m(M_i)$$

The factor rel_c represents the reliability of the communication subsystem, which depends on the failure rate, λ_C, of the communication system and the response time for the application resp_t(T) according to the formula

$$rel\_c = e^{-\lambda_C \cdot resp\_t(T)}$$

The factor rel_m(M_i), which represents the reliability of the process M_i, depends on the reliability of the node (or nodes) on which it is executed. In particular, if M_i has several replicas running on different nodes, it is considered to have been successfully terminated if at least one replica completes execution. Under this hypothesis, the subsystem of nodes which execute the replicas of M_i can be considered as a *parallel system* [9], and the reliability of the whole application T can therefore be expressed as

$$rel\_t(T) = rel\_c \cdot \prod_{i=1}^{m}\left[1 - \prod_{k=1}^{n}\left(1 - rel\_p(P_k)\cdot x_{ik}\right)\right]$$

where the quantity in round brackets represents rel_m(M_i). The reliability of the application as a whole therefore depends on the reliability of the single component processes.
If the reliability value obtained is lower than the one supplied by the user, rule block D, which introduces replication, is activated. This block of rules first tries to introduce a single replica per module, then it calculates the overall response time of the application and compares it with the user's specifications. If the user's time requirements are still met, rule block D reactivates rule block C (as illustrated above) which calculates the overall reliability of the task. If the value obtained is still lower than that requested by the user, rule block D is activated again to introduce more replicas, still of course respecting the time constraints imposed.
If, on the other hand, the reliability required by the user cannot be achieved, the user will receive a message informing him of the fact that the allocation obtained taking into account the time constraints he indicated has given certain response time and reliability values and asking whether he wishes to continue with these specifications or to go back and modify his requests. In the latter case various situations may occur.

1. If the user can make the response time less restrictive, the E.S. repeats the allocation procedure on the basis of the new specifications.
2. If, on the other hand, the user cannot modify his specifications (because, for instance, the dynamic of the application will not allow it) the E.S. offers him the chance to increase the overall reliability of the application by replicating only some of the processes it comprises. On the basis of his knowledge of the application, in fact, the user/designer knows the most critical processes and can therefore instruct the E.S. which ones to replicate and how many replicas to provide for each of them.
3. If (and this is more likely, given the real-time context), it is necessary to seek a trade-off between response time and reliability and the user prefers to give priority to the former, he will accept the reliability value indicated by the system.
4. Lastly, if the user can neither modify his requests nor accept the response time and reliability values obtained, the E.S. will make a fresh attempt, going back to the beginning and using a greater number of processing nodes (rule block G).

Response time for application T	$resp\_t(T) = \max_i ae\_t(M_i)$	M_i is the last-completed module of application T
Absolute elapsed time of module M_i	$ae\_t(M_i) = start\_t(M_i) + resp\_t(M_i)$	
Start_time for module M_i	$start\_t(M_i) = \max_P ae\_t(M_p)$	P is the set of M_p, modules of T which are immediate predecessors of M_i .
Response time for module M_i taking into account synchronization with other modules M_j	$resp\_t(M_i) = \max_C [resp'\_t(M_i), resp'\_t(M_j)]$	C is the set of modules M_j of T related to M_i by synchronization
Response time of M_i without taking into account synchronization	$resp'\_t(M_i) = \min_k resp'\_t(M_i)_k$	Index k (k=1..n) represents a generic node P_K
Response time of M_i when executed on node P_k	$resp'\_t(M_i)_k = comp\_t(M_i) + comm\_t(M_i)_k + \sum_{l=1}^{s} cs_{il} \cdot serv\_t(S_l)$	
Computation time for M_i on node P_k	$comp\_t(M_i) = \frac{D_i}{R_k} comp\_ref(M_i)$	
CPU time required by a reference node to cope with the computational load of M_i	$comp\_ref(M_i)$	
Relative speed of the node P_k as compared with that of the reference node	R_k	
Delay due to the fact that M_i shares node P_k with other processes.	$D_i = A_i + B_i$	
Delay due to the modules belonging to other applications already assigned to P_k	$A_i = \frac{a_k}{Pri(M_i)+1}$	a_k is number of modules belonging to other applications already assigned to P_k
Priority of M_i , intended not in the traditional sense, but as module ordering. Each module (except for the server modules) is given a priority number which depends on its position in the application graph (start-up modules will have a priority of 1, their direct successors will have a priority of 2 and so on).	$Pri(M_i)$	Priority of modules weights the impact of one module on another; modules with highly different priority execute at distant times and affect each other less than modules with similar priorities.
Delay due to the modules of application T assigned to P_k which can be executed concurrently with M_i	$B_i = \sum_{h=1}^{b_i} \frac{1}{\lvert Pri(M_h) - Pri(M_i)\rvert + 1}$	b_i is the number of active modules of application T assigned to P_k (including M_i)
Time for communication between module M_i and other modules of T not resident in P_k.	$comm\_t(M_i)_k = \sum_{j \neq i}^{m} \sum_{l \neq k}^{n} comm\_t(M_i, M_j) \cdot x_{ik} x_{jl}$	
Time required for communication between M_i and M_j	$comm\_t(M_i, M_j)$	
Binary variables equal to one if, respectively, M_i resides in P_k or M_j in P_l, otherwise zero.	x_{ik}, x_{jl}	
Number of times the client calls the server	cs_{il}	
Time spent waiting for server response	$serv\_t(S_l) = Q_l \cdot resp\_t(S_l)$	
Queue of S_l during execution of M_i (number of clients who in this period can require a service from S_l concurrently with M_i).	Q_l	

Figure 2: Summary of formulas

3.2.3 Cost

Once the user has accepted the reliability value obtained with the mechanism described above, rule block E is activated, which calculates the computational cost of executing the overall task, according to the following formula:

$$cost\_t(T) = \sum_{i}^{m} \sum_{k}^{n} cost\_p(P_k) \cdot comp\_t(M_i) \cdot x_{ik}$$

in which cost_p(P_k) is the unit running cost for the node P_k, *comp_t(M_j)* is the computation time for the process M_j and x_{ik} is 1 if the process M_i resides in the node P_k, or otherwise 0. This cost value can be used, for instance, for accounting functions or to evaluate the computational load and occupation of system resources required by the task. The cost thus obtained is compared with that indicated by the user in the specification stage: if it is lower, the allocation has been successful. If, on the other hand, it is greater, the E.S. asks the user if he will accept it; if he will not, he is invited to modify his requests or review the choices made previously to improve the reliability or response time. In this case, the E.S. tries to re-allocate according to the new user specifications.

4 AN EXAMPLE OF ALLOCATION

In this section we will describe the results obtained using the E. S. to assign the modules of a known distributed application to the nodes of a heterogeneous system.
The scenario is a petrochemical industry and the distributed process control application comprises a set of modules which implement regulation loops. Some of these modules are executed sequentially, others cooperate. Communication between the various processing nodes in the system is supported by a local high-speed network which guarantees low delivery times for messages.
Figure 3 shows the application graph; the number next to each module refers to its priority.
The processing and communication times for the modules are given in Table I, whereas Table II shows the parameters relative to the client/server relationships for the application in Figure 3. All these times are expressed in milliseconds, a suitable unit of measurement for this kind of application task.
The heterogeneity of the system is expressed in terms of different processing capacity on the part of the nodes. As faster nodes generally also have higher costs, we made this heterogeneity entail differences in the unit cost of utilization of the processing nodes. We also assumed that all the nodes have the same failure rate. Finally, since we are working with a real system, we inserted previously allocated modules belonging to other applications, already residing on the nodes, to which maximum priority is given.

The features of the system are summarized in Table III.

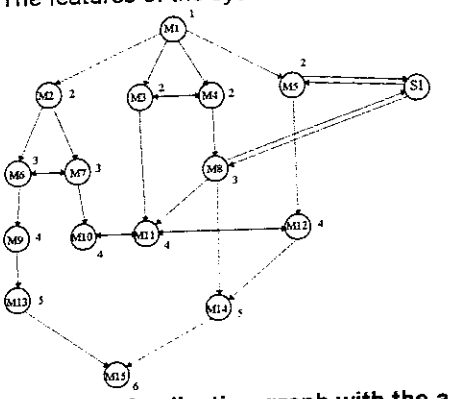


Figure 3: Application graph with the assigned priorities

	comp_t	comm_t	comm-t	comm-t	comm-t
M_1	14	M_2 - 1	M_3 - 1	M_4-2	M_5-2
M_2	14	M_1 - 1	M_6 - 1	M_7 - 1	
M_3	16	M_1 - 1	M_4 - 1	M_{11}-1	
M_4	14	M_1 - 2	M_3 - 1	M_8 - 2	
M_5	16	M_1 - 2	M_{12}-1		
M_6	13	M_2 - 1	M_7 - 2	M_9 - 2	
M_7	13	M_2 - 1	M_6 - 2	M_{10}-1	
M_8	15	M_4 - 2	M_{11}-2	M_{14}-2	
M_9	12	M_6 - 2	M_{13}-1		
M_{10}	14	M_7 - 1	M_{11}-2		
M_{11}	17	M_3 - 1	M_8 - 2	M_{10}-2	M_{12}-2
M_{12}	16	M_5 - 1	M_{11}-2	M_{14}-2	
M_{13}	13	M_9 - 1	M_{15}-1		
M_{14}	13	M_8 - 2	M_{12}-2	M_{15}-1	
M_{15}	14	M_{13}-1	M_{14}-1		

Table I: Processing and communication times for the application in Fig. 3

	comp_serv	comm_serv (cs)	comm_serv (cs)
S_1	8	M_5 - 2 (2)	M_8 - 2 (2)

Table II:Times relative to the client/server relationships in Figure 3

	R_k	cost_p	λ_{wsk}	a_k
P_1	1	10	0.0001	1
P_2	1.5	15	0.0001	2
P_3	2	20	0.0001	3
P_4	1	10	0.0001	1
P_5	1.5	15	0.0001	2
P_6	2	20	0.0001	3
			λ_c	
Comm. subsystem			0.00001	

Table III: Parameters of the heterogeneous distributed system.

We will now illustrate how the number of processors used by the E. S. varies as the user-specified response time and reliability vary.

4.1 Analysis of Results Obtained with Varying Numbers of Processing Nodes

Figure 4 illustrates the dependence of the number of processing nodes employed on the user's response time specifications.
As can be seen, until the user-specified response time goes below 350 ms, the E.S. uses a single processor to execute the whole task. As the user-indicated response time descends below 350 ms, the number of processors used rises, up to a maximum of three, which provide a response time of 150 ms. As it is not possible to reduce the response time any further in the application being considered, a request of this kind on the part of the user (asking for a response time of 100 ms, for instance) will not make the E. S. try to use all the processors; the user will be notified of the impossibility of achieving this aim.

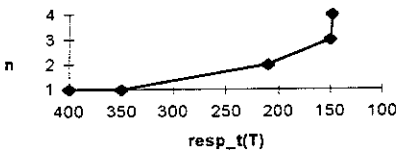


Figure 4: Number of nodes versus response time

On the other hand, as can be seen in Fig.5, which shows how the number of processors used varies according to the reliability required by the user, an increase in the number of processors used from three to four causes an actual increase in reliability. So the E. S. will ask the user if he wishes for the maximum reliability value, in which case it will choose four processors, or whether he prefers to privilege cost, in which case it will choose only three processors.

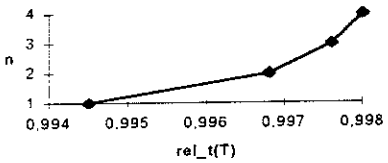


Figure 5: Number of nodes versus reliability.

5 CONCLUSION

In this paper we have described an Expert System for planning the task allocation of a real-time distributed application. We have discussed the allocation criteria used in the construction of the rules. The functioning of the E. S. has been tested on various cases of application, one of which is presented as an example. Performance evaluation has been made by simulation, considering a system of six processing nodes with different calculation power. The results obtained show the capacity of the E.S. to find solutions which, when possible, meet the specifications supplied by the user. If these specifications are not compatible with the system being considered, the E.S. notifies the user of this incompatibility, indicating the best performance obtainable for the specific application.

6 REFERENCES

[1]P.Pleinevaux, J. D. Decotignie, "Time critical communication neyworks: Field Buses", IEEE Network, May 1988, vol.2, no.3.

[2] N. S. Bowen, C. N. Nikolaou, A. Ghafoor, " On the Assignment Problem of Arbitrary Process Systems to Heterogeneous Distributed Computer Systems ", IEEE Trans. on Computers, Vol. 41, no. 3, pp. 257-273, March 1992.

[3] D. L. Eager, E. D. Lazowska, J. Zahorjan, " A dynamic load sharing in homogeneous distributed systems ", IEEE Trans. Software Eng., Vol. SE -12, no. 5, pp. 662-675, May 1986.

[4] W. W. Chu, L. J. Holloway, M. T. Lan, K. Efe, " Task allocation in distributed data processing ", IEEE Comput. Mag., pp. 57-69, Nov. 1980.

[5] S. M. Shatz, J. P. Wang, M. Goto, " Task Allocation for Maximizing Reliability of Distributed Computer Systems ", IEEE Trans. on Computers., Vol. 41, no. 9, pp. 1156-1168, Sept 1992.

[6] V. M. Lo, "Heuristic Algorithms for Task Assignment in Distributed Systems", IEEE Trans. on Computers, Vol. 37, no. 11, pp. 1384-1397, Nov. 1988.

[7]C. J. Hou, K. G. Shin, "Load Sharing with Consideration of Future Task Arrivals in Heterogeneous Distributed Real-Time Systems", IEEE Trans. on Computers, Vol. 43, no. 9, pp. 1076-1090, Sept. 1994.

[8]O. Kremien, J. Kramer, "Methodical Analysis of Adaptive Load Sharing Algorithms", IEEE Trans. on Parallel and Distributed Systems, Vol. 3, no. 6, pp. 747-760, Nov. 1992.

[9] J. G. Rau, "Optimization and Probability in Systems Engineering", Van Nostrand Reinhold, 1970.

Student Responses and Follow Up Tutorial Tactics in an ITS*

Gregory Hume
Valparaiso University
ghume@exodus.valpo.edu

Joel Michael
Rush Medical College
jmichael@steve.iit.edu

Allen Rovick
Rush Medical College
aar@steve.iit.edu

Martha Evens
Illinois Institute of Technology
mwe@schur.math.nwu.edu

Abstract

We report on an ongoing study of experienced tutors and the development of an Intelligent Tutoring System (ITS). Human tutors monitor the progress of their students. An ITS records this information in a student model. Human tutors consider their student model when determining both *what* is to be tutored and *how* it is to be tutored. We have identified five tactics that draw the student into different levels of cognitive activity. Passive tactics require that students absorb information while active tactics encourage students to discover concepts. In determining *what* topic should be tutored, the correctness of student responses to problems or questions is an obvious input to the student model (whether it is a human tutor or ITS). In determining *how* the topic should be tutored (i.e., what tactic), the student's ability to respond appropriately is an input to the student model. For example, it is better that a student recognizes that a hint was provided, even if a correct answer is not forthcoming, than to not recognize the intention of the tutor. We have identified an algorithm our tutors use to select follow up tactics.

INTRODUCTION

CIRCSIM-Tutor (CST) is an Intelligent Tutoring System (ITS) that employs natural language via the screen and keyboard to communicate with students (Kim et al., 1989). We have conducted human-to-human, keyboard-to-keyboard tutoring experiments (Li et al., 1992) to better understand the dynamics of tutoring. From these experiments, we have identified and characterized our tutors' frequent use of hints (Hume et al., 1993; 1996). We subsequently identified some other tactics our tutors employ and estimated the cognitive activity they elicit from the student (Hume et al., 1995). This paper reports our findings regarding the process our human tutors use to determine what tactic to employ.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/168 ©1996 FLAIRS

CIRCSIM-TUTOR

The domain of CST is cardiovascular physiology. Our subjects (first year medical students) are given a description of a perturbation (a disturbance to the human circulatory system) and are asked to reason about the ensuing causal effects. The protocol of the human-to-human (via computers in separate rooms) tutoring sessions is essentially the same as the human-to-ITS sessions (students know when they are communicating with a human). In both cases, students are paid volunteers and all dialogue is recorded on disk.

There are three levels of causal relationships represented in our physiology knowledge base. The top level contains those relationships the student must understand in order correctly predict behavior of the human circulatory system when it encounters a perturbation. For example, one top level relationship is that when the cardiac output (CO) increases, the mean arterial pressure (MAP) increases. Many of these relationships are counter-intuitive and difficult to master. The intermediate level variables can be used to further describe top level relationships. For example, the following two intermediate level relationships expand the CO/MAP relationship: (1) when CO increases, the arterial blood volume (ABV) increases and (2) when ABV increases, MAP increases. The goal of CST is to enable students to reason correctly at the top level. This intermediate level knowledge is used to construct hints (see below). Likewise, the deep level contains variables that expand the intermediate level relationships. Relationships at the two deeper levels are used for the construction of explanations and hints.

*Acknowledgment: This work was supported by the Cognitive Science program, Office of Naval Research under Grant N00014-94-1-0338 to the Illinois Institute of Technology. The content does not reflect the position or policy of the government and no official endorsement should be inferred.

HINTS AND OTHER TUTORIAL TACTICS

Previously we have recognized that our experienced tutors regularly use hints. A hint is defined as (Hume et al., 1993, p. 564):

> a rhetorical device that is intended to either: (1) provide the student with a piece of information that the tutor hopes will stimulate the recall of the facts needed to answer the question, or (2) provide a piece of information that facilitates the student's making an inference needed to arrive at an answer to a question or the prediction of system behavior.

We then identified two categories of hints (Hume et al., 1993): hints that convey information (CI-Hints) and hints that point to pertinent information but do not explicitly convey information (PT-Hints). We have assumed that tutors use hints because they force the student to be active in the learning process. We have also assumed that PT-Hints require a greater degree of cognitive activity on the part of the student than do CI-Hints. This is because the student must reflect on why the pointed to information is pertinent.

Finally, we have identified three other tactics; each draws the student into a different level of cognitive activity (Hume et al., 1995). The first of these tactics is a sequence of simple questions and answers that guide a student through a solution. We refer to this as a directed line of reasoning (DLR). This tactic allows the student to be less active than with CI-Hints. Another tactic is the generation of a summary. In a summary, the tutor reminds the student of material that has been discussed. This didactic tactic allows the student to be somewhat passive; they only need to integrate the information being summarized. Finally, the most passive tactic is an explanation. We are, of course, just estimating degrees of cognitive activity; any one utterance may trigger, in the student, a complex sequence of reasoning.

The five tactics described here can be positioned along a continuum spanning passive to active learning (Hume et al., 1995; see Figure 1.). Passive tactics require that the student store information. Active tactics require the student to integrate information.

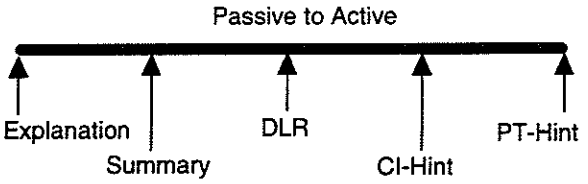


Figure 1. The Passive to Active Learning Continuum

THE STUDENT MODEL

The traditional use of an ITS's student model is to help determine *what* to tutor. Our tutoring protocol (CST and human experiments) allows the student to make several uninterrupted predictions about system behavior; the tutor does not immediately intervene once a prediction error is made. This allows the tutor to recognize patterns of errors and to rank order the severity of the problems these patterns suggest. A topic to be tutored is selected. At this point, a tactic is selected to assist the student in remediating misconceptions. We believe that the student model should help in this selection. In other words, the student model should help determine *how* to tutor.

We have concluded, from interviews with our experienced tutors, that human tutors keep only a very coarse-grained assessment of the student's cognitive activity. They tend to form an overall assessment of the student; we refer to this as the global assessment. They also assess the student's performance while working on a phase of a problem; we refer to this as the local assessment.

The student model (human tutor or ITS) is affected positively when a student responds to a problem or question with a correct answer. However, through observations and interviews we have determined that there are other factors that experienced tutors use in modelling the progress of a student. The ability of a student to respond appropriately to hints is a major determinant of the tutor's selection of tactics. Many types of student responses, not just a completely correct response, may lead the tutor to believe that the student has a good understanding of the topic (see Table 1).

SELECTION OF TACTICS

After the selection of a topic to tutor, our tutors usually ask a question that is designed to verify an assumption about the nature of the student's understanding of the material. When an error is present, however, they will (virtually always) provide one or more hints.

Our assumption has been that tutors will use tactics (specifically, hints) that force the student to be cognitively active so long as the student continues to respond positively. We have recorded and analyzed sequences of tutor utterances and student responses in order to verify this assumption. Some of our findings regarding the use of hints by experienced tutors in our domain are:

1. Tutors almost always provide at least one hint when the student makes an incorrect prediction(s)
2. Only when a student has demonstrated that he/she
   a. is deficient in prerequisite knowledge and/or
   b. is regularly unable to understand the tutor's intention

   does the tutor resort to an explanation as the first tactic when tutoring a topic.

At this time, we have been unable to observe a regular pattern for selecting what type of hint is used as the first hint (PT- Hint or CI-Hint). We can only attribute the choice of the first hint to the personal style of the tutor. One of our tutors tends to use a PT-Hint as the first hint; the other tends to use a CI-Hint as the first hint. However, we observed regular patterns for subsequent hints. It appeared that the student response dictates what type of follow up hint, or other tactic, is employed. Therefore, we identified categories of student responses to hints and questions (Table 1).

A correct response is, of course, a response that will promote a positive local assessment by the tutor. Likewise, an incorrect response, or a response that suggests that the student was unaware that a hint was provided, will promote a negative local assessment. It follows that some responses will tend to produce a partially positive assessment in the tutor's mind. For example, a student who responds with a question to clarify the tutor's intent may have either displayed (1) a partial understanding of the topic at hand and/or, (2) some understanding of the tutor's intention.

A	Incorrect.
B	Partially correct.
C	Correct.
D	Not incorrect, but it does not address the issue at hand.
E	Clarification question.
F	Explicit statement that he/she does not know.
G	Inappropriate response. This may suggest that the student does not recognize the tutor's intention.

Table 1. Categories of Student Responses.

Our tutors continue to provide hints on a particular topic as long as the local assessment does not become too low. In other words, the tutor will not resort to an explanation because of just one negative response. They tend to provide PT-Hints (provoking the most active thought) when the local assessment is very high (following correct and other positive responses). CI-Hints (more directive) are provided when the local assessment starts to decrease. Our tutors will resort to summaries and explanations when the local assessment becomes substantially lower. The global assessment determines these local assessment threshold values.

In our experiments, the sequences of tutor prompts (hints and questions) and student responses (we refer to such a sequence as a string of hints) are usually terminated by either (1) a correct student response or (2) a tutor's explanation. A string of hints usually guides the student to a discovery of the intended concept (in approximately two thirds of the sequences we have observed). The strings of hints terminated by an explanation tend to be very short. In other words, students made poor responses early.

EXAMPLES FROM HUMAN TUTORING

TRANSCRIPTS

The following excerpts have been edited slightly for readability. Both tutor and student understand that '0' means unchanged, 'D' means decreases and 'I' means increases.

T1: This is the DR. How will HR change?
S1: MAP changing affects BR changing, affecting HR
T2: In DR no reflex changes have occurred yet.
S3: So HR will not change.

Dialogue 1. Example of Student Responses.

In T1 of Dialogue 1, the tutor provides a PT-Hint. The tutor alludes to the phase of the response without providing an explanation. In S1, the student offers an incorrect response. In T2, the tutor provides a CI-Hint The incorrect answer caused the tutor to select a tactic that is slightly less active on our the passive to active continuum (see Figure 1). The student's response in S2 is correct and the topic is terminated. In this situation, the tutor's global assessment was sufficiently high so that hinting was continued after an incorrect response.

In T1 of Dialogue 2, the tutor provides a CI-Hint. He produces an explanation and prompts the student for a reply. The student gives the intended reply in S1. The tutor responds with a PT-Hint in T2 by alluding to other variables. The tutor hopes that the student will recognize that the physiological variable CC will behave exactly the same as the physiological variable TPR discussed in T1. The student, in S2, correctly answers the explicit question but does not recognize the tutor's subtle intention. The tutor responds with a CI-Hint in T3. The student provides an incorrect answer in S3. The tutor, believing the student may still self-discover the concept, provides another CI-Hint in T4. The student again replies with an incorrect response in S4. The tutor now simply provides an explanation in T5. The local assessment had become too low.

T1: TPR is primarily under neural control. We're talking about what happens before there are any neural changes. Now what do you say about TPR?
S1: 0
T2: Right. So let me ask you, are there any other of these variables that are primarily under neural control?
S2: CC, HR
T3: Right again. But we know in this problem, HR is solely under the control of the artificial pacemaker. So let's leave it out. But if CC is under neural control, how would it be affected in the DR period?
S3: I EDV
T4: You can't have it both ways. If CC is under neural control and we're talking about the period before any change in neural activity then CC???
S4: But, it is ALSO under intrinsic control
T5: You are confusing Starling's Law with a change in contractility. The length/tension relation of the heart is not a change in contractility. A change in contractility moves the length/tension curve from one location to another. Increased contractility means that at a given EDV you get more contractilt performance out of the ventricle.

Dialogue 2. An Example of a Low Local Assessment.

STRINGS OF HINTS

To identify an algorithm for follow up tactics, we defined a string of hints as:

1. Commencing with the utterance of a first hint (e.g., T1 on Dialogue 1 and 2).
2. Continuing with a sequence of student responses and tutor prompts. Some of these prompts may contain hints, the others are only questions.
3. Terminating because either:
   a. The student has demonstrated an understanding of the topic by correctly answering hints and questions, or by changing predictions.
   b. The local assessment becomes too low. Again, the threshold value for the local assessment is determined by the global assessment. The topic is concluded with a summary or an explanation

We examined 153 strings of hints. Table 2 lists the frequency of final student responses in those strings of hints. Our tutors have a clear tendency to prompt the student to the point where the student has demonstrated an understanding of the topic. When this fails, usually signaled by an incorrect response, our tutors resort to an explanation (a passive tactic).

Student Response:	Frequency
Incorrect	27
Partially Correct	5
Fully Correct	109
Did Not Address Tutor's Intention	2
Clarification Question	5
Does Not Know	4
Does Not Recognize Question	1

Table 2. Frequency of a Student's Final Response in a String of Hints.

We suspect that CI-Hints and PT-Hints serve different purposes, but it does not appear that one tactic is more effective than the other. Of the 109 strings that were terminated with a fully correct response (see Table 2), 62 times the final hint was a CI-Hint versus 47 times for a PT-Hint. PT-Hints, however, produced more fully correct immediate responses. Of the 295 hints identified in the 153 strings of hints, 151 times the immediate response was fully correct. PT-Hints produced an immediate fully correct response 79 times versus 72 times for CI-Hints. Determining the effectiveness of hints is difficult for two reasons. First, it is impossible to completely reconstruct the tutor's intentions and instructional goals. Second, hinting may have long term benefits more significant than the correctness of immediate responses.

A very interesting pattern emerges upon analysis of the subsequent hints in a string of hints. A CI-Hint, by definition, provides explicit information. It appears our tutors gauge the student's need for explicit information. If a student displays a higher degree of difficulty (e.g., an incorrect response), then the tutor is likely to provide a CI-Hint. On the other hand, a student that shows some progress (e.g., a partially correct answer) is likely to receive a PT-Hint. Table 3 shows the frequency of follow up hints (PT-Hint or CI-Hint) after a student response.

	Incorrect	Partially Correct	Fully Correct
CI-Hint	36	6	17
PT-Hint	16	11	35

Table 3. Types of Hints that Follow a Response.

It should be noted that a completely correct response does not necessarily terminate a topic. The tutor may partition the topic into sub-topics.

The following is a summary of the rules we have identified for determining what tactics to use:

1. If an error is made, provide a hint. The exception is when the global assessment is very low.
2. If (a) the student has not discovered the concept and (b) the global assessment continues to be sufficiently high and (c) the number of hints in a string is sufficiently low, try subsequent hints.
3. If a follow up hint is to be provided then use a PT-Hint when the local assessment is sufficiently high. Otherwise use a CI-Hint.

CONCLUSION

We have analyzed the tutoring behavior of experienced cardiovascular physiology tutors. Their hinting behavior influences their modelling of the student; their model of the student influences their hinting behavior. The key elements of their behavior are:

1. They allow students to make several predictions before they engage in an interactive dialogue. This allows them to identify key patterns of errors (error patterns).
2. They use a very coarse grained scheme (the student model) to evaluate students. They form a global assessment of the student, a measure of the student's behavior throughout the tutoring session. They constantly form local assessments, an assessment of the student throughout tutoring on a given topic. As the local assessment changes, the global assessment is updated.
3. They attempt to use tactics that promote active learning. Specifically, they regularly use hints. The use of didactic tactics (e.g., explanation) is used when there is evidence that hinting will not be productive.
4. They rely heavily on their global and local assessments to determine when and how long to hint.
5. While hints come in many surface forms, there are two significantly different categories of hints: convey information hints (CI-Hints) and point to information hints (PT-Hints).
6. While there are some differences in the patterns of hinting between our tutors, there are many similarities. The differences are due to personal style; the similarities are due to the functions of CI-Hints and PT-Hints.

Our tutors use their global and local assessments to select tactics (*how* to tutor). They want students to discover concepts; the use of didactic tactics is a last resort There is some anecdotal evidence, from our human tutoring experiments, that hinting is an effective tactic. This needs to be verified. We would also like to study other tutoring domains to generalize out findings. CST's knowledge base and student model are fully implemented; other modules are partially implemented. The language modules need further enhancements to make our ITS robust.

REFERENCES

Hume, G., Michael, J., Rovick, A., & Evens, M. (1993). The use of hints as a tutorial tactic. Proceedings of the 15th Annual Conference of the Cognitive Science Society (pp. 563-568). Boulder, CO.

Hume, G., Michael, J., Rovick, A., & Evens, M. (1995). Controlling active learning: how tutors decide when to generate hints. *Proceedings of the 8th Florida Artificial Intelligence Research Symposium* (pp. 157-161). Melbourne Beach, FL.

Hume, G., Michael, J., Rovick, A., & Evens, M. (1996). Hinting as a tactic in one-on-one tutoring. *The Journal of Learning Sciences*, 6, 23-47.

Kim, N., Evens, M., Michael, J. A., & Rovick, A. A. (1989). CIRCSIM-TUTOR: An intelligent tutoring system for circulatory physiology. In H. Maurer (Ed.), Computer Assisted Learning: Proceedings of the Second International Conference on Computer-Assisted Learning, ICCAL '89 (pp. 254-266). Dallas, TX.

Li, J., Seu, J., Evens, M., Michael, J., & Rovick, A. (1992). Computer dialogue system (CDS): A system for capturing computer-mediated dialogue. Behavior Research Methods, Instruments, & Computers, 24, 535-540.

A COMPUTER-AIDED APPROACH TO SPC

Chung-Yu Pan
Tunghai University
pancy@ie.thu.edu.tw

Abstract

SPC has been recognized as a very useful tool in monitoring a production process. The link between the production process and the control procedure is highly dependent on humans for manipulation and reaction, and is difficult to be replaced by non-human devices. "On-line Process Control Rules" which is expected to monitor the condition of a process is developed in this paper. Those rules would make the linkage replaceable and be able to eliminate the delay of reaction in traditional process control .

INTRODUCTION

Control charts and inspection rules which are utilized in statistical quality control (SPC) were proposed by Shewhart in 1931. Since then, SPC has been recognized as a very useful tool in monitoring a production process and widely applied to various manufacturing process to control product quality. The information processing of SPC is based on judgment, experience, and intuition of the inspection personnel. When the inspection personnel monitor and diagnose, based on the control chart, that the process is deteriorating, he/she takes actions to find and solve the problem. is highly dependent on humans for manipulation and reaction, and is difficult to be replaced by non-human devices. The progress in science and computer technology makes automated inspection available and important. However, traditional inspection rules are not adapted to automation. It is necessary to develop the "On-line Process Control Rules", which is expected to monitor the condition of a process and make an immediate decision to avoid the production of a next defective. The developed rules would automatically make the judgement of quality control of ongoing processes and be able to eliminate the delay of reaction in traditional process control.

According to Groover's (1987) definition of on-line procedure, it can be distinguished as on-line/in-process and on-line/post-process. On-line/in-process inspection is to inspect products during the manufacturing operation, meanwhile, on-line/post-process inspection is to inspect products immediately following the production process. Even through the later inspection procedure follows the process, it is integrated with the manufacturing process and its result can immediately influence the production operation. The On-line Process Control Rules which are developed in this paper may be applied to either on-line/in-process or on-line/post-process situation. It is assumed that observations were independent and normally distributed. Moreover, the test of the hypothesis of the multi-population proportion was applied with a type I error of 0.05 through the development of rules. Finally, two examples by using the technique of image process are made to reveal the practical application of those rules.

ON-LINE PROCESS CONTROL RULES

Since observations were normally distributed with an average of μ and a standard deviation of σ, they could be divided into eight regions. According to the property of normal distribution, the proportion of observations associated with its region are as follows:

p_1=0.00135, $\mu+3\ \sigma \leq$ obs..

p_2=0.02135, $\mu+2\ \sigma \leq$ obs. $\leq \mu+3\ \sigma$.

p_3=0.13600, $\mu+1\ \sigma \leq$ obs. $\leq \mu+2\ \sigma$.

p_4=0.34130, $\mu+0\ \sigma \leq$ obs. $\leq \mu+1\ \sigma$.

p_5=0.34130, $\mu-1\ \sigma \leq$ obs. $\leq \mu+0\ \sigma$.

p_6=0.13600, $\mu-2\ \sigma \leq$ obs. $\leq \mu-1\ \sigma$.

p_7=0.02135, $\mu-3\ \sigma \leq$ obs. $\leq \mu-2\ \sigma$.

p_8=0.00135, $\mu-3\ \sigma \geq$ obs..

If observations of i^{th} region were concerned and the proportion is p_i, then the remaining proportion of observations is 1- p_i. Therefore, the property of binomial distribution would be employed for all of the observations. Based on traditional inspection rules, in this paper 12 process control rules have been developed which are:

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/173 ©1996 FLAIRS

1. $n_1 \geq 1$, an unusual situation in the process.
2. $n_8 \geq 1$, an unusual situation in the process.
3. $n_3 \geq 11$ or $n_3 \leq 3$, an unusual situation in the process.
4. $n_6 \geq 11$ or $n_3 \leq 3$, an unusual situation in the process.
5. $n_4 \leq 11$, an unusual situation in the process.
6. $n_5 \leq 11$, an unusual situation in the process.
7. $n_1+n_2 \geq 3$, an unusual situation in the process.
8. $n_7+n_8 \geq 3$, an unusual situation in the process.
9. $n_4+n_5 \leq 29$, an unusual situation in the process.
10. $\bar{x}_t \leq CL - 1.28(\frac{USL - LSL}{6})$ or
    $\bar{x}_t \geq CL + 1.28(\frac{USL - LSL}{6})$,
    an unusual situation in the process.
11. $b_t > 1$ or $b_t < -1$, an unusual situation in the process.
12. $x_t > USL$ or $x_t < LSL$, an unusual situation in the process.

where n_i : number of sampling statistics fall within i^{th} region.
x_t : sampling statistics in time t.
b_t : slope of regression in time t.
$\bar{x}_t$: average of samples in time t.
USL : upper specification limit.
LSL : lower specification limit.
CL : central of specifications.

DISCUSSION

Control charts for variables are the basis for the developments in this paper. A process variable has two factors which must be considered which are expected value and dispersion. The expected value should be controlled and maintained as close as possible to the desired value of that variable. The dispersion of the variable which indicates how far the individual values deviate from the expected value should be tracked closely in order to keep it as small as possible. The purpose of setting up a control chart is to track the expected value and the dispersion over time. An x-bar control chart is basically applied to track the expected value. For some variable x, this chart is established by taking subgroups of reading of x values (i.e. observations), where each subgroup typically consists of 4-10 individual readings. The average of each subgroup called x-bar is plotted on the chart. According to the central limit theorem, these values of x-bar should be consistent with the property of normal distribution despite the distribution of x.

Likewise, a range control chart (R chart) is established to track the dispersion. A range of each subgroup determines when the x-bar was calculated. A range is the difference between the largest x and the smallest x within each subgroup. These range values are treated and examined in the same way as x-bar values. In general, three horizontal lines are drawn on a control chart. The Central line is located at the average value of all the charted values (the x-bars or the ranges). The Upper control limit and the Lower control limit are drawn at plus and minus three standard deviations from the Central line.

The x-bar and range values should appear evenly above and below the Central line of each chart. Moreover, all the values should lie between the upper control limit and the Lower control limit with a majority of values grouped near the Central line. In general, a process is considered in control if the points in the chart look random and reveal no trends or cycles to the eye for both x-bar and R charts. For a detail performance of control charts refer to Grant (1988).

Seven situations in a process proposed by Cesarone (1991) were used to evaluate these process rules. The process situations and test results are shown in Table 1 and described as follows:

1. IC -1/2 : the rules did not raise any alarm either in the expected value or in the dispersion.
2. Point : the rules detected unusual situations and raised alarm on 75^{th} subgroup.

Table 1 Evaluation of the On-line process control rules through the test conditions of Cesarone's model

Example	Comment	Probability of detecting	Average time of detecting
IC - 1	In control, first example	0.0	--
IC - 2	In control, second example	0.0	--
Point	Spurious point added to middle of data	1.0	75.0
Shift	Mean decreases by one standard deviation after 40 samples	1.0	81.3
Drift - 1	Mean increases gradually by one standard deviation during base period	1.0	46.6
Drift - 2	Mean decreases gradually by one half standard deviation during base period	1.0	61.1
Spread - in	Standard deviation gradually decreases during base period	1.0	28.8

where the time unit is measured by number of drawing of subgroups.

Table 2 Comparison between results of the On-line process control rules and Cesarone's model

Example	Comment	the On-line process control rules	Cesarone's model
IC - 1	In control, first example	--	In, CF=100
IC - 2	In control, second example	--	In, CF=100
Point	Spurious point added to middle of data	Out, CF>95	Out, CF=80
Shift	Mean decreases by one standard deviation after 40 samples	Out, CF>95	Out, CF=99
Drift - 1	Mean increases gradually by one standard deviation during base period	Out, CF>95	Out, CF=96
Drift - 2	Mean decreases gradually by one half standard deviation during base period	Out, CF>95	Out, CF=99
Spread - in	Standard deviation gradually decreases during base period	Out, CF>95	Out, CF=96

3. Shift : the rules detected unusual situations and raised alarm on 81.3^{th} subgroup.
4. Drift-1 : the rules detected unusual situations and raised alarm on 46.6^{th} subgroup.
5. Drift-2 : the rules detected unusual situations and raised alarm on 61.1^{th} subgroup.
6. Spread : the rules detected unusual situations and raised alarm on 22.8^{th} subgroup.

The developed rules could accurately detect these situations in a process as mentioned above. In addition, the confidence factor (abbr. CF) of each test situation is very close to what it was in Cesarone's model as shown in Table 2. A confidence factor is a numerical measurement of how true or false a fact is.

Comparisons were made between the On-line process control rules and traditional control charts as well as Cesarone's model, and were described as follows.

1. Comparison between the developed On-line Process Control Rules and traditional control charts.
   On traditional control charts, sampling intervals and subgroup size have a great impact on detecting an unusual condition in a process. The detecting procedure is discrete due to the sampling statistics. As to the rules developed, if 50 was the subgroup size then the rules could operate right after the first 50 samples and could continue from the 2^{nd} to the 51^{st} sample and so on. It is a continuous detecting procedure. An alarm will be raised immediately whenever there is an unusual situation in the process. Furthermore, the developed rules are used on-line which is capable to make an automatic judgement of quality control of ongoing processes and is able to do without the delay of reaction in traditional process control.
2. Comparison between the developed On-line Process Control Rules and Cesarone's model.
   In Cesarone's model, samples had to take away from the process and computed for the sampling statistics. It is a type of off-line inspection. Both methods in the accuracy of detecting an unusual situation in a process are equivalent. Moreover, the developed rules is a type of on-line inspection. Without any delay of processing and inspection the developed rules is quicker than Cesarone's model.

APPLICATION

Two applications were made to verify the validity of the On-line process control rules. Two sheets of paper with different colors (light grey and blue) were selected. A scanner and image process technique was applied to obtain the grey scale value of each color and then the developed rules was applied to verify whether the color was evenly dyed in the process. The light gray paper has been verified evenly dyed by an optical instrument. Meanwhile some tiny white spots are visible on the blue paper.

Regarding the light gray paper, 100 samples were drawn with a sample size of 25. The grand average was estimated as 20.474 and the standard deviation of the samples was 0.884. The statistics of these samples were plotted on a traditional x-bar control chart shown as in Fig. 1. According to inspection rules, the process had run in an appropriate condition and papers were dyed evenly.

Likewise, the grand average of the blue paper was estimated as 8.725 and the standard deviation of the samples was 0.318. A traditional x-bar control chart was set up shown as in Fig. 2. According to inspection rules, the process had not run appropriately. In fact, the rules developed detects an ordinary condition in the process at the 50^{th} sample.

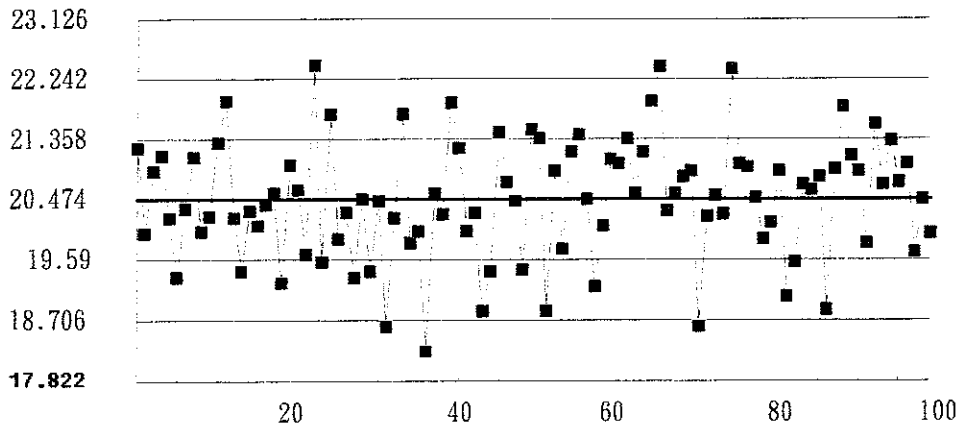


Fig. 1. x-bar control chart for light grey paper

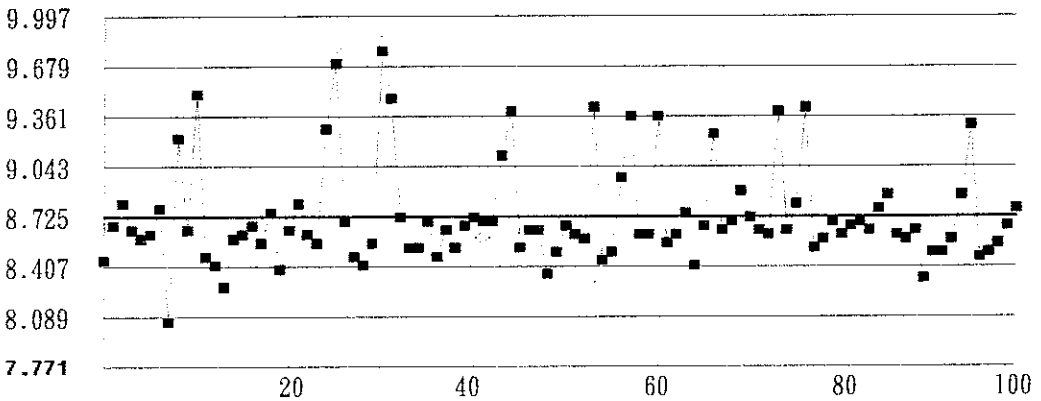


Fig. 2. x-bar control chart for blue paper

CONCLUSION

Generally speaking, the On-line Process Control Rules functioned very well on the test data. Both the in-control example, the light grey paper, was verified as an appropriate process, and the out of control example, the blue paper, was detected with an out of control situation to some extent. In viewing the applications, the On-line Process Control Rules have the same functions as traditional control charts. Moreover, they detect an undesired situation quicker than the traditional method.

The On-line Process Control Rules show that an automated diagnosis in the production process is possible. Yet it is still a work in progress in that, it is necessary to make the assumptions closer to the reality and apply to real data from actual process to check if it performs as well as it did in this paper.

References

Cesarone, J. QEX: An In-Process Quality Control Expert System, *Robotics & Computer-Integrated Manufacturing*, Vol. 8, No 4, pp.257-264, 1991.

Grant, E. L. and Leavenworth, R. S. *Statistical Quality Control*, Singapore, McGraw-Hill.

Groover, M. P. *Automation, Production Systems, and Computer Integrated Manufacturing*, Englewood NJ, Prentice-Hill, 1987.

Acknowledgment

The research is supported by National Science Council, Taiwan, R.O.C. under the contract No. NSC 84-2213- E -029-004.

APPLYING THEORY INTERPRETATION TO KNOWLEDGE INTERCHANGE

Bernd Bachmann
German Research Center for AI (DFKI)
bachmann@dfki.uni-kl.de

Abstract

One of the major obstacles for reusing and sharing knowledge-based systems is the fact that each system is constructed using a domain model best suitable for the task at hand thus idiosyncratic compared to similar problems in the same domain. In order to overcome this problem we will present an approach based on theory interpretation that tries to bridge the gap between the underlying conceptualizations of different knowledge bases. The methodology enables the exchange of messages between heterogeneous knowledge–based systems by applying semantics preserving transformation steps.

1 INTRODUCTION

Reusing knowledge–based systems (KBSs) for constructing new ones has been identified as one of the key technologies to reduce the costs during the construction phase and to overcome the knowledge acquisition bottleneck [10]. However, methodologies to achieve this goal are still in their infancy. One of the major obstacles that hinders a straightforward reuse of existing KBSs is the fact that the domain of discourse is modeled and structured best suitable to enable an adequate representation for the problem at hand. However, a similar problem in the same domain but with different requirements concerning the right level of abstraction, an adequate structure of the world or idiosyncratic terminology always demands a complete reinvention of the domain model and its respective representation.

Within the context of knowledge sharing the development of domain *ontologies* it used to establish a means for a common understanding of knowledge among different agents. More specifically, an ontology (of a domain) may be regarded as the explicit definition of the conceptualization that is shared among different agents. *Conceptualization* is used in the sense of [6]: the basic sets of objects that exist in a domain and the relationships among them. Consequently, all knowledge that is represented on the basis of the same conceptualization using the same language (an interlingua) can be exchanged among agents without any translations. Within AI the term (domain) *ontology* denotes a set of definitions that explicitly specifies a conceptualization together with the semantics of the identifiers in use.

The basic assumption in this research community—problem-independent, common ontologies can be identified—at least must be challenged if we consider the situation where already existing knowledge and data is considered as sharable. Typically the knowledge bases or the respective systems have not been constructed with the intention for a later reuse. The underlying ontologies were defined with respect to a problem–specific terminology, a suitable abstraction level, and and adequate granularity of the domain. A reuse of such systems would either require to adapt the ontology to the new problem or to provide a means to translate from the given ontology to the one we have in mind to capture the domain of the new problem.

Consequently, there is a need for combining ontologies of the same to domain but with different underlying conceptualizations as a prerequisite to achieve knowledge sharing and reuse as a standard technology.

2 PROBLEM DESCRIPTION

We are basically focusing a dynamic reuse and sharing approach that regards KBSs as deductive machineries with a "black box" behavior. As a direct consequence

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/177 ©1996 FLAIRS

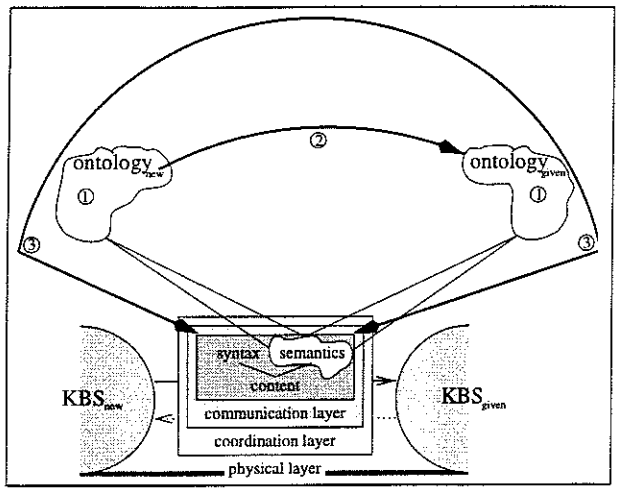


Figure 1: A communication model and the principle solution approach

these kinds of knowledge bases enable a functional view that can be characterized in terms of what they can be asked or told, not in terms of the particular structures that are used to represent its knowledge [8]. Based on this model the reuse scenario can be described as follows:

> We consider a situation in which a knowledge–based system is going to be constructed (KBS_{new}). In order to exploit the idea of sharing and reuse we assume the existence of another KBS (KBS_{given}) that already provides some of the reasoning capabilities needed to solve the task of KBS_{new}. The overall objective is to provide a methodology that enables the integration of KBS_{given} in such a scenario without modifying its knowledge base internally.

For that purpose we need a specification existing systems must obey to allow an integration on various levels of communication. We assume a dynamic *communication model* between heterogeneous knowledge–based systems that consists of three layers as sketched in figure 1 (lower part). The layers can be regarded as abstractions of what happens on the physical layer when two heterogeneous KBSs exchange knowledge, cf [4]:

- The *coordination layer* fixes the addresses of the sender and the recipient, the identification of the messages, etc.
- The *communication layer* defines the logic of the communication, i.e. the message type, the quality of the message, etc. For instance it specifies whether a message has to be interpreted as new valid knowledge or whether the message is a query to be derived from the knowledge base.
- The *content layer* specifies the content of the message, both syntactically and semantically. Ontologies define the semantics of the identifiers that are used to compose a message.

Each of these layers has to be specified statically for each or the participating KBSs to achieve dynamic interoperability, cf [2]. In the context of this work we are primarily interested in the semantics aspect of the content layer. We assume the existence of an interlingua in which the contents of a message is directly represented or into which the interface language of the KBSs can be translated, e.g. KIF.

For the following discussion we adopt a rather logical view on ontologies. We regard an ontology as a finitely axiomizable first order theory T in first order predicate logic (FOPL) that tries to capture the intended interpretation $\hat{\mathcal{I}}$ of the identifiers of the conceptualization and their well–formed use. The *intended interpretation* is characterized by two properties. First, it must be a model of the ontology $\hat{\mathcal{I}} \models T$, i.e. all sentences of the theory must denote "true" applying the standard interpretation function. Secondly, the intended interpretation is the one that describes the behavior of a KBSs with respect to the functional view. We say that the KBS commits to the ontology if its behavior is consistent with the intended interpretation [7]. The latter property is the deeper purpose for defining ontologies: A means to define parts of the communication capabilities of a KBs and to interpret the outcome of a KBS semantically correct.

We assume the existence of two conceptualizations that define the identifiers at the interface of KBS_{new} and KBS_{given}. Formally these are two sets of identifiers together with their intended interpretations. The relationship between the conceptualization is that they describe the same domain but structure it differently. More formally, if we have two sentences φ of KBS_{new} and φ^{τ} of KBS_{given}[1] and two corresponding models $\hat{\mathcal{I}}'$ and $\hat{\mathcal{I}}$ we have to achieve

$$\models_{\hat{\mathcal{I}}'} \varphi \text{ iff } \models_{\hat{\mathcal{I}}} \varphi^{\tau} \qquad (1)$$

Moreover, for the purpose of exchanging knowledge represented on the basis of one ontology into an equivalent representation according to another ontology it is required that not only we can show that two given

[1] The notation τ is anticipated for *translation* and will be formally introduced in the next section.

sentences (φ and φ^τ) "say the same thing" but furthermore we would like to generate φ^τ constructively from φ. It is obvious that we are also not free in choosing our models $\mathcal{I}$ and $\mathcal{I}'$ independently of each other. Informally, we cannot expect to represent common knowledge if we have different understandings of what is true in the world we want to model.

3 SOLUTION APPROACH: THEORY INTERPRETATION

Constrained by the problem description we developed a solution approach that consists of three steps:

1. We provide a knowledge representation language and a system that allows for the definition of ontologies for each of the participating KBSs individually.
2. We define a methodology for describing translation rules between terms of the two ontologies at hand.
3. We apply the translation rules at run time to achieve a semantics preserving transformation of messages that are sent from a KBS to another one.

To achieve this goal we build on some technologies and techniques that are well known. The representation language for the ontologies is based on the Ontolingua approach [7]. Additionally, we added a few more structuring primitives and a semantics for inheritance. For the purpose of describing the translation rules we use the technique of theory interpretation developed in mathematical logics [3]. This approach also states the necessary conditions on the relationship between the two ontologies and their models such that knowledge can be translated. The intrinsic translation steps for messages sent from one KBS to the other is then a straightforward application of a sequence of structure preserving translations that are applied to the message sent to the other KBS. The overall solution approach is shown in figure 1.

We will now sketch the development of the terminology that we need to formulate a theorem that captures the intuitive notion of "says the same thing" in (1) along the line of [1, 3]. Steps 2 and 3 above are included in this procedure. In the following we will use a simple example that should help to clarify the formal definitions.

Example 1 *As an example consider on the one hand an ontology that defines the conceptualization for a knowledge base of metals, i.e.*

- *the sets of individuals*

  metals: $\{Li, \ldots, Hg, \ldots, Ru\}$
  temp-qual: $\{Lowroom, Roomtemp, Highroom\}$

- *the binary predicate '$\succ$' defining the obvious total order on temp-qual*
- *the binary function melt-point: metals $\rightarrow$ temp-qual*

Example 2 *On the other hand consider the theory of all elements of the periodic system represented as*

- *the set of elements with their numbers and the melting temperature as quantities*

  elements: $\{1, \ldots, 80, \ldots, 104\}$
  temperature-^oC: $\{-273^oC, \ldots\}$

- *the binary function melting-point: elements $\rightarrow$ temperature-^oC*
- *the binary predicate '$>$' defining an order on temperature-^oC and the binary predicate '$\gg$' defining an order on the set of elements by their element numbers*

We informally introduced the notion of a sorted logic, but will not use it explicitly in the following. However, the reader familiar with FOPL will not have difficulties to translate it into pure first order logic.

Let L_{new} be a language of first order predicate logic with fixed sets of n-place predicate and function symbols, possibly augmented by the equality symbol '$\doteq$'. T_{given} is a theory in a (possible different) language L_{given}, which includes equality, i.e. we assume that the ontologies are already defined.

Definition 1 *A* **translation** τ *of L_{new} into T_{given} is a function $\rightsquigarrow$ on the quantifier $\forall$ and the sets of predicates and functions of L_{new} such that*
— τ assigns to $\forall$ a formula $\tau_\forall$ in which at most v_1 occurs free, such that

$$T_{given} \models \exists v_1 \tau_\forall \tag{2}$$

— τ assigns to each n-place predicate symbol p a formula τ_p of L_{given} in which at most the variables $v_1, \ldots, v_n$ occur free.
— τ assigns to each n-place function symbol f a formula τ_f of L_{given} in which at most $v_1, \ldots, v_n, v_{n+1}$ occur free, such that

$$\begin{aligned} T_{given} \models\ & \forall v_1 \cdots \forall v_n (\tau_\forall(v_1) \rightarrow \ldots \rightarrow \tau_\forall(v_n) \\ & \rightarrow \exists x (\tau_\forall(x) \wedge \forall v_{n+1} (\tau_f(v_1, \ldots, v_{n+1}) \\ & \quad \leftrightarrow v_{n+1} \doteq x))) \end{aligned} \tag{3}$$

(2) states that in any model of T_{given} the formula $\tau_\forall$ defines a nonempty set to be used as the universe of an interpretation of L_{new}. Formula (3) states that for all $\vec{v}$ in the set defined by $\tau_\forall$, there is a unique x such that $\tau_f(\vec{v}, x)$, and furthermore x is in the set defined by $\tau_\forall$, thus in any model of T_{given}, τ_f defines a function on the universe defined by $\tau_\forall$. In case of a constant c (n = 0) (3) becomes

$$T_{\text{given}} \models \exists x(\tau_\forall(x) \wedge \forall v_1(\tau_c(v_1) \leftrightarrow v_1 \doteq x))$$

i.e. τ_c defines an object that is also in the set defined by $\tau_\forall$. In case that L_{new} and L_{given} are identical we get the trivial translation independent from T_{given}.

Example 3 *We regard the language of example 1 as our language L_{new} and example 2 as the theory T_{given}. Hence we get the following definition for a translation τ.*

$$\begin{aligned}
\tau_{\forall:\text{metals}}(x) &\rightsquigarrow \gg(104, x) \wedge \gg(x, 87) \wedge \ldots \\
&\textit{The numbers of the metallic elements} \\
\tau_{\forall:\text{temperature-qual}} &\rightsquigarrow x \doteq \textit{-273° C} \vee \\
&\quad\ x \doteq \textit{-272° C} \vee \ldots \\
\tau_{\text{Hg}}(x) &\rightsquigarrow x \doteq 80 \\
\tau_{\text{Room--temp}}(x) &\rightsquigarrow x \doteq \textit{20° C} \\
\tau_{\text{Low-room}}(x) &\rightsquigarrow \textit{20° C} > x \\
\tau_{\text{High-room}}(x) &\rightsquigarrow x > \textit{20° C} \\
\tau_{\text{melt-point}}(x, y) &\rightsquigarrow \textit{melting-point}(x, y) \\
\tau_{\succ}(x, y) &\rightsquigarrow \succ(x, y)
\end{aligned}$$

The reader should check that the above definitions is indeed a translation by checking conditions (2) and (3) in definition 1. Especially, we have to consider the translation rule for melt-point *which must yield condition (3)*

$$\begin{aligned}
T_{given} \models\ & \forall x\text{: elements}(\tau_{\forall:\text{metals}}(x) \\
& \rightarrow \exists y(\tau_{\forall:\text{temperature-qual}}(y) \wedge \\
& \qquad \forall z(\tau_{\text{melt-point}}(x, z) \leftrightarrow y \doteq z)))
\end{aligned}$$

Once we have expressed the identifiers of L_{new} by formulas of T_{given} we have to check whether the interpretation we imposed on the identifiers is consistent with what we intended. The way we impose an interpretation ${}^\tau\hat{\mathcal{I}}$ for L_{new} from the model $\hat{\mathcal{I}}$ of T_{given} can be straightforwardly defined:

$$\begin{aligned}
|{}^\tau\hat{\mathcal{I}}| &:= \text{the set defined in } \hat{\mathcal{I}} \text{ by } \tau_\forall \\
p^{{}^\tau\hat{\mathcal{I}}} &:= \text{the relation defined in } \hat{\mathcal{I}} \text{ by } \tau_p, \\
&\quad\ \text{restricted to } |{}^\tau\hat{\mathcal{I}}|, \\
f^{{}^\tau\hat{\mathcal{I}}}(a_1, \ldots, a_n) &:= \text{the unique } b \text{ such that} \\
&\quad\ \models_{\hat{\mathcal{I}}} \tau_f\lfloor a_1, \ldots, a_n, b\rfloor, \\
&\quad\ \text{where } a_1, \ldots, a_n \text{ are in } |{}^\tau\hat{\mathcal{I}}|.
\end{aligned}$$

It is obvious that ${}^\tau\hat{\mathcal{I}}$ is indeed an interpretation of L_{new}.

Definition 2 *A* **translation interpretation** ${}^\tau\hat{\mathcal{I}}$ *for a language L_{new}, τ as in definition 1 is an interpretation as defined above.*

Definition 1 states the conditions (2 and 3) for a translation τ on a semantic basis. Thus we have finished step 2 of our principle solution approach. Our ultimate goal to translate sentences constructively (step 3 above) is now furthermore approached by applying τ on sentences recursively in a systematic manner.

We start by considering translations of atomic formulas α of L_{new}, e.g. α is pfx (p a predicate symbol, f a function symbol, x a variable). Since we cannot apply the translations rules to nested formulas we have to transform them into an equivalent unnested one, e.g. α is logically equivalent to

$$\forall y(fx \doteq y \rightarrow py)$$

Applying the translation τ to the predicates and functions finally leads to the L_{given}–formula α^τ

$$\forall y(\tau_f(x, y) \rightarrow \tau_p(y))$$

We will not give the formal definition of α^τ for the ease of shortness. Structure preserving translations of nonatomic formulas are defined as follows; φ and ψ are formulas:

$$\begin{aligned}
(\neg\varphi)^\tau &:= (\neg\varphi^\tau) \\
(\varphi \rightarrow \psi)^\tau &:= (\varphi^\tau \rightarrow \psi^\tau) \\
(\forall x\varphi)^\tau &:= \forall x(\tau_\forall(x) \rightarrow \varphi^\tau)
\end{aligned}$$

Example 4 *By using the definitions in example 3 we can constructively apply the translation rules to a sentence that expresses that all metals except mercury have a higher melting-point than room temperature.*

$$\forall x\text{: metals}[\neg(\doteq(x, \text{Hg})) \rightarrow \succ(\textit{melt-point}(x), \textit{Room-temp})]$$

yielding the sequence of translated sentences:

$$\begin{aligned}
&\Big[\forall x\text{: metals}[\ \neg(\doteq(x, \text{Hg})) \\
&\qquad \rightarrow \succ(\textit{melt-point}(x), \textit{Room-temp})]\Big]^\tau \\
&\forall x\text{: elements}\Big(\tau_{\forall:\textit{metals}}(x) \\
&\qquad \rightarrow [\neg(\doteq(x, \text{Hg}))]^\tau \\
&\qquad \rightarrow [\succ(\textit{melt-point}(x), \textit{Room-temp})]^\tau\Big) \\
&\forall x\text{: elements}\Big(\tau_{\forall:\textit{metals}}(x) \\
&\qquad \rightarrow [\neg(\doteq(x, \text{Hg}))]^\tau \rightarrow \forall z(\tau_{\text{melt-point}}(x, z) \\
&\qquad \rightarrow \forall y(\tau_{\text{Roomtemp}}(y) \rightarrow \tau_{\succ}(z, y)))\Big)
\end{aligned}$$

and finally the sentence in T_{given}

$$\forall x\text{: elements}\Big(\gg(104,x)\wedge\gg(x,87)\wedge\ldots \rightarrow\neg(\doteq(x,80)) \rightarrow\forall z(\textit{melting-point}(x,z) \rightarrow\forall y(y\doteq 20^{o}C\rightarrow>(z,y)))\Big)$$

For the ease of readability this sentence has already been simplified since we do not have the ability to translate constants into other constants within the '$\doteq$' predicate.

We now have the terminology to prove the following theorem that formalizes the intuitive notion of "means the same thing as" as formalized in 1:

Theorem Let τ be an translation of L_{new} into T_{given}, let $\hat{\mathcal{I}}$ be a model of T_{given}, and ${}^{\tau}\hat{\mathcal{I}}$ a translation interpretation for L_{new}. For any sentence φ of L_{new},

$$\models_{{}^{\tau}\hat{\mathcal{I}}}\varphi \text{ iff } \models_{\hat{\mathcal{I}}}\varphi^{\tau}$$

Proof By induction on φ.

From a knowledge sharing perspective we can now state: If we construct the mappings for identifiers of the ontology of KBS_{new} into the ontology $\text{KBS}_{\text{given}}$ thus imposing an interpretation ${}^{\tau}\hat{\mathcal{I}}$ on KBS_{new} which must be compatible with the intended interpretation then we can translate any message from KBS_{new} to $\text{KBS}_{\text{given}}$ represented in our interlingua such that the semantics remains the same.

4 CONCLUSION

We presented an approach for knowledge sharing and reuse based on a specific model of knowledge exchange. This approach gives deeper insights about the issues of what knowledge exchange between heterogeneous KBSs is about and what the prerequisites to achieve this goal are. However, from an AI point of view the pure mathematical approach is too crude for two reasons: First, it is only defined on "flat" formulas thus neglecting structuring information, e.g. that a given formula is a definition of an object class. Secondly, it does not take into account that the translation rules may contain meta–knowledge, e.g. that the translation is a typical abstraction [9] and therefore not providing schemas for well–known translation situations. These issues are the content of further research.

The approach has been realized in a prototypical client–server architecture implemented in VisualWorks 2.0. The server supports the definition of ontologies and translation rules. The underlying concepts for the ontology representation language are adopted from Ontolingua [7] which is chosen as the syntax for a human readable form of the ontologies. Once the ontologies and the translation rules are defined clients can send a (typed) KIF expression to the server that returns a translated expression which is equivalent with respect to the target ontology.

References

[1] B. Bachmann. Mappings between ontologies in multidisciplinary design. In M. Alberts, editor, *Workshop Notes (AI & Design '94): A Semantic Basis for Knowledge and Data in Design*, 1994.

[2] B. Bachmann and F. Dridi. Definition of a communication layer for expert systems. In J.G. Chen, editor, *Proc. 6th Int. Conf. on AI and Expert System Applications, Houston, Texas*, pages 117–123, 1994.

[3] H.B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, San Diego – New York – Berkeley, 1972.

[4] T. Finin, R. Fritzson, and D. McKay. A language and protocol to support intelligent agent interoperability. In *Proc. CE & CALS Washington '92 Conference*, June 1992.

[5] M. Genesereth and R. Fikes. Knowledge interchange format version 3.0 reference manual. Logic Group Report Logic–92–1, Computer Science Department, Stanford University, June 1992.

[6] M.R. Genesereth and N.J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, 1987.

[7] T. Gruber. A translation approach to portable ontology specification. *Knowledge Acquisition*, 5(2):199–220, 1993.

[8] H.L. Levesque. Foundations of a funcational approach to knowledge representation. *Artificial Intelligence*, 23:155–212, 1984.

[9] P.P. Nayak and A.Y. Levy. A semantic theory of abstractions. In J. Van Baalen, editor, *Workshop on Change of Representation and Reformulation*, 1994.

[10] R. Neches, F. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W.R. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, pages 36–56, Fall 1991.

MEANING REPRESENTATION FOR KNOWLEDGE SHARING IN PRACTICAL MACHINE TRANSLATION

Kavi Mahesh and Sergei Nirenburg
Computing Research Laboratory, New Mexico State University
mahesh@crl.nmsu.edu, sergei@crl.nmsu.edu

Abstract

Knowledge-based machine translation can be viewed as the problem of extracting and representing the meaning of a text and generating a translation in a target language using the meaning representation. Meaning extraction requires the integration of information present explicitly in a text with common sense and domain knowledge given to the system. Thus, integrating linguistic knowledge of each language with general world knowledge is a central problem in machine translation, especially when more than two languages are involved. In this article we consider the design of a meaning representation that enables language-specific lexicons to share knowledge with a language-independent world model. We illustrate how the underlying core meaning representation can be enhanced in three different ways to arrive at lexical, ontological, and text meaning representations. The meaning representations presented here have been implemented in the Mikrokosmos machine translation system and used to represent Spanish and Japanese lexicons in addition to a broad-coverage ontological world model.

1 Sharing Lexical and World Knowledge

Most human knowledge is represented and communicated via natural languages in spite of relatively recent efforts such as Cyc (Lenat and Guha, 1990) to "computerize" common sense knowledge. Extracting the meaning of a natural language text involves integrating the information present overtly in the text (which is typically incomplete) with the linguistic and world knowledge sources given to the system. The design of such a knowledge based natural language processing (NLP) system is inseparable from issues of sharing knowledge between linguistic and world knowledge sources, and, in turn, sharing knowledge representations between lexical, textual, and ontological meaning representations.

Machine translation (MT), especially when it involves more than two languages, is an NLP problem that provides a particularly interesting context to study knowledge sharing between multiple knowledge bases containing different types of knowledge. In multilingual knowledge based machine translation (KBMT), the meaning of a source text is extracted using both linguistic knowledge of the source language and general world knowledge. The resulting meaning representation is then used to generate translations in one or more target languages. Language generation also requires both world knowledge and knowledge of the target language. The internal representation of the meaning of a text must be independent of particular languages to enable translation to or from any of a set of languages.

A fundamental issue in the design of multilingual MT (and other NLP) systems is how to represent (i) linguistic knowledge for each language, (ii) general world knowledge, and (iii) text meanings. It is obvious that there must be one lexicon for each language since languages differ at least in the words they use. For simplicity we assume that all linguistic knowledge for a language is represented in its lexicon without considering the possibility of sharing linguistic knowledge across languages. The question that remains is how to share general world language with individual lexicons for different languages.

In a unilingual NLP system, linguistic and world knowledge can be combined in a single, monolithic representation. World knowledge can be distributed throughout the single knowledge base and inextricably intertwined with linguistic knowledge. Such a scheme has in fact been employed in a variety of NLP systems (e.g., Birnbaum and Selfridge, 1981; Jurafsky, 1992). If the monolithic design is applied to a multilingual system by representing one knowledge base for each language, there must be both a common meaning representation and sharable world knowledge that are left unspecified but are duplicated for each language.

In our approach, linguistic knowledge is represented in separate lexicons for each language while world knowledge is in a language-independent ontology that is shared by each of the lexicons. Sharing of the ontological world model is enabled by having a core language for meaning representation (MR) that is shared by the lexicons and the ontology and also used to represent text meanings internally. Figure 1 shows the relationships between the various knowledge sources in a multilingual MT situation.

This scheme has been developed over the years in a series of MT projects (Nirenburg, et al, 1992; Nirenburg and Levin, 1992; Onyshkevych and Nirenburg, 1994) and is the basis for our current KBMT project called Mikrokosmos. In Mikrokosmos we have developed a Spanish lexicon of over

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/182

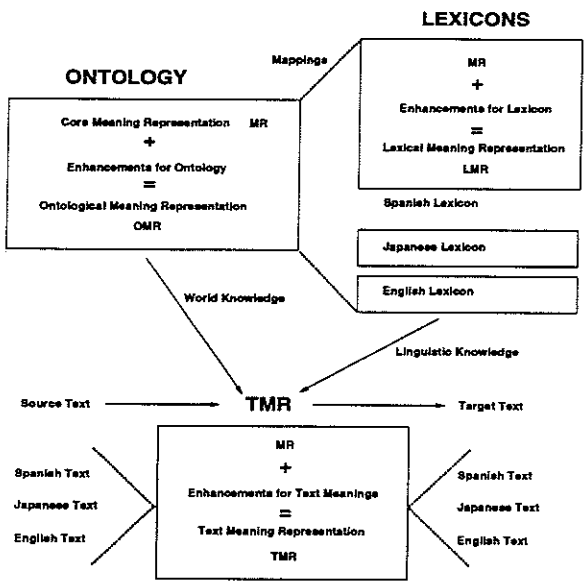


Figure 1: Several Lexicons Sharing a Single Ontology in a Multilingual MT System.

6000 words and a broad-coverage ontology of over 4500 concepts. Acquisition of a Japanese lexicon has also begun recently. Mikrokosmos can currently produce text meaning representations from article-length real-world Spanish texts about company mergers and acquisitions. Although multilingual MT is not new, Mikrokosmos can be seen to be a truly large-scale project in semantic analysis when the richness and coverage of its meaning representations (with respect to a single word or text) are considered together with the nontrivial sizes of its knowledge bases.

The methodology for designing knowledge representations followed in our work has implications for the study of knowledge sharing in general. We first consider the content of knowledge needed for the entire system and identify a core meaning representation that is to be shared by each of the knowledge bases in the system. This core representation is then enhanced minimally in turn for each type of knowledge (or each individual representation) based on a functional analysis of the requirements of the overall task. In this paper, we present the design of the core meaning representation (MR) and illustrate how it is enhanced to arrive at lexical, ontological, and text MRs (Figure 1) to enable knowledge sharing. This work can also be viewed as the beginnings of knowledge sharing between natural language texts in different languages.

2 Meaning Representation

Designing a meaning representation for NLP involves determining its content and its representation. We address the issue of content in terms of the nature of primitives used to share knowledge between linguistic and world knowledge representations. The structure of the representation is shown to be based on the needs for composing the primitives in different ways, the expressiveness of which is determined by the machine translation task and its needs for linguistic and world knowledge.

Semantic Primitives: A fundamental question in designing knowledge representations for NLP is which and how many primitives to use. Our approach takes an intermediate position between a highly minimalist and a highly excessive number of primitives. The question of primitives is often not addressed in knowledge representation research in other areas of AI and sometimes left unspecified even in NLP systems. Semantic lexicons are sometimes built by introducing a number of primitives as needed for representing word meanings. Neither the set of primitives, nor the taxonomic or other relationships between the primitives is specified in such systems.

Eleven +/- Two Primitives: The best example of an extremely minimalist position can be seen in Schank's (1973) well-known Conceptual Dependency theory (CDs), an ontology of about 11 events.[1] Such a small number of primitives is not practical for building large scale systems that attempt to capture the full richness of meaning that is necessary for MT or other NLP tasks in a domain. When we attempt to decompose complex events such as "a takeover bid for a company" in CDs, the resulting meaning representations will be lengthy, convoluted, and hard to acquire on a large scale. Moreover, they are unsuitable for MT since it is very hard to generate the equivalent word(s) in a target language from such highly decomposed meaning representations.

Each Word Is a Primitive: The other extreme position, a popular one in NLP, makes almost every word sense in a natural language a primitive by itself. Examples of such "ontologies" are WordNet (Miller, 1990) and Sensus (Knight and Luk, 1994). In this approach, the large set of primitives is necessarily tied to a particular language (English in the above systems) which may be desirable for MT if the target language was always the same (say English). In a more general multilingual situation, however, the set of primitives must be independent of any natural language. A more compositional meaning representation with a smaller number of language independent primitives is much more practical for constructing large-scale semantic lexicons for multiple languages.

A Practical Intermediate Position: We take an intermediate approach and propose a set of primitives that is much bigger than CDs or LCSs but significantly smaller than the typical size (of the order of 50,000) of a "word sense taxonomy" such as WordNet (Miller, 1990). Our experience in Mikrokosmos and its predecessor projects shows that fewer than 10,000 primitives are sufficient for building practical MT systems in a nontrivial domain, such as company mergers and acquisitions.[2] For successful, multilingual MT, such a system must be provided with a rich compositional structure in its meaning representations. The 6000-8000 primitives must also be organized in a highly interconnected ontological network. Using such a scheme, we have built a Spanish lexicon with

[1] Other well-known minimalist approaches include Jackendoff's (1990) lexical-conceptual structures (LCS). See also (Dorr 1993) See Wilks (1992) or Onyshkevych and Nirenburg (1994) for criticisms of the minimalist approach to meaning representation.

[2] It must be noted that we do not propose to encode only those meanings of words that are in the chosen domain. We in fact encode all meanings of words in a corpus using much less than 10,000 primitives.

over 6000 words that use less than 2500 primitives in their meaning representations.[3]

Composing Primitives and Expressiveness: How do we compose the chosen primitives to build lexical representations for word meanings, representations of extra-linguistic world knowledge, and meanings of entire texts? This question is far too often predetermined by one's methodological commitments. For example, first-order predicate calculus or a frame-based knowledge representation system is commonly chosen without analyzing the functional requirements and practical constraints from the NLP task on the expressiveness of meaning representations. We show how limited expressiveness is an enviable virtue from the practical point of view of building large scale semantic NLP systems. Starting with a minimally compositional representation that is highly limited in its expressiveness, we show what enhancements are necessary in order to match the needs of linguistic, textual, and world knowledge representations.

2.1 Core Meaning Representation

The core of our meaning representation (MR) is the set of primitives that are defined as concepts in a language-independent ontology. Since we have fewer primitives than words (or word senses) in a natural language, we must be able to compose two or more primitives to build MRs. In the core MR, primitives can be composed with one another as outlined below:

1. The primitives are first partitioned into free-standing *concepts* (namely, *events* and *objects*), *properties*, and a small set of *literals*. Literals (such as values of color properties: red, blue, etc.) are atomic symbols that cannot be modified further.[4] Properties are not first-class concepts either in that they cannot exist by themselves in the MR. They can only be used to modify other concepts.
2. Properties are further partitioned into *attributes* and *relations*. A property modifying a concept together with the value of that property (i.e., a named value) is called a *slot*. There can be only one value per slot.
3. A multitude of slots can be composed with a concept. Attributes are the names of slots where the values are numbers or literals.
4. Relations are used to represent modifications where the values are other concepts. All conceptual relations are binary and their values must be names of other concepts. A relation always links the concept it is modifying to the single concept that is the value.

The core MR resulting from the above rules for composing primitives is shown in the form of a BNF grammar in Figure 2. It is very limited in expressiveness but constitutes the common foundation for constructing linguistic and world knowledge representations that share a significant amount of language-independent knowledge. In order to show how limited this core MR is, we list some commonly used representational apparatus that is missing in the MR:

1. There is no negation operator in the MR.
2. There is no taxonomic organization of primitives or any form of inheritance.
3. It is not possible to compose properties. Properties cannot be attached to other properties.
4. There are no quantification operators. For example, it is not possible to quantify over the set of properties attached to a primitive or represent the existence of an unknown value in a slot.
5. There is no reification operation. It is not possible to raise a slot to the level of a free-standing concept in the MR.
6. It is not possible to make arbitrary (first or higher order) assertions over the representation.
7. The only form of equality test is through simple identity of primitives.

```
<concept> := <concept-name> <slot>*

<slot> := <attribute-slot> | <relation-slot>

<attribute-slot> :=
    <attribute-name> ( <number> | <literal> )

<relation-slot> :=
    <relation-name> <concept-name>

<concept-name> := <primitive>          <literal> := <primitive>

<attribute-name> := <primitive>        <relation-name> := <primitive>

Note: The same primitive cannot be more than one of a literal,
a concept-name, an attribute-name, or a relation-name.
```

Figure 2: A BNF for the Core Meaning Representation for Knowledge Sharing.

We now take this drastically limited MR and enhance it in necessary ways to formulate suitable representations for lexical, world, and text meanings. This approach guarantees that there will be virtually no need for duplicating any piece of knowledge between the different knowledge bases. Any part of the system's knowledge that can be expressed in the core MR need be represented only once and can be shared by all the lexical and world knowledge bases. Only those parts that require the following enhancements to the core MR may not be sharable between all the different representations.

2.2 Ontological Meaning Representation

In a multilingual MT system, it is best to keep the common world knowledge in a single knowledge base that is shared by the lexicon for each language. We call this common knowledge base the *ontology* and use it to define taxonomic and other relationships between the primitive concepts in the MR. Conceptual relations represent the constraints on potential fillers of slots, rather than actual values (and hence are known in NLP as *selectional restrictions*). In order to represent such conceptual relationships, we need to enhance the MR as follows:

[3] Construction of a Japanese lexicon using the same set of primitives has also begun recently.

[4] Literals are used to bottom out on unending decompositions of meanings in order to keep MRs simple. Indiscriminate use of literals takes us closer to the word sense approach and is strongly discouraged by our methodological guidelines for representing lexical and world knowledge (Mahesh, 1996; Mahesh and Nirenburg, 1995).

1. We introduce a new type of filler for slots that denotes a constraint on potential values and refer to it as a *sem facet*. A slot now has a property, a sem facet, and a *value facet*. The value facet carries the actual value of a property as formulated originally in MR.
2. To represent constraints on attributes that take numerical values, we allow the sem facets of attribute slots to be a closed or open numerical range (e.g., (0 <= x <= 1) and (>= 0) respectively).
3. Often, constraints on slots cannot be specified using a single primitive in the sem facet. As a first step, we allow *multiple fillers* in sem facets and define their semantics to be a disjunction of those fillers. For example, the gender property of a human being is constrained to be male or female. Often, such enumeration is prohibitively uneconomical. For example, we could not possibly list all the different types of animals in the world in order to constrain the fillers of a slot to just animals. Hence, we arrange the primitive concepts in a *taxonomic hierarchy*. The new semantics for a sem facet says that the actual value can be any descendant of the constraint in the hierarchy. Since it is very difficult to arrange all concepts in the world in a simple tree, we allow *multiple parents* and make the hierarchy a plex structure.
4. For economy in representing properties of concepts, we introduce *inheritance* so that slots that are defined for parent nodes in the hierarchy need not be repeated in children concepts unless we want to change the fillers. We often need to block inheritance of a particular property to a particular child. Such nonmonotonic inheritance is indicated by a special filler, `*nothing*` in the particular slot for the child. For example, all animals can be purchased for a price, but not humans (in today's post-slavery world).
5. Sometimes, a semantic constraint excludes just one or two concepts in an entire subtree in the taxonomy. For economy in representing such constraints, we introduce a limited form of *negation* in sem facets. For example, to constrain the agents of flying to birds, we can say "bird, not penguin, not ostrich" instead of having to list all the other birds.
6. In NLP (and other tasks such as reasoning), it is often useful to know the "typical" value of a slot in addition to the set of all possible values (i.e., the sem constraint). In order to represent such typicality information, we introduce another facet called the *default facet*.
7. Ambiguity resolution and other semantic processes such as the interpretation of non-literal expressions (metaphor, metonymy, etc.) require making selections based on the "semantic closeness" of concepts. In order to support such search operations in the ontology, for each slot that is a relation in any concept, we need to be able to find the inverse relation from the filler to the modified concept. Hence we introduce an *inverse* relation between pairs of relations and arrange the relations also in a hierarchy.

The resulting representation, called Ontological Meaning Representation (OMR) has been used to build an ontology of 4500 concepts covering a wide variety of meanings in the world while paying particular attention to our domain of company mergers and acquisitions.[5] Knowledge represented in the ontology is shared (not duplicated) by lexicons for different languages. It is also shared by the meaning representation for a text.

2.3 Lexical Meaning Representation

Meanings of words and other linguistic knowledge necessary for semantic analysis are represented in separate lexicons for each language. Lexical representations must map meanings of words to the syntactic structures in which the word can occur in that language. In addition, lexical representations must be able to modify selectional constraints represented in the ontology to capture the many idiosyncrasies of what is acceptable and what is not in a particular natural language. Lexical meaning representation (LMR) is obtained through the following enhancements to MR:
1. We declare that all properties (i.e., slots) defined for a concept in the ontology are implicitly present in the lexicon whenever a word meaning is represented using the concept name. Thus, ontological knowledge is automatically shared by lexicons. Properties of concepts are repeated in the lexicon only when the values or constraints specified in the ontology must be changed to represent a word meaning.
2. Word meanings are mapped to syntactic structures by binding variables between the syntactic and semantic fields of lexical entries. Such variable binding enables the NLP system to use syntactic guidance in extracting the meaning of a text.
3. In order to further constrain a conceptual relation or an attribute slot specified in the ontology, we borrow the *sem facet* from OMR. To allow the lexicon to relax an ontological constraint, we introduce a fourth facet called *relaxable-to*. These facets allow the lexicon to represent language-specific idiosyncrasies.
4. Since we have fewer primitives than words, meanings of many words are represented by composing more than one primitive concept. Word meanings often map to particular relationships between concepts. Since these concepts can be modified further within the lexical entry, there is a need for referring to the modified concepts in the entry. This is accomplished by introducing variable bindings between the different concepts used in the entry for a word.
5. Word meanings often include various linguistic embellishments such as attitudes, modalities, time, aspect, and so on. These element of meaning cannot be adequately represented in the ontology since they can be attached to almost any concept and since their usage tends to be highly language specific. We view them as additional properties that are not part of the property hierarchy in the ontology.
6. Words often map to sets of objects or events or set-theoretic relations between them. We introduce a set and subset notation to capture such meanings. For example, the word "majority" means a subset of a universal set that is more than half the size of the universal set. Without the expressiveness of the set notation, we would need many more primitives in the system.
7. We also introduce a limited form of negation using a property called *polarity* (with possible values *positive* and *negative*). This is used, for instance, to say that an event did not occur. A limited form of existential quantification is added

[5]The other type of world knowledge useful for NLP, namely, episodic knowledge of remembered instances of events and objects (such as people, places, organizations, etc.) is represented in a separate knowledge base called the *onomasticon* using the same OMR representation.

to LMR to represent an existing, but unknown filler of a property using the special symbol *unknown*.

Using this representation, we have built a Spanish lexicon with over 6000 entries where meanings of words are represented in LMR by sharing ontological knowledge.

2.4 Text Meaning Representation

A text meaning representation (TMR) is made up of particular instances of meanings represented in the ontology and the lexicon. To distinguish between concepts and instances, we introduce an *instantiation operator* that works as follows. An instance has all the properties that its concept has either in the ontology or in the lexicon for which a value is available. Instances only have value facets in their properties. Values are found as follows: any value represented in the lexicon supersedes any value in the ontology. If no value is specified in either the lexicon or the ontology, a default filler from the ontology is the value for the instance.

In addition, the following enhancements are made to MR to get the TMR language:

1. Properties in TMRs often need to refer to chunks bigger than individual instances in the TMR. For example, in "John said that Bill's obsession with guns sent him to prison," what was said was more than a single object or event. In order to support such scoping needs, we introduce a "super structure" called *proposition*. A proposition has a head that must be a concept. In addition, it has a limited set of properties including time, aspect, and attitude. Propositions group word meanings into bigger structural units to represent meanings of entire sentences. A proposition is to TMR what a sentence is to a text.
2. An instance often needs to refer to a particular slot of another instance. We introduce a *reification* operator that raises a slot in an instance to the TMR level by making it an individual instance in the TMR.
3. In order to capture coreferences in a text, we introduce a (reified) *coreference* relation that identifies two or more instances in the TMR as referring to one and the same entity in the world.
4. Certain other linguistic embellishments are also added to the TMR. For example, *time relations* and *quantitative relations* are used to represent the corresponding information.

3 Discussion and Conclusions

We have presented the design of linguistic and world knowledge representations for a multilingual KBMT system based on the principles of parsimony, modularity, and compositionality, and the needs of practical methodology. In particular, we showed how having a core meaning representation that is shared by all the knowledge bases enables sharing of general world knowledge among a set of language (or possibly domain) specific lexicons.

Our experience shows that limited expressiveness of representations is a strong virtue from the practical point of view of acquiring large scale knowledge bases. Our restricted representations allow nontechnical acquirers and users to visualize entire knowledge bases statically by browsing through them. Highly expressive representations inevitably turn into massive black boxes that one can ask queries and get answers from, but cannot browse through to see certain parts independently of the rest of the knowledge base. We believe that our experiences in MT and the above methodological points have significant implications for other application areas as well.

References

Birnbaum, L. and Selfridge, M. (1981). Conceptual analysis of natural language. In Schank, R. and Riesbeck, C., editors, *Inside Computer Understanding*, pages 318-353. Lawrence Erlbaum Associates.

Dorr, B. (1993). *Machine Translation: A View from the Lexicon*. Cambridge, MA: MIT Press.

Jackendoff, R. (1990). Semantic Structures. Cambridge, MA: MIT Press.

Jurafsky, D. (1992). An on-line computational model of human sentence interpretation. In Proc. 10th National Conf. on AI, AAAI-92, pages 302-308.

Knight, K. and Luk, S. K. (1994). Building a Large-Scale Knowledge Base for Machine Translation. In *Proc. Twelfth National Conf. on Artificial Intelligence*, (AAAI-94).

Lenat, D. B. and Guha, R. V. (1990). *Building Large Knowledge-Based Systems*. Reading, MA: Addison-Wesley.

Mahesh, K. (1996). Ontology development for machine translation: Ideology and methodology. Technical Report MCCS-96-292, Computing Research Laboratory, New Mexico State University, Las Cruces, NM.

Mahesh, K. and Nirenburg, S. (1995). A situated ontology for practical NLP. In Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Canada, August 1995.

Miller, G. (1990). WordNet: An on-line lexical database. *International Journal of Lexicography* 3(4) (Special Issue).

Nirenburg, S. and Levin, L. (1992). Syntax-driven and ontology-driven lexical semantics. In Pustejovsky and Bergler, pp. 5-20.

Nirenburg, S., Carbonell, J., Tomita, M., and Goodman, K. (1992). Machine Translation: A Knowledge-Based Approach. Morgan Kaufmann Publishers, San Mateo, CA.

Onyshkevych, B. and Nirenburg, S. (1994). The lexicon in the scheme of KBMT things. Memoranda in Computer and Cognitive Science MCCS-94-277. Las Cruces, NM: New Mexico State University. To appear also in *Machine Translation*.

Schank, R. (1973). Identification of conceptualizations underlying natural language. In Computer Models of Thought and Language, Schank, R. and Colby, K., editors, San Francisco:W. H. Freeman Co.

Wilks, Y. (1992). Review of Ray Jackendoff's *Semantic Structures*, Computational Linguistics, vol. 18:1, pp 95-97.

Representing Appearance Information in a World of Interchangeable Documents

Ethan V. Munson*
Electrical Engineering and Computer Science
University of Wisconsin-Milwaukee

Abstract

Document processing researchers have long been interested in easing the exchange of documents between people using different software. The central results of this research have been interlingua for structured documents which represent logical structure and content while ignoring presentation. Adoption of these interlingua has been slow because of the lack of an acceptable standard for specifying presentation.

Proteus is a presentation specification system for structured documents. It provides a uniform specification language for all media and also supports synchronized multiple presentations, allowing the user to view the same document in several styles simultaneously even while the document is being edited. Technology like that provided by Proteus is critical if document interlingua are to be widely adopted.

Research on Proteus is also relevant to intelligent software systems. It provides a higher-level model of presentation that may be more appropriate for reasoning about the presentation of complex documents. Also, Proteus's uniform treatment of all media is based on a formal model of media that may find direct application in intelligent systems.

1 INTRODUCTION

Easing the exchange of information among people using a wide variety of applications and hardware has been a central concern of document processing researchers over the last two decades. After all, natural language documents remain a critical means by which information is shared. Beginning in the 1970's, the development of electronic mail, remote file transfer, and lightweight, medium-capacity storage in the form of floppy disks made it possible for computer users to easily transfer electronic documents over long distances. However, upon doing so, they quickly found that incompatibilities between the document representations of their software packages made electronic document interchange problematic, at best.

Two different approaches to solving this problem have been taken. Most commercial personal computer applications now solve the interchange problem by including a collection of software modules which handle import from and export to the representations of competitor's applications.

The second approach, and the one focused on by the research community, was to define interlingua for document systems. Authors exchanging a document represented in an interlingua would still use their own editing and formatting software to display, edit, or print the document, but would use the interlingua for export and import operations. The designers of these interlingua focused primarily on representing the content of the document and its logical structure, rather than its appearance. Appearance information was excluded because it was the primary source of application incompatibilities. Furthermore, experience with the Scribe [19] and LaTeX [14] batch formatters had shown that separating style information from the stored representation of the document allowed authors to quickly change styles to suit different circumstances without changing the document's underlying semantics.

Two such interlingua emerged in the form of international standards, the Standard Generalized Markup Language (SGML) [6] and the Open Document Architecture (ODA) [12]. Both SGML and ODA represent documents as trees with named nodes which must adhere to a context-free grammar defining the document's type. Connections between nodes which violate the tree structure (such as cross-references) are represented by secondary linking structures. The named tree nodes are used to represent the document's logical structure, while the document's content (text, graphic objects, etc.) is stored in the leaves of the tree. Because of their emphasis on logical structure, these systems are said to support *structured documents*. In practice, neither standard is widely accepted in its full form, though SGML has been more widely adopted than ODA. The most important use of SGML is for documents on the World Wide Web, which are represented using an SGML document type, the Hypertext Markup Language (HTML) [2].

A key roadblock to the widespread adoption of SGML is the lack of an acceptable system for describing the appearance of SGML documents. A draft standard, the Document Style Semantics and Specification Language (DSSSL) [13], has been under development for some time. While a complete implementation of DSSSL is not yet widely available, DSSSL specifications appear to work well for documents that are primarily textual. DSSSL has two serious limitations:

1. In order to support document presentations that rearrange the order of elements in a document, contain generated information like section numbers, or elide parts of the document, DSSSL has a module that can perform arbitrary transformations on the document tree. These unconstrained transformations make it difficult to map on-screen selections back to the pre-transformation document tree and thus hinder the construction of direct-manipulation editors for SGML documents. As a result, DSSSL is considered unsuitable for interactive applications.

2. DSSSL has no facility via which it can be extended to serve new media or to provide new formatting features for existing media. While DSSSL does have optional features that may be

*Author's Internet address: munson@cs.uwm.edu

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/187 ©1996 FLAIRS

left out of some implementations, it is simply not possible to add features that the designers have not already conceived of.

Several other presentation specification systems have appeared which address the difficulty of constructing an interactive editor based on DSSSL. DSSSLite [5] and Synex Information AB's Viewport [22] eliminate the tree transformation module. The Grif Toolkit [9] supports a restricted form of tree transformation, that can be used to automatically generate material such as labels and numbers. None of these systems can rearrange the order of document elements and none are extensible.

2 PROTEUS

Proteus [8, 17] is a new presentation specification tool initially designed and implemented as part of the Ensemble software development and multimedia document environment [7]. Like the systems described in the previous section, Proteus interprets presentation specifications and provides an interface through which an application can determine the values of formatting and layout parameters for the elements of a document. The design of Proteus's specification language addresses the central weaknesses of DSSSL:

1. Proteus is configurable and extensible. Its configuration is determined by a meta-specification of the presentation information required by the application. This meta-specification is used to adapt Proteus's presentation specification language to each application. One part of the meta-specification can describe extensions, called interface functions, which applications can use to give Proteus access to externally-maintained information that would otherwise be outside the scope of Proteus's specification language.

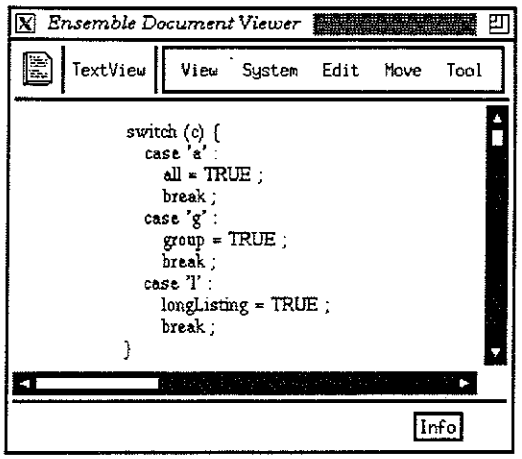
2. Proteus can provide all the effects possible with the tree transformation module of DSSSL through the combination of three features. First, Proteus provides a tree elaboration service, which allows specification-based generation of material not present in the document (such as labels and section numbers). Second, Proteus can perform *out-of-order* layout; that is, the order in which the elements of the document are laid out can be different from their order in a traversal of the document tree. Finally, elision can be performed through manipulation of visibility attributes.

3. Because Proteus only supports a restricted form of tree transformation (tree elaboration), the mapping between what the user sees on the screen and the underlying document is fairly simple. Thus, Proteus is suitable for use by interactive document editors.

There are two other important aspects of the research on Proteus: multiple, synchronized presentations and a new model of media.

- Proteus can support an arbitrary number of simultaneous presentations of the same document. Figures 1 and 2 show examples of multiple presentations for a program document and a graphics document, respectively. Multiple presentations are important for information-rich domains, such as software development environments, in which there are many "correct" ways to view the same information. Certain graphics applications (especially CAD applications) already provide multiple presentations, but the set of choices is hard-coded.

- The meta-specifications which configure Proteus for different applications are the starting point for a new model of media. This model is not yet complete, but it represents a significant advance toward a sound conceptual foundation for discussions of multimedia and of multimedia systems.



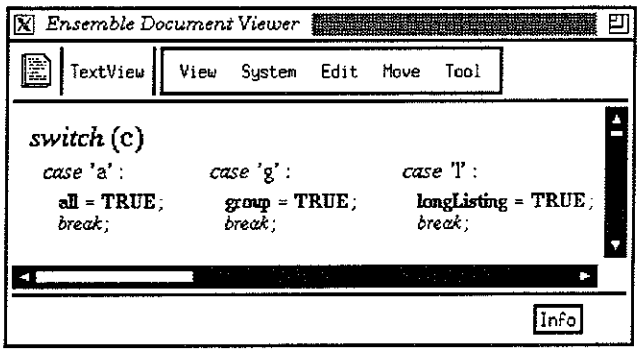


Figure 1: Two presentations of the same C switch statement using an early version of Ensemble. The first presentation uses a normal, line-oriented style. The second presentation uses a tabular style that cannot be reproduced in existing software environments.

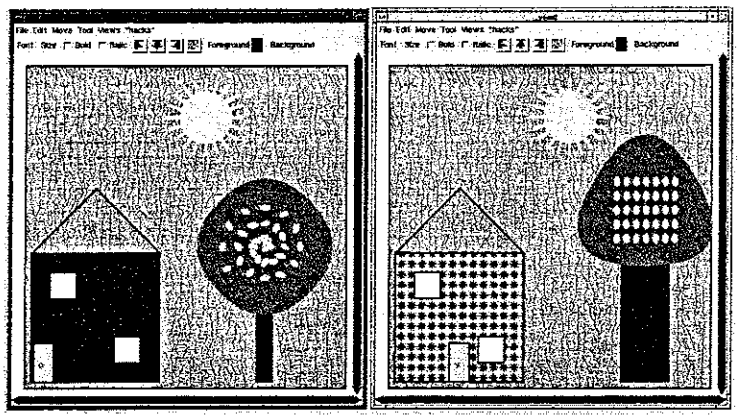


Figure 2: Two presentations of a simple graphical figure depicting a house, sun, and tree using a more recent version of Ensemble. The presentations differ in the color and texture of the house wall, the position of the door, and the trunk size, leaf area, and fruit placement of the tree.

2.1 Presentation Specifications

Presentation specifications describe how Proteus's presentation services should be applied to documents. Proteus has three configurable presentation services (attribute propagation, box layout, and tree elaboration) and an extension service (interface functions).

The *attribute propagation* service is used to define values of the parameters that control an application's formatting operations. Each parameter is treated as an attribute of the nodes of the document tree, values for which are propagated through the tree by a system of unidirectional constraints. Thus, a presentation rule like

```
StrokeWidth = Parent . StrokeWidth / 2 ;
```

constrains the node's `StrokeWidth` attribute to be half the value of the parent node's `StrokeWidth` attribute. The implementation of the attribute propagation service directly evaluates the expression tree for the right-hand side of the rule and uses a simple caching system to improve performance. To configure the attribute propagation service, an application must specify a name and a type for each formatting parameter.

The *box layout* service is used to define the layout of the elements of the document using constraints between nested boxes. There are four layout attributes for each dimension (minimum, maximum, center, and extent). Values for these attributes are managed using the mechanisms of the attribute propagation service with some additional support to handle the fact that there are only two degrees of freedom among the four attributes of each dimension. For example, the following rule would center a node with respect to its left sibling.

```
HorizPos: HMiddle = LeftSib . HMiddle ;
```

The box layout service is configured by specifying the number of dimensions in which the application lays out the document and the names by which the four attributes of each dimension will be specified.

The *tree elaboration* service is used to automatically generate material that is not found in the document itself, such as section numbers or standardized labels. Tree elaboration is specified in two stages. First, the new nodes that can be generated are declared. This declaration defines a subject field label for use in a memorandum document.

```
SubjectLabel : Text("Subject:") :
  BEGIN
  END;
```

Then, declared nodes are generated with creation commands, such as this one that creates the subject field label as a left sibling of the node for which the rule appears.

```
CreateBefore (SubjectLabel) ;
```

The tree elaboration service is implemented using a shadow tree system that reuses non-generated nodes wherever possible. An application configures the tree elaboration service by providing the names of the primitive data types for which nodes can be generated and, for each data type, the name and type signature of a function to be used to generate the nodes.

Proteus can be extended using the *interface function* service. This is a system of callback functions provided by the application. They are used when the application maintains information that can be useful for controlling presentation that Proteus would not otherwise have access to. In Ensemble, interface functions are primarily used to allow presentation specifications to be based on the results of program analysis. Interface functions can also be viewed as a system for giving Proteus access to externally-maintained attributes.

2.2 Limitations

In spite of its ability to address critical problems of other presentation specification systems, Proteus has a number of important limitations.

Language: The experience of the Ensemble user's community with Proteus's specification language exposed several inadequacies. Because the language has no abstraction or grouping facilities, specifications are excessively redundant. Tree elaboration commands are confusing when many elements are generated. The primitives for specifying constraints on other document elements are navigational rather than being based on element types, which often makes constraint specifications hard to understand.

Performance: The runtime performance of Proteus does not scale well as document size increased because of a naive implementation of the constraint system. For most documents, the time to reformat increases as the square of the document's size.

Dependence on Ensemble: Proteus was designed as a module of Ensemble, not an independent tool. Thus, it assumed the availability of many of Ensemble's features and could not be used by other programs. Furthermore, there is no experience using Proteus with applications other than Ensemble, so the suitability of its presentation services for other applications has not been conclusively demonstrated.

Configuration Specifications: Currently, Proteus is configured by a series of hand-coded function calls. The configuration process should be performed using a simple specification language.

2.3 Attempts to Improve Proteus

The performance problems with Proteus stimulated additional research.

Muthukkaruppan developed SPINE [18], an incremental attribute grammar system for use as a presentation specification system in Ensemble. SPINE's runtime performance was linear in the size of the document, but its specification language was low-level and difficult to use and the system could not support tree elaboration. Unlike Proteus, SPINE requires that documents adhere rigidly to their grammars and is not well-behaved when this rule is violated. Furthermore, SPINE specifications had to be translated into C++, compiled, and dynamically-linked into Ensemble, a process that would be difficult even for relatively sophisticated end-users.

Mittal developed another presentation system for Ensemble, SHILPĒ [16], which preserved Proteus's specification language while using an incremental and lazy runtime system based on Hudson's work on incremental attribute evaluation [11]. SHILPĒ's runtime performance is linear in the size of the document, but it requires about five times as much memory per document tree node as Proteus and SPINE. Like Proteus, SHILPĒ does not require any knowledge about the the document's grammar in order to run correctly.

2.4 A General Model of Media

Proteus was designed to be configurable and extensible because Ensemble was designed to have separate modules for each medium that would be added incrementally. This is, in fact, how the system was developed, starting with initial support for text and followed by later modules supporting two-dimensional graphics and digital video. Proteus was adapted to support these new media without any substantial modifications, proving the sufficiency of the configuration system.

Based on this success, it is natural to conclude that the configuration specifications express fundamental qualities of the media

that Ensemble supports. In fact, I have begun to develop a general model of media based on experience with Proteus. The concept of a medium is defined as follows.

Definition 1 *A* medium *is a triple $M = (T, D, O)$, where T is a set of primitive data types, D is a set of dimensions in which layout is performed, and O is a set of formatting operations with typed parameters.*

Based on this definition, it is possible to define rules of equivalence for media.

Definition 2 *Two media, M_1 and M_2 are* equivalent *iff the three sets that comprise them are equivalent. That is,*

$$M_1 \equiv M_2 \quad \textit{iff} \quad \begin{array}{l} T_1 \equiv T_2, \\ D_1 \equiv D_2, \textit{and} \\ O_1 \equiv O_2. \end{array}$$

Definition 3 *Two formatting operations, o_1 and o_2 are* equivalent *iff they take the same set of parameters, and given the same parameter values and input data, they produce identical output.*

The connection between the configuration specifications of Proteus and the definition of a medium is clear. The tree elaboration service needs to know the set of primitive types T and how to generate nodes of each type in the set. The box layout service needs a description of each dimension in the set D. The attribute propagation service needs to know the names and types of the parameters used by the formatting operations in the set O.

Still, the model is by no means complete. The most important problem is that the test for equivalence of media is very fine-grained. In order for two media to be considered "equivalent," there cannot be even the slightest difference between them. This doesn't match our intuitive notion of a medium, which is a broad and vaguely-defined category, such as "text," "graphics," or "video." More work needs to be done in order to clarify how the definition can be used to understand similarities between media, as well as their differences. Other aspects of the model which need more exploration are the role of user interaction in defining a medium, whether or not interface functions are part of a medium, whether there is a fixed relationship between dimensions in different media, and whether some media have discrete dimensions.

There is a compelling need for a sound and complete model of media. Common usage of the term "multimedia" often equates it with anything supporting digital video and audio, even though the speaker may, in the next breath, name static media like text and two-dimensional graphics as examples of media. Furthermore, it is common to equate the concept of a medium with a data type. This is implicit in the following definition of multimedia by Buford.

> It is the simultaneous use of data in different media forms (voice, video, text, animations, etc.) that is called multimedia. [3, page 2]

The data type definition is also implicit in Herlocker and Konstan's paper on "Commands as Media: ..." [10] which presents a system that added commands in the Tcl extension language to the data stream managed by the Berkeley Continuous Media Toolkit [21]. In this paper, no discussion is made of why the presence of commands in the same stream as video and audio data makes commands worthy of the label "medium."

A model of media which looks only at data types is not satisfactory. It opens the door to labeling any arbitrary data type as a medium. Furthermore, it is possible to identify pairs of applications that are conventionally perceived to support different media (such as LaTeX and Adobe Illustrator) while supporting nearly identical sets of data types (text and two-dimensional graphics primitives). The descriptions that are used to configure Proteus show that there is a more useful way to think about media than to view them simply as data types.

2.5 Current Status and Future Research

Recent work has produced an independent Proteus library which uses the runtime system of SHILPĒ. This library is written in about 7000 lines of C++. Future research on the internals of Proteus will explore performance enhancements, improvements to the language design, and new presentation services.

Work is proceeding on integrating Proteus into three existing applications: NCSA Mosaic, a commercial HTML editor for Microsoft Windows, and Doc, a text editor which uses the Interviews X11 toolkit [4] and which will be the basis for an SGML editor. An important goal of this research is to demonstrate that a system like Proteus can be used to move the World Wide Web away from the simple document representation of HTML toward arbitrary document structures specified by a standard like SGML. In the future, Proteus will be applied to a number of applications in other media, including digital video and three-dimensional graphics.

3 IMPLICATIONS FOR INFORMATION INTERCHANGE

Research on information interchange is primarily concerned with communication between intelligent software systems, but an intelligent software system must also interchange information with *people* and will likely do so by producing some kind of document. Beyond the daunting problem of choosing which information these documents should contain, such a system must also solve two presentation problems, media allocation (determining which media to use) and media realization (given the media, determining how to layout and format the information) [15]. Research on Proteus is relevant to these problems for two reasons. First, Proteus can be a valuable tool for media realization. Second, the model of media developed for Proteus may provide new ideas for the ways that intelligent systems reason about media.

3.1 Media Realization

Proteus is designed to help *people* perform media realization for structured documents. It gives them a uniform language for specifying layout and formatting, no matter what medium is being used, and also supports multiple presentations of the same document.

Perhaps Proteus can provide the same service to intelligent software systems. Rather than reasoning about how to present information from first principles, an intelligent system could produce its information in the form of structured documents and could then reason about the best presentation style for each document. This would free the intelligent system from low-level decisions about colors, fonts, and line widths, while preserving the system's ability to control appearance.

Another reason to use technology like Proteus is to increase configurability and the level of user control. The assumption in most intelligent multimedia presentation systems is that the system, not the user, should decide how to present material. This approach reduces the number of decisions that the user must make and can work well for the average case, but it fails to take into account the wide variations in human perception and preference. Individuals vary considerably in their ability to perceive color, to accurately estimate areas, and to process tabular data.

Furthermore, the user may be better qualified to determine how information needs to be displayed than the system, particularly when the information is complex. In their survey of intelligent multimedia presentation systems, Roth and Hefley note that the presentation of more complex information remains a difficult problem [20] If an intelligent system determines how to present information from first principles, it may be difficult for the user to specify an alternate presentation style. In contrast, if the system chooses the presenta-

tion from a palette of styles, then a dissatisfied user can choose a preferred style from the same palette.

3.2 Models of Media

The model of media on which Proteus's configurability is based was presented in Section 2.4. Arens, Hovy, and Vosser [1] have independently developed a very rich model of the knowledge required to create a multimedia presentation, which I call the AHV model. They have identified key characteristics of media which they use as the basis for reasoning about media allocation.

The two models have some similarities, notably the use of dimensions as a key characteristic. However, there are several differences, primarily because Proteus is designed for formatting, while the AHV model is primarily designed for choosing media.

- Unlike the AHV model, Proteus does not distinguish between spatial and temporal dimensions — the same layout rules that apply to space can also be applied to time. While the temporal dimension is clearly different from the spatial dimensions in important ways, the differences do not substantially change how layout is specified.
- The AHV model does not identify primitive data types as characteristics of a medium.
- The AHV model does not identify formatting operations as characteristics of a medium. However, the model does have elements called *channels* which appear to be analogous to the parameters of Proteus's formatting operations.
- The AHV model distinguishes *carriers*, which convey the central information, from the background. Proteus does not distinguish between foreground and background elements — all elements must be presented and are controlled in similar ways.

4 CONCLUSIONS

This paper has described the Proteus system, a tool for managing presentation of structured documents. Proteus provides its users with a uniform language for specifying presentation, no matter what medium is to be used. It also provides support for synchronized multiple presentations, allowing the user to view the same document in several styles simultaneously even while the document is being edited. Technology like that provided by Proteus is critical for the continued development of easily-exchanged documents which are defined using higher-level paradigms, such as SGML, that emphasize logical structure and content rather than appearance.

Tools like Proteus can also be useful to intelligent software systems by providing a higher-level mechanism for controlling document or user interface appearance. Rather than having to reason from first principles about how to present a document, an intelligent system that uses Proteus can choose presentation styles from a predefined palette. Furthermore, the system's users can be given more control because they can pick alternate presentation styles from the same palette.

One result of research on Proteus has been the development of a model of media that is the basis for Proteus's configurability. This model is related to the characteristics of media identified by Arens, Hovy, and Vosser [1]. Future research may bring these models closer together.

References

[1] Yigal Arens, Eduard H. Hovy, and Mira Vossers. On the knowledge underlying multimedia presentations. In Mark T. Maybury, editor, *Intelligent Multimedia Interfaces*, chapter 12. AAAI Press/MIT Press, 1993.

[2] T. Berners-Lee and D. Connolly. Hypertext markup language - 2.0. An Internet-Draft available by ftp from ds.internic.net, June 1995.

[3] John F. Koegel Buford. *Multimedia Systems*. SIGGRAPH Books. ACM Press, New York, 1994.

[4] Paul R. Calder and Mark A. Linton. Glyphs: Flyweight objects for user interfaces. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, pages 92–101. ACM Press, October 1990.

[5] James Clark. DSSSL Lite. Available on the WWW at URL `http://www.jclark.com/dsssl/`, 1995.

[6] Charles F. Goldfarb, editor. *Information Processing — Text and Office Systems — Standard Generalized Markup Language (SGML)*. International Organization for Standardization, Geneva, Switzerland, 1986. International Standard ISO 8879.

[7] Susan L. Graham. Language and document support in software development environments. In *Proceedings of the Darpa '92 Software Technology Conference*, Los Angeles, April 1992.

[8] Susan L. Graham, Michael A. Harrison, and Ethan V. Munson. The Proteus presentation system. In *Proceedings of the ACM SIGSOFT Fifth Symposium on Software Development Environments*, pages 130–138, Tyson's Corner, VA, December 1992. ACM Press.

[9] Grif SA marketing information, 1994. Available from Grif SA, Immeuble "Le Florestan", 2, boulevard Vauban, B.P. 266, St. Quentin en Yvelines, 78053 Cedex.

[10] Jonathan L. Herlocker and Joseph A. Konstan. Commands as media: Design and implementation of a command stream. In *Proceedings of the Second ACM International Conference on Multimedia*, San Francisco, CA, November 1995. Also available at URL `http://www.cs.umn.edu/users/konstan/`.

[11] Scott E. Hudson. Incremental attribute evaluation: A flexible algorithm for lazy update. *ACM Transactions on Programming Languages and Systems*, 13(3):315–341, July 1991.

[12] International Standards Organization. *Office Document Architecture*, 1986. Draft International Standard 8813.

[13] ISO/IEC. *Information technology — Text and office systems — Document Style Semantics and Specification Language (DSSSL)*, August 1994. Draft International Standard ISO/IEC DIS 10179.2.

[14] Leslie Lamport. *LaTeX: A Document Preparation System. User's Guide and Reference Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

[15] Mark T. Maybury. Introduction. In Mark T. Maybury, editor, *Intelligent Multimedia Interfaces*. AAAI Press/MIT Press, 1993.

[16] Alok Mittal. SHILPÊ: A presentation system for ensemble. Master's thesis, University of California, Berkeley, California, December 1995.

[17] Ethan Vincent Munson. *Proteus: An Adaptable Presentation System for a Software Development and Multimedia Document Environment*. PhD dissertation, University of California, Berkeley, December 1994. Also available as UC Berkeley Computer Science Technical Report UCB/CSD-94-833.

[18] Kannan Muthukkaruppan. SPINE, a synthesizer for practical incremental evaluators. Master's thesis, University of California, Berkeley, CA, May 1994.

[19] Brian K. Reid. *Scribe: A document specification language and its compiler*. PhD dissertation, Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania, October 1980. Available as technical report CMU-CS-81-100.

[20] Steven F. Roth and William T. Hefley. Intelligent multimedia presentation systems: Research and principles. In Mark T. Maybury, editor, *Intelligent Multimedia Interfaces*, chapter 1. AAAI Press/MIT Press, 1993.

[21] Lawrence A. Rowe and Brian C. Smith. A continuous media player. In *Proceedings of the Third International on Network and Operating Systems Support for Digital Audio and Video*, pages 376–86,.1994.

[22] Synex Information AB's home page. World Wide Web home page at `http://www.synex.se`, 1995.

Information Brokers: Gathering Information from Heterogeneous Information Sources

Richard Fikes, Adam Farquhar, Wanda Pratt
Knowledge System Laboratory
Stanford University
{fikes,adam_farquhar,pratt}@ksl.stanford.edu

Abstract

The Internet provides dramatic new opportunities for gathering information from multiple, distributed, heterogeneous information sources. However, this distributed environment poses difficult technical problems for the information-seeking client, including finding the information sources relevant to an interest, formulating questions in the terms that the sources understand, interpreting the retrieved information, and assembling the information retrieved from several sources into a coherent answer. In this paper, we describe techniques that will enable vendors and buyers to build and maintain network-based *information brokers* capable of retrieving information about services and products via the Internet from multiple vendor catalogs and data bases for both human and computer-based clients.

1. INTRODUCTION

The Internet provides dramatic new opportunities for gathering information from multiple, distributed, heterogeneous information sources. However, this distributed environment poses difficult technical problems for the information-seeking client, including finding the information sources relevant to an interest, formulating questions in the terms that the sources understand, interpreting the retrieved information, and assembling the information retrieved from several sources into a coherent answer. In this paper, we describe techniques that will enable vendors and buyers to build and maintain network-based *information brokers* capable of retrieving information about services and products via the Internet from multiple vendor catalogs and data bases for both human and computer-based clients.

The ability to obtain relevant information in a timely and cost efficient manner is central to the performance of most tasks. The widespread availability of computer-based information brokers will provide that ability by facilitating access to the broad range of information that is rapidly becoming available on the Internet. The general availability of the technology to build and maintain information brokers will enable the establishment of an industry whose primary products are computer-based network-accessible brokering services.

2. TECHNICAL BARRIERS

Effective information brokering involves a level of understanding of the information being brokered that requires the use of symbolic domain models to reason about relevance. Information brokering requires many capabilities, including

- Helping a human or computer client formulate a query in the broker's vocabulary about some class of products or services.
- Identifying information sources that are relevant to answering a query.
- Generating a plan to answer the query using the given set of relevant information sources.
- Executing the plan and integrating information from multiple sources. This involves translating the query into the information source's vocabulary and syntax, obtaining responses to the query, and translating the responses into the broker's vocabulary and syntax.
- Presenting the responses to the client. This involves explaining to a client how the response relates to the query, defining the terms used in the response, and suggesting alternative queries that may provide additional relevant information.

While some of these tasks are characteristic of many information retrieval activities, the Internet environment imposes special requirements: the need to deal with variety, change, and autonomy of both clients and information sources. The information seeking client:

- May be a human or a software agent representing a human's interests.
- Does not know the vocabulary or access methods of all the information sources.
- May not know the vocabulary or the range of services of the information broker.

The information sources:

- Are created and maintained by independent information providers.. A broker must provide value to the information providers so that they will be motivated to couple their sources to the broker.
- Have heterogeneous access methods (e.g., SQL databases, information agents, WAIS or HTTP document servers).
- May cover different domains to different degrees and use different vocabularies to model the domain.
- May be fully structured (e.g., database relations, sentences in a logic) or semi-structured (natural language documents on a document server, e-mail, indexed multimedia).
- May return information that is incomplete or irrelevant to a query.
- Are subject to change over time in both terminology and available information.

Because widespread use of the Internet is a relatively new phenomenon, current information retrieval technologies do not adequately deal with these problems.

Our main technical claim is that domain-independent information gathering schemes are limited to syntactic matching techniques and are too weak for effective information brokering. Like human brokers, effective computer-based information brokers will take advantage of specialized domain knowledge, such as:

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/192

- The terminology used to describe products in that domain.
- Functional descriptions of products that support queries from users who need products providing a specific functionality but who do not know the type of product that can supply this functionality.
- Abstractions and assumptions that will enable the agent to retrieve information that is relevant to a query.
- Methods for appropriately combining and summarizing retrieved information.

Building such brokers as ad hoc, monolithic applications will not scale, and the resulting brokers will not be able to interoperate with the new protocols and services being developed for the Internet.

3. AN INFORMATION BROKER ARCHITECTURE

We are developing a detailed architecture for network-based, domain-specific information brokers which is shown in Figure 1. The architecture includes the following modules:

Domain Model — A logical theory which describes the broker's domain of expertise. The theory specifies the broker's vocabulary of objects, relations, functions, and product classes that it uses to model the domain. The theory describes tasks that typical users of the system perform, the functions of the products in the domain, heuristics for identifying relevant information sources, etc.

Source Models — Structured descriptions of the *competence* of each information source that the broker uses. Each description includes a logical theory of the portion of the broker's domain of expertise about which the source provides information. The theory describes the source's vocabulary in which it accepts queries and provides information. A source description also includes specifications of which relations the source can provide instances of, whether the source has complete information about some relation, the cost of accessing the source, etc.

Formulator — Assists a client in the formulation of queries in the broker's vocabulary. The formulator includes:

Product Description Browser — A service for informing clients about the broker's query vocabulary and domain of expertise. The service provides a browsable product description taxonomy consisting of hierarchies of object-oriented product descriptions from the broker's domain model.

Query By Reformulation Assistant — A service which helps a client iteratively refine a query by providing example responses to portions or all of the query.

Alternatives Advisor — A proposer of alternative queries which may provide additional useful information or better satisfy a client's goals.

Planner — Formulates a plan for answering a query. The plan specifies a sequence of subqueries to find instances of a given relation or members of a given class, constraints against which retrieved instances are to be filtered, and answer composition operations to be performed on retrieved descriptions.

Executor —Answers the query by performing the query plan. It includes the following submodules:

Source Identifier — Identifies information sources that are relevant to a subquery by determining which source model contains relations or classes whose instances can be used to answer the subquery.

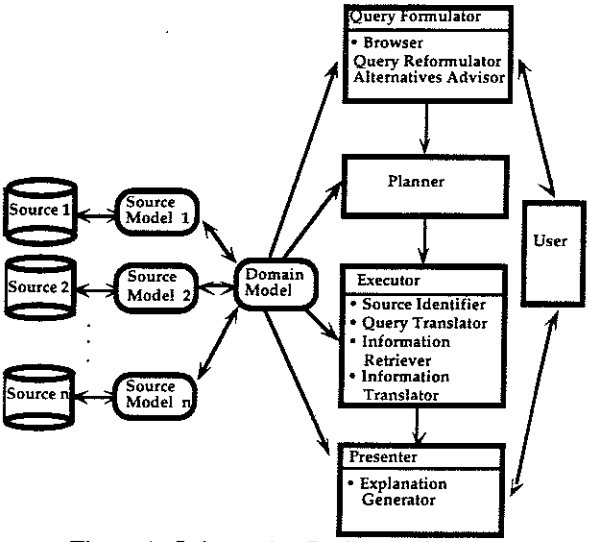


Figure 1: Information Broker Architecture

Query Translator — Translates a subquery from the broker's query language to each source's language.

Information Retriever — Obtains product descriptions by sending translated queries to information sources.

Information Translator — Translates product descriptions obtained from information sources into the broker's vocabulary and syntax.

Presenter — Presents to a client in an appropriate format the broker's response to a query. The response may include information that is known to answer the query, describes likely answers to the query, is relevant to answering the query, elaborates the answers to the query, or explains the answers to the query. The presenter makes use of an:

Explanation Generator — Explains the relevance of information presented in response to queries and the meanings of the terms in the presented information.

3.1. Domain and Source Modeling

We are developing brokers that maintain declarative, logic-based, object-oriented models of their domain of expertise and of the domains of expertise of each of their information sources. The broker's domain model specifies the vocabulary of object, relation, function, and product class names that the broker's clients can use to formulate queries. Each source description includes a logical theory of the broker's domain of expertise. The theory describes the vocabulary in which the source accepts queries and provides information.

We are developing a tool kit for broker developers based on the tools in the Ontolingua system (Gruber 1992; Gruber 1993a; Gruber 1993b). Ontolingua, which is being developed by the Knowledge Systems Lab (KSL) as part of the Knowledge Sharing Initiative (Fikes et al. 1991a; Patil et al. 1992), is an integrated tool system for developing domain-specific ontologies in the Knowledge Interchange Format (KIF) (Genesereth and Fikes 1992; Neches et al. 1991) and for translating the resulting ontologies into application-oriented representation languages. An ontology is a set of

relations and axioms that attempt to precisely characterize some domain of discourse. Ontolingua:

- Augments KIF with a frame language ontology that provides convenient representation primitives for specifying class-subclass taxonomies in an object-oriented style.
- Identifies many classes of errors in ontology specification such as under-constrained variables, undefined concepts, and missing theory inclusion relationships.
- Produces hypertext documentation for browsing.
- Provides high fidelity translation into multiple representation languages, including LOOM (MacGregor 1991), Prolog, the CORBA Interface Definition Language (IDL), and C-based CLIPS.

The hypertext documentation produced by Ontolingua is of particular importance to information brokering because it enables the definitions of product description terminology to be easily accessed by both broker developers and clients. Ontolingua generates hypertext webs of ontologies in the format of the World Wide Web (WWW) that read like reference manuals instead of source code. Putting ontologies in WWW format also makes it easy to integrate formal KIF theories with semiformal text used in documentation. For example, the introductory documents explaining the purpose and use of an ontology can have hypertext pointers directly from the use of a word in text to its formal definition in KIF. Similarly, terms used in the formal definitions can point directly to their definitions.

3.2. Domain-Specific Query Formulation Assistance

An important service that the information broker provides is assistance with the formulation of queries. For example, clients may not know or understand the idiosyncratic vocabularies used by vendors to describe the features of their products and may not know how to relate their functional objectives for the product to those product feature descriptions. The broker must use its knowledge to appropriately constrain the query and elicit information that will be sufficient to answer it. Furthermore, the broker must ensure that an answer to the query will provide sufficient information to satisfy the client's goals.

The broker architecture includes the following query formulation facilities:

- A product description browsing service that explains the vocabulary used in the descriptions.
- A query vocabulary for retrieving products by functional objectives.
- A "query by reformulation" service which helps a client iteratively refine a query by providing example responses to portions or all of the query.
- A proposer of alternative queries which may provide additional useful information or better satisfy a client's goals.

The basic facility in our broker architecture for assisting clients with query formulation is a browsable product description taxonomy consisting of hierarchies of object-oriented product descriptions from the broker's domain model. The class descriptions in the taxonomies will indicate to a client both the types of products accessible by the broker and the vocabulary that can be used in queries. The product taxonomies will be accessible as both a fully cross-referenced HTML document for browsing by human clients and as a formally defined knowledge base of structured descriptions for computer-based clients.

A broker can assist a client by providing a sample of possible answers to a query or portions of a query in situations where finding all answers to the complete query would require significant time or produce a large number of answers. The user then has the option of refining the query based on the feedback or authorizing the broker to continue retrieving answers to the initial query. If undesired answers are returned by the query, the client can examine those items to determine which characteristics would have eliminated those items. The client can use this information to appropriately refine the query. The descriptions may also suggest additional features that the client had not previously considered. Such *query by reformulation* techniques (Tou et al. 1982; Yen, Neches, and DeBellis 1988) are particularly useful for assisting clients who are unfamiliar with the descriptive vocabulary available for use in queries or with the range of information that is available for responding to queries.

An important application of the information broker's domain-specific expertise is the ability to suggest alternative queries to a client that may provide a more desirable solution to the client's goals. (E.g., consider departing from a neighboring airport when all flights are booked or extending a trip over a Saturday night to reduce the cost of the airline ticket.) In order to suggest useful alternatives, the broker needs to be able to assume or determine the client's goals and to know for a given query-goal pair what alternative queries might be useful.

3.3. Query Planning

Given a query from a client in some formal language, an information broker must decompose the query into a sequence of subqueries that can be sent to relevant information sources, constraints against which retrieved descriptions are to be filtered, and answer composition operations to be performed on retrieved descriptions. We expect each subquery to be a request for tuples that satisfy a given relation or structured descriptions of members of a given class.

We will develop a query planner to perform this decomposition by adapting existing techniques from the data base community to the information brokering environment. Because of the dynamic nature of the network environment in which information brokers will work, we anticipate a strong need for intermixing query planning and execution. That is, because the planner may not know what information sources are available to answer the query or how many answers may be found for a given subquery, the planner may not be able to form a complete plan that will be effective or efficient. We will explore strategies which execute the next steps in a query plan as soon as they are determined and then continue generation of the remainder of the plan taking into account the results of the steps already executed.

3.4. Query Plan Execution

The query plan executor answers the query by identifying which information sources are relevant to answering a subquery, translating a subquery into the vocabulary and syntax of each identified source, retrieving information by sending translated queries to each identified source, and translating the answers retrieved by an information source into the broker's

Table Name			
Product	**name**	**size**	**cost**
	television-1	19	256
	simm-1	256	8
Product-Type	**name**	**type**	
	television-1	television	
	simm-1	memory-chip	

Figure 2: Partial Contents of Database

Table Name	Field	Data Type
Product	name	char
	size	int
	cost	int
Product-Type	name	char
	type	char

Figure 3: Schema Definitions

vocabulary. We will focus on translation problems in query plan execution.

We use a context logic (Buvac and Mason 1993; Guha 1991; McCarthy 1993) to represent the schemata and views of the individual information sources, shared ontologies, implicit semantics, and integrated information. Context logic allows us to incrementally strengthen the representation of an information source. An initial representation can be generated automatically from an information source's schema. The representation can be strengthened over time by adding axioms that make explicit the assumptions behind the schema and translate from the information source's vocabulary into the terms that are shared with other information sources. Clients may query the information source as in a loosely coupled system that provides a uniform query interface to a number of heterogeneous systems.

Consider the simple relational database specified in Figure 2. Using the schema definition in Figure 3, the tables of a relational database can be translated into assertions in first order logic, as shown in Figure 4. Each table in the database corresponds to a relation in the logic and each database schema definition corresponds to an axiom.

```
(∀ x,y,z product(x, y, z)
  ⇒ string(x) & integer(y) & integer(z))
relation(product) & arity(product, 3) &
  primary-key(product, 1)
(∀ x,y,z product_type(x, y)
  ⇒ string(x) & string(y))
relation(product_type) & arity(product, 2) &
  primary-key(product_type, 1)
```

Figure 4: Representation of the product and product-type schema in logic.

Although this context represents the database schema and its contents in logic, it suffers from two basic problems:

- The representation does not resolve the ambiguities when attributes are used polymorphically within a single table. For example, the size attribute in the product database stores the size in whatever unit is appropriate for the particular product. For televisions this may be the number of inches across the diagonal of the screen, for memory chips this may be in kilobytes. A logician might consider this to be a sloppy representation, but it is typical of the representations that occur in actual databases.
- The representation allows values to have a non-unique denotation. A single value may be used in a database to mean different things in different tables, tuples, or columns. For example, in the product database of Figure 1b, the number 256 appears in both the size and cost columns. This is not, in itself, an inconsistency. If we want to make the units explicit, however, care must be taken to avoid assigning incompatible interpretations to different occurrences of 256.

These problems are related, and can be solved with a combination of existential quantification and systematic renaming. For example, we could write an axiom to disambiguate the product relation such as:

```
∀ x,y,z product(x, y, z) ⇔
    (∃ y',z' product-1(x, y', z') &
      magnitude(y', us-dollars) = y)
```

That is, we could introduce a new relation `product-1` that is similar to product, except that new objects, y' and z', are introduced to represent the size and cost. The magnitude of the cost, y', must then be equal to the number in the database product table. The existential quantification is an important, but straightforward, trick. The renaming, however, is rather clumsy, and it becomes even more awkward once multiple information sources are integrated.

Context logic provides us with a more elegant and powerful mechanism to overcome the representational problems without the clumsy renaming. Context logic (McCarthy 1993)is an extension of first order logic in which sentences are not simply true, but are true within a context. The key extension is a modality **`istrue`**, which takes two arguments: a context and a sentence. It asserts that the sentence is true in the specified context. Contexts are logical individuals and, as such, can be quantified over. Furthermore, it is possible to write axioms that span several contexts. An axiom that lifts sentences from one context to another context is known as a **lifting axiom**. Lifting axioms provide a very powerful and expressive means of shifting information from one context to another. They can be used to perform renaming, change structure, and make implicit assumptions explicit. Context logic allows us to restate the disambiguation axiom without renaming:

```
istrue(c1, product(x,y,z)) ⇔
  istrue(c2, (∃ y',z' product(x,y',z') &
          magnitude(y', us-dollars)= y))
```

Two contexts are used to represent each information source. The **syntactic context** is a direct translation of a database schema into logic *without* resolving semantic conflicts, so that the translation can be done automatically. The **semantic context** holds the translation with the semantic conflicts resolved. The lifting axioms that perform the translation from the syntactic context into the semantic context cannot be automatically generated, because they are making the semantics that were not represented in the database schema explicit. Figure 5 shows lifting axioms to define the semantic context for the product database.

```
istrue(SemCl, product_type(x, y)) <==
  istrue(SynCl, product_type(x, y))
istrue(SemCl, ∃ y',z'
  (magnitude(y', natural-size-units (x))=y &
    magnitude(z', us-dollar) = z))
  <== istrue(SynCl, product(x, y, z)))
istrue(SemCl, natural-size-units(x)=bit*1024
  <== product-type(x, memory-chip))
istrue(SemCl, natural-size-units(x) = inch
  <== product-type(x, television))
```

Figure 5: Lifting axioms between the syntactic context and the semantic context

The first axiom simply lifts all `product-type` facts from the syntactic context into the semantic context. The second axiom lifts tuples from the `product` table into the semantic context, but it disambiguates the meaning of the numbers in the table. Every number in the cost column becomes a quantity whose magnitude, when measured in US dollars, is the original number. Translating the size column is somewhat more complicated because the unit varies with the type of the product. The last two axioms associate a product type with the unit most naturally used to measure its size.

The author of the lifting axioms must choose an appropriate way to represent the intended implicit semantics of the database. In Figure 5, the functions magnitude and natural-size-units were used along with the constants us-dollar, bit, and inch. The decisions required to construct these representations can be subtle. For instance, we have chosen to use a magnitude function that takes two arguments: a quantity and a unit. This is because the unit used to measure a quantity is not an inherent property of the quantity, whereas its dimension is. For example, the dimension of all prices is currency, but the magnitude of a price can be measured by any unit of currency such as dollars or yen.

3.5. Results Presentation and Explanation

Effective information brokering requires presenting information obtained in response to a query in an easily understandable format and assisting the client in understanding that information. The task is nontrivial because the results may not provide precisely the information needed or intended. The query reformulation process will typically be incomplete and approximate so that the response may include information that is known to answer the query, describes likely answers to the query, is relevant to answering the query, or elaborates the answers to the query. Also, when the amount of information gathered is large, summarization will be required for human readers.

In order to enable brokers to assist their clients in understanding retrieved information, we will include in our broker architecture an explanation generation facility that can be used to provide clients with explanations of the rationale for the relevance of information presented in response to queries and of the meanings of the terms occurring in the presented information. For example, if the query is to find airline flights from London to Washington D.C. on a given date, and the retrieved information describes flights from Heathrow to Dulles, an explanation could be added to Heathrow (e.g., as a hypermedia link) saying that it is a London airport and to Dulles saying that it is a Washington D.C. airport.

We are developing tools that enable a broker to compose explanations from term definitions and from the inferences it makes, and to annotate the information it provides to a client with those explanations, either as hypertext links for human clients or as relational links for computer-based clients. The tools are based on the explanation technology we have developed in the How Things Work Project (Fikes et al. 1991b) for providing interactive documentation of engineering designs. That technology dynamically generates device descriptions and causal explanations of simulated device behavior from symbolic device models, mathematical simulation models, and simulator output (Gautier and Gruber 1993). The explanations are produced as HTML documents that are generated dynamically by a network server when a user requests an explanation.

4. RELATED WORK

There is a growing body of work that addresses the problems of gathering and integrating information from multiple heterogeneous information sources. Three relevant pieces of work include SIMS (Arens et al. 1993; Arens and Knoblock 1992), CARNOT (Huhns et al. 1992), and the work of Siegel and colleagues (Goh, Madnick, and Siegel 1994; Sciore, Siegel, and Rosenthal 1994). The work described in this paper differs in three major ways. First, the context logic provides a sound formal basis for describing the semantics of multiple sources and for articulating the relations between them. Second, a library of reusable ontologies reduces the cost of building these discriptions and helps users to understand the meaning of the terms that they can use in their queries. Third, the explanation techniques help users to understand the results of a query.

5. DISCUSSION

There are several powerful consequences of using our approach to integrate information sources including the ability to:

- Integrate new information sources incrementally.
- Share assumptions among information sources without making them explicit.
- Exploit shared ontologies.
- Provide a richer model of integration that goes beyond global schema or federated schema methodologies.

One important consequence of our approach is that it eliminates most of the up-front cost of integrating a new information source. The cost is reduced because the information source context can be automatically generated from the source's export schema. Once this has been done, it is possible to make queries in the context of the new information source as if it were a loosely coupled heterogeneous database system. The query must be expressed in the vocabulary of the new source, but the syntax and interface are consistent with the old sources. Once the information source context has been established, it is possible to incrementally add lifting axioms to populate the source's semantic context. Furthermore, this incremental integration can be performed in response to perceived and actual usage patterns, rather than expectations about usage. Our approach is in stark contrast to the global schema approach in which the ontology of the new

information sources must be completely decontextualized and translated into the existing global schema.

It is possible for a new information source to exploit commonalties with an existing source in two ways. First, we can copy lifting axioms from the old source into the new source's semantic context. Second, we can write lifting axioms that map from the new source's context into the old source's context. The key point here is that the relations between contexts are much richer than theory inclusion. Lifting axioms may connect sibling contexts and exploit commonalties in numerous ways. This is similar to, but more powerful than, the federated database approach in which the schemas of subsets of the databases known to the federation are combined. Context logic enables shared implicit assumptions to be shared across information sources without the need to first disambiguate them.

The semantic contexts make use of shared ontologies. Ontologies for domains such as quantities, finance, product descriptions, and so on, will ease the work of integrating information sources. In our formulation, each ontology is defined in its own context. The semantic contexts of information sources may then include the ontology contexts. If the semantic contexts of several information sources share common ontologies, it is much easier to operate across them. This is one way of decomposing the otherwise daunting problem of constructing a global context.

We have presented an information broker architecture that will help with query formulation, identify information sources relevant to those queries, retrieve information from various information sources, and present the responses as well as relevance rationale. Our approach focuses on using domain knowledge and context logic to provide a scalable approach to gathering information from heterogeneous sources.

6. REFERENCES

Arens, Y., C. Y. Chee, C.-N. Hsu, and C. A. Knoblock. 1993. Retrieving and Integrating Data from Multiple Information Sources. International Journal on Intelligent and Cooperative Information Systems 2 (2): 127-158.

Arens, Y. and C. Knoblock. 1992. Planning and Reformulating Queries for Semantically-modeled Multidatabase Systems. In Preceedings of the 1st International Conference on Information and Knowledge Management:92-101.

Buvac, S. and I. Mason. 1993. Propositional Logic of Context. In Proceedings of the Eleventh National Conference on Artifical Intelligence:412-419: AAAI Press/MIT Press.

Fikes, R., M. Cutkosky, T. Gruber, and J. van Baalen. 1991a. Knowledge Sharing Technology Project Overview. KSL 91-71. Stanford University, Knowledge Systems Laboratory.

Fikes, R., T. Gruber, Y. Iwasaki, A. Levy, and P. Nayak. 1991b. How Things Work Project Overview. KSL 91-70. Stanford University, Knowledge Systems Laboratory.

Gautier, P. O. and T. R. Gruber. 1993. Generating Explanations of Device Behavior Using Compositional Modeling and Causal Ordering. In Proceedings of the Eleventh National Conference on Artificial Intelligence. Washington, D.C.: AAAI Press/The MIT Press.

Genesereth, M. R. and R. E. Fikes. 1992. Knowledge Interchange Format, Version 3.0 Reference Manual. Logic-92-1. Computer Science Department, Stanford University.

Goh, C. H., S. E. Madnick, and M. D. Siegel. 1994. Context Interchange: Overcoming the Challenges of Large-scale Interoperable Database Systems in a Dynamic Environment. In Proceedings of the Third International Conference on Information and Knowledge Management (CIKM-94). Gaithersburg, Maryland.

Gruber, T. R. 1992. Ontolingua: A mechanism to Support Portable Ontologies. KSL 91-66. Stanford University, Knowledge Systems Laboratory.

Gruber, T. R. 1993a. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In International Workshop on Formal Ontology, ed. Nicola Guarino. Padova, Italy.

Gruber, T. R. 1993b. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 5 (2): 199-220.

Guha, R. V. 1991. Contexts: A Formalization and Some Applications. PhD Thesis, Stanford University.

Huhns, M., N. Jacobs, T. Ksiezyk, W. M. Shen, W. Singh, and P. Cannata. 1992. Enterprise Information Modeling and Model Integration in CARNOT. In Enterprise Integration Modeling: Proceedings of the First International Conference: MIT Press.

MacGregor, R. 1991. The Evolving Technology of Classification-based Knowledge Representation Systems. In Principles of Semantic Networks: Explorations in the Representation of Knowledge, ed. John Sowa:385-400. San Mateo, CA: Morgan Kaufmann.

McCarthy, J. 1993. Notes on Formalizing Context. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence.

Neches, R., R. E. Fikes, T. Finin, T. R. Gruber, R. Patil, T. Senator, and W. R. Swartout. 1991. Enabling Technology for Knowledge Sharing. AI Magazine 12 (3): 16-36.

Patil, R. S., R. E. Fikes, P. F. Patel-Schneider, D. McKay, T. Finin, T. R. Gruber, and R. Neches. 1992. The DARPA Knowledge Sharing Effort: Progress report. In Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference. Cambridge, MA: Morgan Kaufmann.

Sciore, E., M. Siegel, and A. Rosenthal. 1994. Using Semantic Values to Facilitate Interoperability among Heterogeneous Information Systems. Transactions on Database Systems 19 (2): 254-290.

Tou, F. N., M. D. Williams, R. E. Fikes, D. A. Henderson, and T. W. Malone. 1982. RABBIT: An Intelligent Database Assistant. In *Proceedings of the National Conference on Artificial Intelligence*:314-318. Pittsburgh, PA.

Yen, J., R. Neches, and M. DeBellis. 1988. Specification By Reformulation: A Paradigm for Building Integrated User Support Environments. In Proceedings of the National Conference on Artificial Intelligence:814-818.

A Sketch of a Qualification Calculus

Klemens Schnattinger Udo Hahn

Freiburg University
Computational Linguistics Lab
Europaplatz 1, D-79085 Freiburg, Germany
{schnattinger,hahn}@coling.uni-freiburg.de

Abstract

We introduce a symbolic, qualitative model of uncertain reasoning and apply it to a concept acquisition problem in the framework of natural language text understanding. Considering uncertain reasoning as a choice problem between different alternatives (hypotheses), the model we provide assigns quality labels to single evidences for or against a hypothesis, combines the generated labels in terms of the overall credibility of a single hypothesis, and, finally, computes a preference order for the entire set of competing hypotheses. This model of quality-based uncertain reasoning is entirely embedded in a description logic framework.

1 INTRODUCTION

In this paper, we develop a symbolic, qualitative model of uncertain reasoning, one which is entirely embedded in a terminological reasoning framework. Decision-making under uncertainty can be considered as choice between several alternatives or hypotheses. The *qualification calculus* we propose can be considered a system of preference computations that treats the problem of choosing from among several alternatives as a *quality-based decision task* and decomposes it into three constituent parts: the continuous *generation* of quality labels for single hypotheses, the estimation of the overall *credibility* of single hypotheses, and the computation of a *preference order* for the entire set of competing hypotheses. The key notion of quality labels can be related to different types of evidences for or against single alternatives. Depending on the availability of observational instances that can be related to single evidences, the calculus we provide characterizes different types of evidences by certain quality labels which indicate the specific status, i.e., the significance, reliability and strength of the evidence under consideration.

This approach is motivated by requirements which emerged from our work in the overlapping fields of natural language parsing and learning from texts. Both tasks are characterized by the common need to evaluate alternative representation structures, either reflecting parsing ambiguities or multiple concept hypotheses. For instance, in the course of learning from texts, various and often conflicting concept hypotheses for a single item are formed as the learning environment usually provides only inconclusive evidence for exhaustively determining the properties of the concept to be learned. Moreover, in "real-world" natural language understanding systems, processing large text corpora, the underdetermination of results can often not only be attributed to incomplete knowledge provided for that concept in the source texts, but it may also be due to imperfect parsing results (originating from lacking linguistic or conceptual specifications, or ungrammatical input). Therefore, competing hypotheses at different levels of validity and reliability are the rule rather than the exception and, thus, require appropriate formal treatment. In order to deal with the emerging indeterminacy, e.g., in the learning task, two types of evidences are considered. The first one reflects *structural linguistic* properties of phrasal patterns or discourse contexts in which unknown words occur (assuming that the type of grammatical construction exercises a particular interpretative force on the lexical item to be learned). The second one reflects *conceptual properties* of particular concept hypotheses as they are generated and continuously refined by the ongoing text understanding process (e.g., consistency relative to already given knowledge, independent justification from several sources). Each of these grammatical, discourse or conceptual indicators is assigned a particular quality label. The application of quality macro operators, taken from the qualification calculus, to these atomic quality labels finally determines which out of several alternative hypotheses actually hold(s).

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/198 ©1996 FLAIRS

Syntax	Semantics
C_{atom}	$\{d \in C^{\mathcal{I}}_{atom} \mid C_{atom}$ is atomic$\}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$\exists R.C$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \cap C^{\mathcal{I}} \neq \emptyset\}$
$\forall R.C$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\}$

Table 1: Syntax and Semantics for Concept Constructors

Terminological A.	
Axiom	Semantics
$A \doteq C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$
$A \sqsubseteq C$	$A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
Assertional A.	
Axiom	Semantics
$a : C$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$

Table 2: Axioms for Concept Constructors

Syntax	Semantics
R_{atom}	$\{(d,e) \in R^{\mathcal{I}}_{atom} \mid R_{atom}$ is atomic$\}$
$R \sqcap S$	$R^{\mathcal{I}} \cap S^{\mathcal{I}}$
$c\|R$	$\{(d,d') \in R^{\mathcal{I}} \mid d \in C^{\mathcal{I}}\}$
$R\|_C$	$\{(d,d') \in R^{\mathcal{I}} \mid d' \in C^{\mathcal{I}}\}$
R^{-1}	$\{(d,d') \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (d',d) \in R^{\mathcal{I}}\}$
$C \times D$	$C^{\mathcal{I}} \times D^{\mathcal{I}}$
$(R_1,..,R_n)$	$R_1^{\mathcal{I}} \circ .. \circ R_n^{\mathcal{I}}$

Table 3: Syntax and Semantics for Role Constructors

Terminological A.	
Axiom	Semantics
$Q \doteq R$	$Q^{\mathcal{I}} = R^{\mathcal{I}}$
$Q \sqsubseteq R$	$Q^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
Assertional A.	
Axiom	Semantics
$a\ R\ b$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

Table 4: Axioms for Role Constructors

2 TECHNICAL FOUNDATIONS

Technically, we consider the problem of uncertain reasoning from a new methodological perspective, *viz.* one based on metareasoning about statements expressed in a terminological representation language. Based on the reification of terminological assertions, we use contexts for the encapsulation of qualifying reasoning processes and truth-preserving translation rules for the mediation between those contexts. Hence, we gain the full classification power from standard terminological systems for our metareasoning approach.

Description Logics. We use a standard concept description language, referred to as $\mathcal{CDL}$, which has several constructors combining *atomic* concepts, roles and individuals to define the terminological theory of a domain (for a subset, see Tables 1 and 3; cf. Woods & Schmolze (1992) for a survey of terminological languages).

Concepts are unary predicates, *roles* are binary predicates over a domain Δ, with *individuals* being the elements of Δ. We assume a common set-theoretical semantics for $\mathcal{CDL}$ — an interpretation $\mathcal{I}$ is a function that assigns to each concept symbol (the set **A**) a subset of the domain Δ, $\mathcal{I} : \mathbf{A} \rightarrow 2^{\Delta}$, to each role symbol (the set **P**) a binary relation of Δ, $\mathcal{I} : \mathbf{P} \rightarrow 2^{\Delta \times \Delta}$, and to each individual symbol (the set **I**) an element of Δ, $\mathcal{I} : \mathbf{I} \rightarrow \Delta$.

Concept terms and *role terms* are defined inductively. Tables 1 and 3 state corresponding constructors for concepts and roles, together with their semantics. C and D denote concept terms, while R and S denote roles. $R^{\mathcal{I}}(d)$ represents the set of *role fillers* of the individual d, i.e., the set of individuals e with $(d,e) \in R^{\mathcal{I}}$. $||R^{\mathcal{I}}(d)||$ denotes the number of role fillers.

REIF $\doteq$ $\forall$BINARY-REL.ROLES $\sqcap$ $\forall$DOMAIN.THINGS $\sqcap$ $\forall$RANGE.THINGS $\sqcap$ $\forall$HYPO-REL.HYPO

BINARY-REL $\sqsubseteq$ ROLES DOMAIN $\sqsubseteq$ ROLES

RANGE $\sqsubseteq$ ROLES. HYPO-REL $\sqsubseteq$ ROLES

Table 5: General Data Structure for Reification

$\Re(a : C) = r$: REIF $\sqcap$ r BINARY-REL INST-OF $\sqcap$ r DOMAIN a $\sqcap$ r RANGE C $\sqcap$ r HYPO-REL h

$\Re(a\ R\ b) = r$: REIF $\sqcap$ r BINARY-REL R $\sqcap$ r DOMAIN a $\sqcap$ r RANGE b $\sqcap$ r HYPO-REL h

Table 6: A Sketch of the Reification Functions $\Re$

By means of *terminological axioms* (for subsets, cf. Tables 2 and 4) a symbolic name can be defined for each concept and role term. We may supply necessary and sufficient constraints (using $\doteq$) or only necessary constraints (using $\sqsubseteq$) for concepts and roles. A finite set of such axioms is called the *terminology* or *TBox*. Concepts and roles are associated with concrete individuals by *assertional axioms* (see Table 2 and 4; a, b denote individuals). A finite set of such axioms is called the *world description* or *ABox*. An *interpretation* $\mathcal{I}$ is a model of an ABox with regard to a TBox, iff $\mathcal{I}$ satisfies the assertional and terminological axioms.

Reification. We here restrict ourselves to the reification of the assertional axioms. The remaining constructors can be reified in a straightforward way based on the scheme outlined below (cf. Schnattinger et al. (1995)).

We have chosen a particular "data structure", itself expressed in $\mathcal{CDL}$ (see Table 5) to make the reification format explicit. It provides the common ground for expressing qualitative assertions at various degrees of plausibility or credibility. ROLES is the concept for all roles including the relations INST-OF and ISA (it thus represents the set $\mathbf{P}^{ext} = \mathbf{P} \cup \{\text{inst-of, isa}\}$), THINGS is the (meta)concept for all concepts and instances (it represents the set $\mathbf{A} \cup \mathbf{I}$) and HYPO is the concept denoting all hypothesis spaces. The symbol REIF denotes a concept and BINARY-REL, DOMAIN, RANGE and HYPO-REL denote its associated roles. With these conventions, we are able to define the (bijective) *reification* function $\Re(t.term_h) = r.term$, where $t.term_h$ is a terminological term known to be true in hypothesis space h (i.e., h : HYPO) and $r.term$ is its corresponding reified term (a fragmentary definition is given in Table 6). By analogy, we may also define the inverse function $\Re^{-1}$. Note that the anchoring term introduced by the reification, the so-called *reificator* r, can be determined by the function π (e.g., $\pi(\Re(a\ R\ b)) = r$).

Multiple Contexts. The use of contexts and corresponding translation rules in our reasoning system primarily builds on the work by McCarthy (1993) and Buvac et al. (1995) Applying their formal results to a terminological reasoning framework, a valid notion of mutual, truth-preserving translatability between different reasoning contexts can be defined. A detailed description of the

QUALIFIED	$\doteq$	REIF $\times$ QUALITY-LABEL
QUALITY-LABEL	$\sqsubseteq$	THING

Table 7: Role and Concept Definitions for Qualifications

application of the formalism for the translation between multiple contexts is given by Schnattinger et al. (1995). For brevity, we here just state a single second-order translation rule schema (called $\mathcal{TRANS}$) such that one context (the *initial* context) be completely translatable to another one (the *meta*context):

$$\forall R : R \in \mathbf{P}^{ext} \rightarrow (\forall d, r \ \exists p, q : \mathbf{ist}(initial,\ d\ R\ r) \leftrightarrow \mathbf{ist}(meta,\ \pi(\Re(d\ R\ r)) = p \sqcap p\ \text{QUALIFIED}\ q))$$

The *initial context* contains the original terminological knowledge base and the text knowledge base reflecting the knowledge acquired from the underlying text by the text parser. Knowledge in the initial context is represented without any explicit qualifications, attachments, provisos, etc. The *metacontext*, on the other hand, consists of the reified knowledge of the initial context. In addition, qualifications can be expressed by instantiating the special role QUALIFIED with respect to some reificator (for a definition, cf. Table 7).

In a similar way, we may construct a translation scheme according to which the metacontext is (re)translatable to the initial context. This scheme incorporates the quality of hypotheses, which must exceed a specific THRESHOLD (more details of this selection process are discussed in Section 5, Table 14).

3 QUALITY LABELS

The set of quality labels we here discuss is taken from our natural language text understanding and concept learning application (though any other application seems equally reasonable). We distinguish two different sources of evidences, viz. linguistic and conceptual ones. *Linguistic* quality labels convey information about the language-specific provenance of the concept hypotheses and their individual strength. This idea is based on the observation that syntactic constructions differ in their potential to limit the degrees of freedom for conceptually interpreting an unknown lexical item and thus to constrain the range of plausible inferences that can be drawn to properly locate it in the domain's concept hierarchy. For example, an apposition like "the operating system OS/2" doubtlessly determines the superclass of "OS/2" (here considered as the unknown item or so-called *target concept* of the learner) to be "operating system", while case frame assignments for the main verb as in "OS/2 is supplied by IBM" are guided by far less selective sortal constraints and, in the example, at best allow to infer "OS/2" to be one of the products of IBM (e.g., a computer or a piece of software). From a quality point of view, appositions are a strong linguistic device, while case frame assignments are a much

APPOSITION	$\sqsubseteq$	QUALITY-LABEL
CASE-FRAME	$\sqsubseteq$	QUALITY-LABEL

Table 8: Linguistic Quality Labels

weaker source of evidence for constraining a target concept's meaning by structural linguistic clues. We may then stipulate that concept hypotheses derived from appositions are more reliable (more certain) than those derived from case frame assignments only, independent of the conceptual properties they are else associated with. Note that depending on the type of syntactic construction encountered and the conceptual constraints associated with the lexical items involved, alternative hypotheses are formed. In addition, the reasons for their emergence (here, their syntactic construction type) are encoded (after reification) in the corresponding QUALIFIED role by means of *syntactic* qualification rules. Dependent on such a role instantiation the classifier may deduce corresponding quality labels such as APPOSITION or CASE-FRAME (as defined in Table 8).

Similarly, the emerging conceptual representations of the content of the text and associated hypotheses at the level of the text knowledge base can be evaluated in terms of qualifying statements. We here introduce four types of quality labels that can be derived from *conceptual* qualification rules (for more details, cf. Hahn et al. (1996)):

i: The *very positive* quality label M-DEDUCED is generated whenever the same role filler has been multiply derived in different hypothesis spaces.

ii: The conceptual proximity of role fillers of a (non-ACTION) concept, which share a common concept class leads to the *positive* quality label SUPPORT.

iii: The inherent symmetry between two instances mutually related via two quasi-inverse relations (figuring as "inverted" role fillers of each other) is expressed by the *positive* quality label C-SUPPORT.

iv: The negative assessment for any attempt to fill the same mandatory case role of an ACTION concept more than once by different role fillers is expressed by the *negative* quality label ADD-FILLER.

These conceptual quality labels are formally defined in Table 9 (note that their generation is already grounded in the availability of certain role fillers of qualifying roles and hence allow for a "richer" definition than the linguistic ones):

C-SUPPORT	$\doteq$	QUALITY-LABEL $\sqcap \forall$C-SUPPORT-BY.REIF			
SUPPORT	$\doteq$	QUALITY-LABEL $\sqcap \forall$SUPPORT-BY.REIF			
ADD-FILLER	$\doteq$	QUALITY-LABEL $\sqcap \forall$ADD-FILLED-BY.REIF			
M-DEDUCED	$\doteq$	QUALITY-LABEL $\sqcap \forall$M-DEDUCED-BY.REIF			
C-SUPPORT-BY	$\sqsubseteq$	ROLES	SUPPORT-BY	$\sqsubseteq$	ROLES
ADD-FILLED-BY	$\sqsubseteq$	ROLES	M-DEDUCED-BY	$\sqsubseteq$	ROLES

Table 9: Conceptual Quality Labels

As with syntactic qualifications, a conceptual qualification relates a reificator via the role QUALIFIED to a *qualifying instance* which itself is related via a special qual-

Result of the reification mechanism after the first two words:		
$r_0 = \pi(\Re(sell\text{-}1 : \text{SELL}))$		1
$r_5 = \pi(\Re(sell\text{-}1 \text{ AGENT } IBM))$		
r_0 HYPO-REL $hypo_0$ $\sqcap$ r_5 HYPO-REL $hypo_0$		

Result of the reification mechanism for the entire sentence:		
for $hypo_1$:	for $hypo_2$:	
$r_1 = \pi(\Re(x : \text{PRODUCT}))$	$r_6 = \pi(\Re(x : \text{PRODUCER}))$	2
$r_2 = \pi(\Re(x : \text{OBJECT}))$	$r_7 = \pi(\Re(x : \text{OBJECT}))$	
$r_3 = \pi(\Re(x : \text{TOP}))$	$r_8 = \pi(\Re(x : \text{TOP}))$	
$r_4 = \pi(\Re(sell\text{-}1 \text{ PATIENT } x))$	$r_9 = \pi(\Re(sell\text{-}1 \text{ AGENT } x))$	
r_0 HYPO-REL $hypo_1$	r_0 HYPO-REL $hypo_2$	
r_5 HYPO-REL $hypo_1$	r_5 HYPO-REL $hypo_2$	
r_1 HYPO-REL $hypo_1$	r_6 HYPO-REL $hypo_2$	
r_2 HYPO-REL $hypo_1$	r_7 HYPO-REL $hypo_2$	
r_3 HYPO-REL $hypo_1$	r_8 HYPO-REL $hypo_2$	
r_4 HYPO-REL $hypo_1$	r_9 HYPO-REL $hypo_2$	

Translation rules yield:		
r_0 QUALIFIED q_0 $\sqcap$ q_0 : QUALITY-LABEL		3
r_5 QUALIFIED q_5 $\sqcap$ q_5 : QUALITY-LABEL		
r_1 QUALIFIED q_1	r_6 QUALIFIED q_6	
q_1 : QUALITY-LABEL	q_6 : QUALITY-LABEL	
r_2 QUALIFIED q_2	r_7 QUALIFIED q_7	
q_2 : QUALITY-LABEL	q_7 : QUALITY-LABEL	
r_3 QUALIFIED q_3	r_8 QUALIFIED q_8	
q_3 : QUALITY-LABEL	q_8 : QUALITY-LABEL	
r_4 QUALIFIED q_4	r_9 QUALIFIED q_9	
q_4 : QUALITY-LABEL	q_9 : QUALITY-LABEL	

Table 10: A Sample Qualification Process

ifying role (e.g., M-DEDUCED-BY) to another reificator. Such *qualifying assertions* are the raw data for the computation of conceptual quality labels by the classifier (e.g., M-DEDUCED). An example of the working of these principles is supplied in the next section (cf. Table 10 and 11).

4 CONCEPT ACQUISITION SCENARIO

Consider the phrase "IBM sells OS/2" where "OS/2" is unknown (in Table 10 "OS/2" is indicated by x). Major steps of the qualification process are illustrated in Tables 10 and 11. The parser analyzes this phrase in an incremental way. We start with the analysis of the first part of this phrase ("IBM sells"). This leads to the creation of an instance of a special SELL action (viz. *sell*-1) and its role AGENT being filled by IBM in hypothesis space $hypo_0$ (cf. phase 1 in Table 10).

The subsequent analysis of the remainder of the sentence and the associated operations of the hypothesis generation rules (cf. phase 2 in Table 10) lead to two conceptual ambiguities, viz. x being the PATIENT filler of *sell*-1 in one alternative, and x being the (second) AGENT filler of the action *sell*-1 in the other. These ambiguities are represented in the initial context in terms of two hypothesis spaces specializing $hypo_0$ — $hypo_1$ holds the conceptual representation of the patient reading, whereas $hypo_2$ holds that of the agent reading. Additionally, we suppose

Conceptual and syntactic qualification rules yield:		
	q_5 ADD-FILLED-BY r_9	4
q_2 M-DEDUCED-BY r_7	q_7 M-DEDUCED-BY r_2	
q_3 M-DEDUCED-BY r_8	q_8 M-DEDUCED-BY r_3	
q_4 : CASE-FRAME	q_9 : CASE-FRAME	
	q_9 ADD-FILLED-BY r_5	

Classifier deductions:		
	q_5 : ADD-FILLER	5
q_2 : M-DEDUCED	q_7 : M-DEDUCED	
q_3 : M-DEDUCED	q_8 : M-DEDUCED	
	q_9 : ADD-FILLER	

Table 11: A Sample Qualification Process (cont.)

that the fillers of the AGENT and PATIENT roles of the concept SELL are sortally restricted to PRODUCER and PRODUCT, respectively, and that the concept PRODUCER as well as PRODUCT are subconcepts of OBJECT and TOP, respectively. The operation of the translation rules (cf. $\mathcal{TRANS}$ from Section 2) which map these hypotheses from the initial context to the metacontext leads to the augmentation by qualification statements as depicted by phase 3 in Table 10. On these data, syntactic and conceptual qualification rules operate and yield the extensions and refinements stated by phase 4 and phase 5 in Table 11. In short, both hypothesis spaces contain multiple deductions of x : OBJECT and x : TOP, both are derived from case frame assignments, while only in $hypo_2$ the same AGENT role is filled twice, hence the ADD-FILLED-BY qualification for r_5 and r_9. The derived qualification labels are given in phase 5 of Table 11. We will further discuss the evaluation and the selection of and hypothesis spaces in the subsequent section.

5 THE QUALIFICATION CALCULUS

Formalization of qualifying instances for hypothesis spaces. Each reificator r is related to exactly one qualifying instance q. In order to compute all qualifying instances for one hypothesis space we employ the classifier and introduce in Table 12 a new role called HYPO-QUALIFIED, the terminological composition (cf. Table 3) of the inverse of the role HYPO-REL and the role QUALIFIED (for definitions cf. Table 5 and Table 7). The fillers of HYPO-REL result from the working of the special hypothesis generation rules which operate as part of the parser and the learning component of our system.

Considering the example from Tables 10 (phases 2 and 3) and 11 (phase 4), in the hypothesis space $hypo_1$ the classifier deduces the fillers $q_0, q_1, q_2, q_3, q_4, q_5$ for the role HYPO-QUALIFIED and in $hypo_2$ the fillers $q_0, q_5, q_6, q_7, q_8, q_9$, respectively.

HYPO-QUALIFIED	$\doteq$	(HYPO-REL^{-1}, QUALIFIED) $\sqcap$ HYPO $\times$ QUALITY-LABEL

Table 12: Qualifying Instances of a Hypothesis Space

H-ADD-FILLED-BY	$\doteq$	(HYPO-QUALIFIED, ADD-FILLED-BY) $\sqcap$ HYPO $\times$ ADD-FILLER
H-C-SUPPORT-BY	$\doteq$	(HYPO-QUALIFIED, C-SUPPORT-BY) $\sqcap$ HYPO $\times$ C-SUPPORT
H-SUPPORT-BY	$\doteq$	(HYPO-QUALIFIED, SUPPORT-BY) $\sqcap$ HYPO $\times$ SUPPORT
H-M-DEDUCED-BY	$\doteq$	(HYPO-QUALIFIED, M-DEDUCED-BY) $\sqcap$ HYPO $\times$ M-DEDUCED

Table 13: Qualifying Assertions of a Hypothesis Space

Formalization of qualifying assertions for hypothesis spaces. Each qualifying instance q carries all qualifying assertions for a corresponding reificator r. With these qualifying assertions the classifier is able to deduce the corresponding quality labels for that reificator. Therefore, we introduce in Table 13 complex roles as compositions of the role HYPO-QUALIFIED and roles indicating the special types of qualification that have occurred.

With these definitions the classifier deduces for each hypothesis space those qualifying assertions which hold for their corresponding reificators. Considering our example in Table 11 (phase 4), in the hypothesis space $hypo_1$ the classifier deduces r_7 and r_8 both to be fillers of the role H-M-DEDUCED-BY. Similarly, for $hypo_2$, r_2 and r_3 are fillers of the same role and, additionally, r_5 and r_9 are fillers of the role H-ADD-FILLED-BY. Correspondingly, the classifier deduces the (very positive) quality label M-DEDUCED for $hypo_1$ and $hypo_2$, while only the latter hypothesis space also receives the (negative) quality label ADD-FILLER (cf. Table 11, phase 5).

THRESHOLD $\doteq$ HYPO $\sqcap$
$\forall$ HYPO-QUALIFIED.$\neg$INCONSISTENT-HYPO $\sqcap$
($\exists$ HYPO-QUALIFIED.APPOSITION $\sqcup$
$\text{MAX}_{\text{HYPO}}|$HYPO-QUALIFIED$|_{\text{CASE-FRAME}}$) $\sqcap$
$\text{MIN}_{\text{HYPO}}|$H-ADD-FILLED-BY

Table 14: Threshold Criterion

Formalization of the Threshold Criterion. The selection among hypothesis spaces is based on a threshold criterion (defined by the concept THRESHOLD in Table 14) which immediately rejects inconsistent hypothesis spaces. Additionally, at least one APPOSITION label or the maximum number of verb interpretations (qualified as CASE-FRAME) is required. If several of these hypothesis spaces exist, the one(s) with the least number of ADD-FILLER labels is (are) chosen. Additionally, we need to define a new terminological concept constructor intended to compute and to compare the minimal/maximal numbers of fillers of composed roles.

First, we will define the terminological concept constructor MAX[1] in Table 15, by which we may compute the instances with the maximal number of paths of a given composed role to its fillers (if any). The constructor MIN can be defined in a similar way.

[1]To compute the numbers of role fillers properly, a problem related to the correct computation of cardinalities arises for different roles which have the same fillers (called the Multiple Path Problem).

Syntax	Semantics
$\text{MAX}(R_1, .., R_n)$	$\{d \in \Delta^{\mathcal{I}} \mid \forall e \in \Delta^{\mathcal{I}}$: **CRFC** $((R_1, .., R_n)(d))$ $\geq$ **CRFC** $((R_1, .., R_n)(e))\}$
$\text{MIN}(R_1, .., R_n)$	$\{d \in \Delta^{\mathcal{I}} \mid \forall e \in \Delta^{\mathcal{I}}$: **CRFC** $((R_1, .., R_n)(d))$ $\leq$ **CRFC** $((R_1, .., R_n)(e))\}$

Table 15: Concept Constructors MAX and MIN

CRFC$(R(d))$ is a function (called *Composed Role Filler Count*) that counts all paths of the composed role R for a given instance d to the fillers of that composed role:
CRFC $((R_1, \ldots, R_i)(d)) :=$

$$\begin{cases} ||R_i(d)|| & \text{if } i = 1 \\ \textbf{CRFC}((R_1, \ldots, R_{i-1})(d)) + \sum_{e \in (R_1, \ldots, R_{i-1})(d)} (||R_i(e)|| - 1) & \text{if } i > 1 \end{cases}$$

Considering Tables 10 and 11, the hypothesis space $hypo_1$ fulfills the Threshold Criterion as defined in Table 14, whereas $hypo_2$ does not. Both, $hypo_1$ and $hypo_2$, fulfill the first condition for the concept THRESHOLD as they are not related with a qualifying instance that belongs to the concept INCONSISTENT-HYPO. Both hypothesis spaces do also not fulfill the first part of the second condition (APPOSITION), but they both have the maximum qualifying instances which belong to the concept CASE-FRAME. The final condition, however, is only satisfied by $hypo_1$ (i.e., $hypo_1 : \text{MIN}_{\text{HYPO}}|$H-ADD-FILLED-BY).

Formalization of the Ranked Prediction Criterion. The hypothesis spaces which have repeatedly fulfilled the threshold criterion in several evaluation cycles will be classified relative to three different prediction levels at the end of the analysis to select the most credible hypothesis. At the first level of predictions all hypothesis spaces are selected with the maximum number of M-DEDUCED labels. At the second level, among the remaining ones, if any, those are selected with the maximum number of SUPPORT labels. Finally, at the third level of prediction, if any, those are selected among the remaining ones with the maximum number of C-SUPPORT labels.

The ranking of prediction labels is reflected in the specialization hierarchy for different prediction types as defined in Table 16. The definition of the concept PREDICTION1 is a specialization of the concept THRESHOLD, that of PREDICTION2 is a specialization of PREDICTION1 and that of PREDICTION3 is a specialization of PREDICTION2. The classifier only tests the condition of a more specialized prediction concept, if that of the more general concept is fulfilled. The conditions of the prediction concepts are expressed using the terminological constructor MAX and the corresponding composed roles. The domain restriction for the composed roles (cf. Table 3) is crucial to restrict the domain instances of the composed role to the desired concept type.

PREDICTION1	$\doteq$	THRESHOLD $\sqcap$ MAX$_{\text{THRESHOLD}}$|H-M-DEDUCED-BY
PREDICTION2	$\doteq$	PREDICTION1 $\sqcap$ MAX$_{\text{PREDICTION1}}$|H-SUPPORT-BY
PREDICTION3	$\doteq$	PREDICTION2 $\sqcap$ MAX$_{\text{PREDICTION1}}$|H-C-SUPPORT-BY

Table 16: Prediction Levels

In the example from above, the only remaining hypothesis space that belongs to the concept THRESHOLD is $hypo_1$. It also fulfills all prediction levels — the maximum for THRESHOLD|H-M-DEDUCED-BY (being 2), and the maximum for the second and third level (all being 0). In any of these cases, the classifier deduces the membership of $hypo_1$ to the corresponding PREDICTION concepts.

6 RELATED WORK

Any qualitative model of uncertain reasoning rivals with the mainstream quantitative ones, e.g., the probabilistic models proposed by Pearl (1988), the Dempster-Shafer theory (Dempster, 1967; Shafer, 1976), certainty factors (Shortliffe & Buchanan, 1984) or fuzzy logics (Zadeh, 1989). We are not concerned with these approaches as we consider them to be too restrictive. In principle, they do not account for the reasons which lead to a particular selection from a set of alternatives. More specifically, conflicting propositions are hidden behind values, i.e., the reasons why some propositions are contradictory cannot be represented and are, therefore, not resolvable. In our approach, it is vital to reason about contradictory propositions and to consider the reasons for contradictions in order to either initialize learning processes or to select among hypothesis spaces.

Our approach bears a much closer relationship to the work of Cohen (Cohen, 1985), who aims at heuristic reasoning *about* uncertainty. The *endorsement theory* he developed is based on the representation of explicit reasons for and against propositions (beliefs, disbeliefs). Similarly, we use quality labels which represent the positive or negative support with respect to a reificator. Unlike the endorsement model, in our approach, the quality labels are strictly formalized. The work of Fox (Fox et al., 1991) shares also similarities to our work. He proposes a decision-theory-based approach which is based on classical FOL and extends it by *argumentations* (viz. inferences) to reason for and against decision options from generalized domain theories. In his proposal quantitative and qualitative uncertainty calculi are integrated. In contrast to his use of classical FOL, we use more restrictive description logics. The main advantage lies in the provision of an automatic classifier which computes the necessary selections.

7 CONCLUSION

We have introduced a qualitative model for reasoning about uncertainty which is fully embedded in a terminological reasoning framework (implemented in LOOM (MacGregor, 1994)). Considering uncertain reasoning as a choice problem between different hypotheses, the model we provide assigns quality labels to single evidences for or against a hypothesis, combines the generated labels in terms of the overall credibility of a single hypothesis, and, finally, computes a preference order for the entire set of competing hypotheses. We are currently running experiments to extend and validate the collection of quality labels, formulate empirically plausible orderings among them, and thus try to formulate a coherent paradigm of qualitative reasoning under uncertainty.

Acknowledgments. This work was supported by a grant from DFG under the account Ha 2097/2-1. We like to thank the members of our group for fruitful discussions. We also gratefully acknowledge the provision of the LOOM system from USC/ISI.

References

Buvac, S., V. Buvac & I. Mason (1995). Metamathematics of contexts. *Fundamenta Informaticae*, 23(3).

Cohen, P. (1985). *Heuristic reasoning about uncertainty: an artificial intelligence approach*. Pitman.

Dempster, A. (1967). Upper and lower probabilities induced by a multivalued mapping. *Annals of Mathematical Statistics*, 38:325–339.

Fox, J., P. Krause & M. Dohnal (1991). An extended logic language for representing belief. In R. Kruse & P. Siegel (Eds.), *Proc. European Conf. ECSQAU on "Symbolic and Quantitative Approaches to Uncertainty"*, pp. 63–69.

Hahn, U., M. Klenner & K. Schnattinger (1996). Learning from texts - a terminological metareasoning perspective. In S. Wermter, E. Riloff & G. Scheeler (Eds.), *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, pp. 453–468. Springer.

MacGregor, R.M. (1994). A description classifier for the predicate calculus. In *Proc. of the AAAI'94*, pp. 213–220.

McCarthy, J. (1993). Notes on formalizing context. In *Proc. of the IJCAI'93*, pp. 555–560.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

Schnattinger, K., U. Hahn & M. Klenner (1995). Terminological meta-reasoning by reification and multiple contexts. In C. Pinto-Ferreira & N. Mamede (Eds.), *Progress in Artificial Intelligence. EPIA'95*, pp. 1–16. Springer.

Shafer, G. (1976). *A mathematical theory of evidence*. Princeton United Press.

Shortliffe, E. & B. Buchanan (1984). A model of inexact reasoning in medicine. In B. Buchanan & E. Shortliffe (Eds.), *Rule-based Expert Systems*, pp. 233–262. Addison-Wesley.

Woods, W. & J. Schmolze (1992). The KL-ONE family. *Computers & Mathematics with Applications*, 23:133–177.

Zadeh, L. (1989). Knowledge representation in fuzzy logic. *IEEE Trans. on Knowl. & Data Engineering*, 1:89–100.

EXPERIMENTATION-DRIVEN OPERATOR LEARNING *

Kang Soo Tae and Diane J. Cook
Department of Computer Science Engineering
The University of Texas at Arlington
Arlington, TX 76019, Box 19015
{tae, cook}@cse.uta.edu

Abstract

As planning systems begin to solve more realistic problems, one of the most important and time-consuming tasks in developing a planning system is knowledge engineering (desJardins 1994). Using this assumption, an example is quickly generalized. Our learning system, WISER, learns an initial operator definition from one positive training example using incomplete and incorrect expert knowledge. To autonomously learn planning knowledge, we first introduce the *naive domain assumption* that every object of a certain type is functionally homogeneous. We present empirical results to demonstrate how much experimentation is necessary to generate an initial operator definition as the function of the degree of imperfect expert knowledge. This definition is refined by removing irrelevant literals. WISER induces the negative literals by transforming the two-valued logic system used in state description into the three-valued logic system. We generalize types into their super types, if allowed, for parameters of the learned operator. Empirical results show that WISER can learn more accurate operator definitions by doing more experiments.

Keywords: Learning By Experimentation

INTRODUCTION

A planner is an inference engine acting like a shell of an expert system. To be commercially successful in the real world, a planning system should have provably rigorous operator definitions for an application domain to be integrated with a planner shell. However, human-generated operator definitions introduce many problems (Tae & Cook 1996). To automate the acquisition of planning domain knowledge, we are building an experimentation-driven operator learning system, called WISER, running on top of PRODIGY (Carbonell *et al.* 1992).

Given incorrect and incomplete expert's knowledge, WISER learns operator preconditions using feedback from the environment. We introduce a naive domain assumption to generalize an example in order to quickly learn operator definitions. An initial operator definition is learned from one positive training example in this paper. We empirically determine how many experiments are needed to find a positive example as a function of incorrect and incomplete expert's knowledge. This definition is refined by removing irrelevant literals. WISER, then, introduces necessary negative literals, and generalizes type information for the parameters of the operator. Finally, we empirically demonstrate how the performance of WISER is improved as it refines the initial operator definition.

RELATED WORK

Much of the existing research in planning assumes that domain operators are perfect. However, since in reality those expert-provided theories are incomplete and incorrect, a fundamentally important area in planning is to autonomously learn and refine domain knowledge. Gil (Carbonell & Gil 1990; Gil 1992) explains a plan failure in terms of the incomplete domain knowledge. EXPO (Gil 1992) refines the incomplete definitions by discovering missing literals through experiments.

OBSERVER (Wang 1995), running on PRODIGY, like EXPO, autonomously learns a new operator by observing an expert's problem solving method. The initial operator definition is refined through the integrated cycles of planning, execution and plan repair. OBSERVER is applicable for both incomplete and incorrect knowledge. desJardins (desJardins 1994) uses an operator editor to build and modify incomplete and/or incorrect operators and an inductive learning system to acquire planning knowledge for missing or incorrect preconditions via feedback from simulators or users. Benson's agent (Benson 1995) adopts an inductive logic programming algorithm to learn operators from its own experience and from its observation of a domain expert. In the presence of noisy instances, the agent produces approximate preconditions of an oper-

This work is supported in part by National Science Foundation Grant IRI-9308308

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/204 ©1996 FLAIRS

ator, called a preimage of an action model, by applying a threshold.

NAIVE DOMAIN ASSUMPTION

Strips-like operators model a robot's predefined actions in terms of a set of preconditions $pre(op)$, an add-list $add(op)$, and a delete-list $del(op)$. An operator includes a list of parameters, $op(v_1, .., v_m)$, for $m \geq 0$. Each parameter, v_i, is type-constrained. The domain theory, DT, specifies the legal actions in terms of a set of operators and specifies a type hierarchy for objects. A type consists of a set of objects. Let the function var map an instance to a variable. Let two instantiated operators, op_i and op_j, be obtained from $op(v_1, .., v_n)$ by instantiating each parameter v_k of op to the same object respectively except that the r^{th} parameter is instantiated to different objects of the same type, say a for op_i and b for op_j. op_j is obtained by substituting a in op_i to b. These two instantiated operators are called r^{th} substituted-ops, represented as $subst(r, op_i/op_j, a/b)$.

Definition: Given $a, b \in ty$ and $\exists(op_i, op_j)[subst(r, op_i/op_j, a/b)]$ in a state S, let $S_i = \text{apply}(op_i, S)$ and $S_j = \text{apply}(op_j, S)$. a is functionally *homogeneous* to b with respect to op, $a \sim_{op} b$, *iff* $var(S_i) \equiv var(S_j)$ for all $S \in |S|$, the space of states. If a is homogeneous to b with respect to all operators, a is *homogeneous* to b, $a \sim b$.

For two instantiated operators, if their instantiated preconditions are both satisfied (or both not satisfied) in a state and their respective actions have the same effects on S with variables, such that $var(add(op_i)) \equiv var(add(op_j))$ and $var(del(op_i)) \equiv var(del(op_j))$, we say that the two resources (or objects) a and b are homogeneous.

Theorem: A homogeneous relation $\sim$ on the set of objects in a type has the equivalence relation.

The equivalence relation for homogeneity $\sim$ measures the equality of objects in a type ty with respect to their applicability to a set of operators. If $\sim$ is satisfied on a subset of ty, defined by $[obj_e]=\{obj_i \in ty | obj_e \sim obj_i\}$, then $[obj_e]$ forms the subtype of the type. The n equivalence relations, $\{[obj_1], .., [obj_n]\}$, partition the set of objects of a type into n subtypes. We can choose an object randomly from a subtype $[obj_e]$, and use it as a typical representative for the subtype. If one resource of a subtype is tested for all the operators, no additional resource of the subtype needs to be tested further. Any object is equivalent to a variable.

Definition: A *naive domain* is a domain where every object in a type belongs to one and only one class.

For $a, b \in ty$, $subst(r, op_i/op_j, a/b)$ implies $a \sim b$ in the naive domain. This says that there is no advantage to selecting a over b in a naive domain. Therefore, we can experiment with an operator using only one object of a given type and make it a variable without testing any more objects of the type. This assumption facilitates learning an operator definition from one example and reduces the search space drastically. The generalization or abstraction of objects and concepts is inevitable in a real domain (Knoblock 1994) and, intuitively speaking, the main problem is just to determine the most effective level of generalization for a given domain.

Interestingly, 'the most conservative constant-to-variable generalization method' employed both by Gil and by Wang is a specific case of adopting the naive domain assumption. In both of these methods, an object is first initialized to a constant. If more than one object of a type is observed, the constant is generalized into a variable. The observation of at least two objects justifies the generalization of infinitely many objects of the type into one class of homogeneous objects. For example, if there is only one robot, ROBOT1, in a domain, Robot-type is represented as a constant, and if there are two boxes, Box-type is represented as a variable. However, this method is less flexible in a multi-agent environment where ROBOT2 can be introduced later. On the other hand, our method generalizes ROBOT1 to Robot-type initially.

LEARNING INITIAL OPERATOR DEFINITION

When a robotic action is executed successfully for the first time in a state S, S constitutes a positive training example for learning preconditions; operator's preconditions are learned as a state containing a variable, $var(S)$ in OBSERVER. As Wang notes, a positive example always contains all the necessary preconditions and is sufficient for learning an initial definition. However, one issue here is that generating a positive training example itself can be expensive: either through observations from an human expert or by search through the state space using experimentation. Since the experimentation design process must evaluate many parameters such as resources and meaningful changes in the environment, minimizing the number of experiments is crucial. To avoid experimenting with an infinite number of objects, the naive domain assumption was introduced in the previous section.

If we must experiment with all the literals in a state,

the complexity of learning preconditions is exponential with respect to the number of the literals. To reduce the size of the search space, Gil uses some domain-independent heuristics to exclude irrelevant literals. Since learning biased with domain-dependent knowledge produces satisfactory results from limited data (Ourston & Mooney 1990; Mahoney & Mooney 1993; Shavlik & Towell 1989), our approach utilizes expert knowledge to avoid an exhaustive search for S. We assume that the expert knowledge is *almost correct*. WISER first heuristically calculates a probably-best state PS based on expert-supplied background knowledge and does a series of experiments on PS to find S.

To probabilistically simulate an expert's approximate knowledge AK, idealistically correct domain knowledge IK is first assumed as in (Gil 1992). If a literal l is a true precondition to an operator and l' is irrelevant to it in IK, we can assign 100% certainty to l and 0% to l' in IK. To map from IK to AK, WISER selects randomly a set of literals to inject error. In assigning certainty to each l in AK, it is assumed that if the expert's knowledge is almost correct, the probability for l is assigned a value near 100% and that for l' near 0%. If l is injected and thus assumed incorrectly as $\neg l$ by an expert, $\neg l$ is assigned a probability accordingly. As the knowledge becomes less correct and, correspondingly, the number of injected incorrect literals increases, l gets increasingly lower probability than 100% and l' gets increasingly higher probability than 0% in AK. A probabilistically biased wheel is used to simulate this mapping and randomly assigns a probability of being an operator precondition to each literal l.

To test how much experimentation is necessary in learning initial operator preconditions on average as the function of the degree (number) of injected errors, we ran WISER 5 times for the Unlock operator, containing 9 positive literals and 3 negative literals out of 34 candidate literals, from the modified Extended Strips Domain. WISER uses an oracle to generate a random initial state. Figure 1 indicates that as domain knowledge becomes more incorrect, the search for S is drastically increased. Fortunately, we can assume that an expert's knowledge is usually almost correct, and thus our experimentation-driven method is effective, while relieving the burden of providing correct knowledge from an expert.

Since initial learning is expensive, we need to selectively experiment with chosen operators. Our decision-theoretic formula based on imperfect knowledge calculates from a given state a probably-best state for experimentation based on the probability that literals

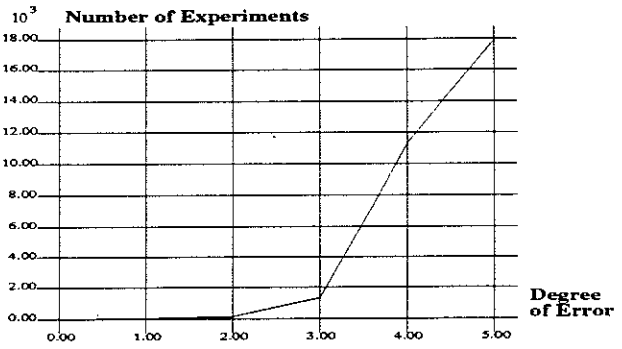


Figure 1: Number of Experiments Required for Learning an Initial Operator Definition

are preconditions for an operator, the utility of the operator, and the cost of changing the literals. Each operator is assigned a utility value, $u(op_i)$ depending on the requirements in an applied domain. Changing the value of a literal has its associated cost, $cost(l_k)$. For an operator op_j and a literal l_k, let $p(l_{kj})$ be the probability that l_k is a precondition of op_j. If $p(l_{kj})$ is above the given threshold, the literal is expected to be a precondition of the operator. For a state S_i, let the difference set $D_{ij} = \{l_1, ..., l_m\}$ be the set of the literals which are expected in the preconditions of op_j with $p(l_{kj})$ with k=1,..,m, but which are not present in the state S_i. That is, D_{ij} is the set of literals that is probably needed in a new state. $u(l_{kj})$, the utility value of l_k to op_j, is proportional to its probability $p(l_{kj})$ and the operator's utility $u(op_j)$. $u(D_{ij})$, which is the utility value of an operator op_j to the state S_i, is the sum of $u(l_{kj})$ in terms of D_{ij} minus the cost of changing the value of the literal:

$$u(D_{ij}) = \sum_{k=1}^{m} [p(l_{kj}) * u(op_j) - cost(l_{kj})]$$

This formula probabilistically finds op_j whose $u(D_{ij})$ is the highest in a state S_i. The learner requests from the oracle a new state, $(S_i - \neg D_{ij} + D_{ij})$, for a new experiment.

REFINEMENT OF OPERATOR DEFINITION

Generalization By Literal Elimination

The preconditions of an operator represent the most general description of all the states, which is Benson's idea of weakest conditions (Benson 1995). The initial definition of an operator learned from one example is over-specific. This definition should be incrementally refined by removing the irrelevant predicates. This process requires many training examples in Wang's

method. However, our insight is that one positive training example (or one observation from an expert) is sufficient in generalizing initial preconditions in a naive domain. This is active and economic learning. For the generalization of the initial preconditions, we adopted Craven and Shavlik's bottom-up search generalization method (Craven & Shavlik 1994). The preconditions for the operator are initialized as S. While the preconditions are over-specific, WISER chooses and negates each literal, $l \in S$ transiting to a new state $S' = \{S - l + \neg l\}$. We test if the operator is still applicable in S'. If applicable, we delete the irrelevant literal, because of $l \vee \neg l$, and generalize S into $\{S - l\}$. We keep on generalizing until the positive state contains only the necessary preconditions of the operator.

Learning Negative Preconditions

Knowledge acquisition is the mapping of human knowledge to a machine. Implicit knowledge is the knowledge which the expert believes the machine possesses after the mapping, but the machine actually does not possess. Implicit negative knowledge is difficult to detect and make explicit. For example, given that *(arm-empty)* is known to both a human and a machine, $\neg$*(holding x)* is known to the human, but not known to the machine. However, negative knowledge is crucial in handling complex real world problems. Carbonell and Gil (Carbonell & Gil 1990) show an incomplete domain theory in which a hidden negative constraint causes a plan failure. The failure is the inevitable result of an expert's poorly-designed domain theory. We propose a novel method to learn the necessary negative information at the initial design stage of a domain theory by unifying two types of logic systems as follows: A state description uses two-value logic: true or false. Our representation conventionally adopts the Closed-World Assumption. If p is not in the state, $\neg p$ is inferred. However, the operator description uses three-value logic: true, false, or irrelevant. If p is not in the preconditions, p is irrelevant. If p should not be in the state, $\neg p$ must appear in the preconditions. The knowledge acquisition of a learner is delimited by its language. The WISER description language is a typed first-order logic that allows conjunction and negation.

Let L represent all the predicates known to WISER and P the predicates true in S. N, the predicates not true in S, is $\{L - P\}$ by CWA. $S^* = \{P \cup \neg N\}$, providing a more comprehensive description of S, is used for inducing preconditions in WISER. WISER successfully avoids inconsistent actions in a noisy state by learning more constrained preconditions. Let S be a positive state, Since {L - P} are not true by CWA, they correspond to N. Removing CWA, S transits to S*=P+Neg({L −P}), where Neg(X) means the negated value of X. S* is identical to S and provides a more comprehensive description of the same state. S* is used for inducing preconditions in this paper. A positive state is represented as the conjunction of predicates. This raw input data constitutes the potentially overly-specific preconditions for the operators in OBSERVER. We adopt three-value logic and transform the positive state to its closure to include the negative information before initializing it to overly-specific preconditions. Let L ={A,B,C,D,E,F}, and the real preconditions Pre of an operator *op* be {A,B,C, ¬D}. In a positive state S_0={A,B,C,E}, OBSERVER initializes the preconditions as {A,B,C,E} and generalizes it to {A,B,C}. WISER initializes the preconditions to {A,B,C, ¬D,E,¬F} and generalizes it to {A,B,C, ¬D}. Given a state S_1={A,B,C,D}, Pre is not met, but *op* internally fires in Wang's method.

Most current planners using a human-generated domain theory cannot handle noise problems. Let the preconditions of *pickup* be {(arm-empty), (next-to robot box), (carriable box)} and the precondition of *put-down* {(holding box)}. Suppose an inconsistent initial state is supplied by a noisy fast-moving camera: {(holding box1), (arm-empty), (next-to robot box1), (carriable box1)}. If the goal state is {(¬ arm-empty)}, a planner generates a plan, PICKUP, and if the goal state is {(¬ holding box)}, it generates a plan, PUTDOWN. The plan execution sometimes succeeds and fails due to perceptual alias in states (Benson 1995). To make a planner robust and prevent a catastrophic result in the real world, WISER, adding negative constraints, generates more constrained preconditions of PICKUP: {(¬holding box), (arm-empty), (next-to robot box), (carriable box)}.

Type Generalization And Improved Accuracy

WISER learns the most specific type for each parameter in the definition. Thus, Pickup(Box) can not be used to pick up a key. This degrades an agent's performance significantly. WISER generalizes the specific type to its super type only if the operator definition works for all its sub types. If Object type is composed of Box type and Key type and Pickup(Key) works, then Pickup(Object) is learned. We call this a type-generalization.

To recap, WISER learns an initial concept, generalizes it by elimination, induces negative constraints, and is type-generalized. To empirically demonstrate the improved accuracy after each stage of refinement, we first generated the testing problems for Box type objects with 70% probability distribution and 30% for

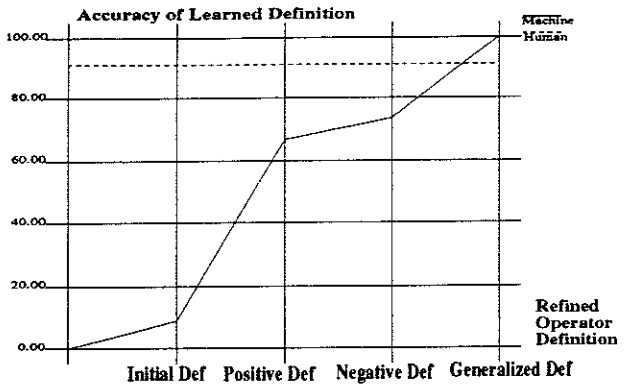


Figure 2: Accuracy of Refined Operator Definitions

Key type objects. We, then, injected noise to each problem with 10% probability[1]. We tested 100 problems 3 times. The empirical results, shown in Figure 2, demonstrate that an initially-learned operator is extremely overspecific and, after the elimination of irrelevant literals, the definition composed of only positive literals improves its performance by more than 50%. Type-generalization improves the performance by about 30%. The human-supplied definition does not have negative constraints, obtained from a PRODIGY domain. The machine-generated generalized definition outperforms the expert-provided definition since it handles noise states.

CONCLUSIONS

As planning systems need to solve more realistic problems, automated knowledge acquisition techniques are becoming increasingly crucial. Our learning system, WISER, learns an initial operator definition from one positive training example based on *naive domain assumption.* We presented empirical results demonstrating how much experimentation is necessary on average to generate an initial operator definition depending on the degree of imperfect expert knowledge. This definition is refined by removing irrelevant literals. WISER induces negative literals by transforming the two-valued logic in state description into the three-valued logic. The learned negative literals prevent inconsistent planning problems in noisy states. We generalize a type into its super type for parameters of the learned operator. Empirical results show WISER's ability to learn more accurate operator definitions by doing more experiments.

[1]Since WISER cannot handle universal quantifier to specify that if an agent holds a box it cannot hold any other box, we exclude this type of noise from testing.

References

Benson, S. 1995. Inductive learning of reactive action models. In *Proceedings of the 12th International Conference on Machine Learning.*

Carbonell, J. G., and Gil, Y. 1990. Learning by experimentation: The operator refinement method. In *Machine Learning, An Artificial Intelligence Approach,* volume 3. San Mateo, CA: Morgan Kaufmann.

Carbonell, J. G.; Blythe, J.; Etzioni, O.; Gil, Y.; Knoblock, C.; Minton, S.; Perez, A.; and Wang, X. 1992. Prodigy4.0: The manual and tutorial. Technical Report CMU-CS-92-150, Carnegie Mellon University, Pittsburgh, PA.

Craven, M. W., and Shavlik, J. W. 1994. Using sampling and queries to extract rules from trained neural networks. In *Proceedings of the Eleventh International Conference on Machine Learning.*

desJardins, M. 1994. Knowledge development methods for planning systems. In *aaai-94 Fall Symposium Siries: Planning and Learning: On to Real Applications.*

Gil, Y. 1992. *Acquiring Domain Knowledge for Planning by Experimentation.* Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, PA.

Knoblock, C. A. 1994. Automatically generating abstractions for planning. *Artificial Intelligence* 68:243–302.

Mahoney, J. J., and Mooney, R. J. 1993. Combining connectionist and symbolic learning to refine certainty-factor rule-bases. *Connection Science* 5:339–364.

Ourston, D., and Mooney, R. J. 1990. Changing the rules: A comprehensive approach to theory refinement. In *Proceedings of the Eighth National Conference on Artificial Intelligence.*

Shavlik, J. W., and Towell, G. G. 1989. Combining explanation-based and neural learning: An algorithm and empirical results. *Connection Science* 1(3):325–339.

Tae, K. S., and Cook, D. J. 1996. Knowledge acquisition for planning with incomplete domain theories. In *AAAI 1996 Spring Symposium on Planning with Incomplete Information for Robot Problems.*

Wang, X. 1995. Learning by observation and practice: An incremental approach for planning operator acquisition. In *Proceedings of the 12th International Conference on Machine Learning.*

The Iterated Version Space Algorithm

Howard J. Hamilton
Dept. of Computer Science
University of Regina
Regina, Saskatchewan, Canada, S4S 0A2
hamilton@cs.uregina.ca

Jian Zhang
Dept. of Computer Science
University of Regina
Regina, Saskatchewan, Canada, S4S 0A2
jian@cs.uregina.ca

Abstract

We present the Iterated Version Space Algorithm (IVSA), which retains many advantages of the Version Space Algorithm while handling disjunctive concepts and noise. IVSA repeatedly generates hypotheses for regions of the concept space and combines these regional hypotheses to produce an overall concept hypothesis. Experiments were conducted to learn English pronunciation rules from word-pronunciation pairs. The rules generated by IVSA produce highly accurate predictions for unseen words.

1 Introduction

We present the Iterated Version Space Algorithm (IVSA), which retains many advantages of the Version Space Algorithm (VSA) [Mit78] while handling disjunctive concepts and noise.

Learning a disjunctive concept is similar to assembling a jigsaw puzzle from a large set of possible pieces. The target concept can be viewed as the puzzle and an ordered list of disjuncts (called regional hypotheses (RH)), can be viewed as pieces of the puzzle. One method of solving this problem is to repeatedly generate candidate pieces and try adding them to the puzzle until it is complete. With IVSA, as shown in Figure 1, the Hypothesis Generator produces candidate RHs (the puzzle pieces) using VSA. Each RH covers some of input instances (part of the concept). The Hypothesis Assembler repeatedly picks the most promising RH (according to a simple ranking heuristic) and tries this RH in each position in a list of accepted RHs.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/209 ©1996 FLAIRS

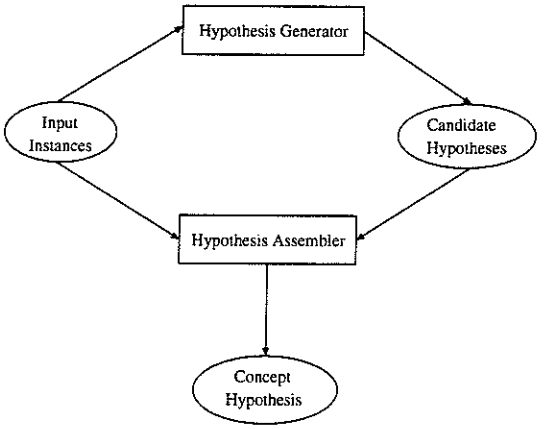


Figure 1: Overview of IVSA

If adding the RH increases the coverage of the concept, it is placed in the position that causes the greatest increase; otherwise the RH is discarded. After all candidate RHs have been examined, all accepted RHs in the list are examined to see if any can be removed without reducing accuracy. Then the process of hypothesis generation and assembly is repeated using, as input, the set of instances that are not yet covered. This process is repeated until further iterations cause no change in accuracy. The final list of accepted RHs is the concept description.

The remainder of this paper is organized as follows. In Section 2, we briefly describe version spaces and the VSA algorithm. We explain the IVSA algorithm in Section 3, and we give a descriptive example showing how IVSA discovers English pronunciation rules for graphemes in Section 4. In Section 5, we present the results of applying IVSA to learn pronunciation rules for one-syllable words. In Section 6, we discuss related research and in Section 7, we conclude.

2 Version Spaces

A *version space* [Mit82] is a representation of a concept which contains two sets: the G set, which contains the most general hypotheses consistent with all

examined instances, and the S set, which contains the most specific hypotheses consistent with all examined instances. Based on a series of positive and negative instances, the *version space algorithm* (VSA) constructs a version space. The initial general hypothesis matches everything, and the initial specific hypothesis matches only the first positive instance. If a complete and consistent set of instances of a single concept is provided, VSA converges to a single hypothesis. If the instance set is incomplete, S and G sets of hypotheses are produced. An accessible description of the VSA algorithm is given in [Win92].

VSA cannot handle *noisy* input instances [CF82], i.e., those not consistent with the existence of a single concept. Either the G set becomes overly specific or the S set becomes overly generalized. In either case, the affected set becomes empty, which indicates that no hypothesis exists that matches all instances. In practical problems, a few exceptions may be intermixed with many mutually consistent instances. In English pronunciation, the pronunciations of a grapheme in a few words contradict the majority pronunciation of the grapheme in similar contexts. Generally, a *grapheme* is a sequence of one or more letters pronounced as a single sound. For example, if the grapheme <a> is in a primarily stressed syllable which ends with a consonant, it is usually pronounced with the /æ/ sound, as in accent, access, and accident. However, in the words 'swan' and 'dwarf,' where the only syllable is primarily stressed and ends with a consonant, the grapheme <a> is pronounced with the /a/ and /open_o/ sounds, respectively. Even one such exception prevents VSA from identifying a concept.

VSA cannot discover disjunctive concepts. Continuing the example above, suppose a version space is constructed such that the G set contains only hypotheses where the conditions of being primarily stressed and ending with a consonant are present. If the next instance is <a> in accede, which is pronounced with the /æ/ sound but which is not in a primarily stressed syllable, the version space will be destroyed. This instance does not contradict the hypothesis that <a> is pronounced as /æ/ in such syllables; it contradicts the assumption that there is a single non-disjunctive hypothesis that describes when <a> is pronounced /æ/.

3 The IVSA Algorithm

In Figure 2, we present the IVSA algorithm. The main loop of IVSA is based on the overall classification accuracy of the result; if all instances are covered (i.e., correctly classified) or the accuracy does not change for *max* iterations, IVSA halts. First, us-

```
Input: I, a set of input instances
Output: H, an ordered list of regional hypotheses
I' := I, H := ⟨ ⟩, Acc_0 := 0
repeat for i = 1, 2, ...
    CH := Generator(I')
    (H, Acc_i) := Assembler(I, H, Acc_{i-1}, CH)
    I' := Incorrect(I, H)
until I' = φ or (i > max and Acc_i = Acc_{i-max})
```

Figure 2: The IVSA Algorithm

```
Input: I', a set of uncovered input instances
Output: CH, a set of candidate regional hypotheses
CH := φ
repeat for each possible decision value
    X := instances from I' classified neg or pos
    repeat until X = φ
        (S, G, x_m) := VSA(X)
        X := X − {x_1, ..., x_{m-1}}
        CH := CH ∪ S ∪ G
    end
end
```

Figure 3: The Generator Algorithm

```
Input: I, a set of input instances;
       H, the initial list of accepted RHs;
       Acc, the initial classification accuracy;
       CH, a set of candidate RHs
Output: updated H and Acc
for each candidate hypothesis h_i ∈ CH
    R_i := (|P_C| + |N_C|) / |I|
end
for h ∈ CH, ordered by decreasing R values
    BestAcc' := 0
    for j = 1, 2, ...|H| + 1
        H' := H with h inserted before position j
        Acc' := |Correct(I, H')| / |I|
        if Acc' > BestAcc' then
            BestH' := H', BestAcc' := Acc'
        end if
    end for
    if BestAcc' > Acc then
        H := BestH', Acc := BestAcc'
    end if
end for
/* Remove unnecessary hypotheses from H */
for h ∈ H
    H' := H − {h}
    Acc' := |Correct(I, H')| / |I|
    if Acc' ≥ Acc then
        H := H', Acc := Acc'
    end if
end for
```

Figure 4: The Assembler Algorithm

ing the Generator algorithm, a set CH of candidate RHs is generated from the set I' of input instances that are not yet covered. Then, using the Assembler algorithm, the candidate RHs in CH are ranked and the best ones are inserted in a list H of accepted hypotheses. Finally, the input set is updated using the function called Incorrect, which returns the set of instances from I that are not yet covered by H.

The Generator algorithm (Figure 3) handles multiple values of the decision attribute. VSA requires that instances be classified as positive and negative. For each decision value, we create a set X of appropriate instances by marking all instances with this decision value as positive and all others as negative. (In our implementation, we actually use a variant of VSA called MVSA [HZ94], which obviates the need for this translation.) Each iteration of the inner loop generates multiple candidate RHs for a single decision value. Each RH is consistent with a series of instances. If an instance x_m is encountered that (according to VSA) would destroy the version space for X, the Generator instead stores the existing S and G sets and then begins constructing a new RH, using the instances beginning with x_m as input. Thus, each RH is constructed from mutually consistent instances. Finding inconsistent instances of a concept provides a natural way of separating regions of a concept to give a disjunctive concept hypothesis.

The Assembler algorithm (Figure 4) first ranks the candidate RHs produced by the Generator. Ranking is done according to measure $R = (|P_C|+|N_C|) \ / \ |I|$, where P_c is the set of positive instances that correctly match the hypothesis, N_c is the set of negative instances that correctly do not match the hypothesis, and I is the complete set of instances. R indicates the quality of an RH.

The Assembler then considers whether each candidate RH h should be added to the list H of accepted hypotheses, as follows. First h is added to H to form H'. If the classification accuracy is higher with H' than with H, then h is kept in the accepted list H. If other hypotheses are already present in H, then h is tested in all positions in the list, and the position resulting in the highest accuracy is selected. This process is repeated for each candidate RH. Given c candidate RHs and d accepted hypotheses, the Assembler performs $O(c(c+d))$ tests. Finally, the Assembler considers whether any accepted hypotheses could now be deleted because of other hypotheses that have been inserted in the list of accepted hypotheses.

4 A Descriptive Example

We now present an application of IVSA to learning rules for pronouncing English graphemes. For efficiency, we used MVSA [HZ94] instead of VSA. The ordered list of rules for pronouncing a single grapheme is treated as a single disjunctive concept. IVSA is applied separately for each grapheme. Here, we consider the grapheme <a>, which has very complicated pronunciation rules. The input instances are drawn from the NETtalk Corpus [SR88], which is an online pronouncing dictionary, and pronunciation is specified according to the International Phonetic Alphabet (IPA) [JG81]. Both the input instances and the regional hypotheses have 10 attributes (Table 4). For a RH, '?' may appear as a value for any attribute to indicate that no restriction is placed on its value.

Attr.	Purpose	Example Values
A_1	sound unit	/æ/, /gz/, /ei/
A_2	stress on a syllable	p, s, n
A_3	type of a syllable	open, closed
A_4	second grapheme before target	any grapheme
A_5	first grapheme before target	any grapheme
A_6	first grapheme after target	any grapheme
A_7	second grapheme after target	any grapheme
A_8	first grapheme of the syllable	any grapheme
A_9	last grapheme of the syllable	any grapheme
A_{10}	# of syllables	1, 2, 3, 4

Table 1: Attributes in Input Instances

Each input instance describes one occurrence, in a word, of the grapheme <a>, including its IPA sound symbol. In the NETtalk Corpus, the grapheme <a> happens to correspond to 10 different sound symbols (10 values for the decision attribute). A positive instance contains the target grapheme <a> and the target sound symbol, while a negative instance contains the target grapheme but not the target sound symbol. We let the sound symbol /æ/ be first learning target; i.e., instances with /æ/ are positive and instances with the other 9 sound symbols are negative. Two instances from the word *abacus* are:

(ash, p, open, empty, empty, b, a, a, a, 3)
(schwa, n, open, a, b, c, u, b, a, 3)

where 'ash' represents /æ/. While learning /æ/, the first instance is treated as positive and the second as negative.

We denote the individual general hypotheses as $G_1, G_2, \ldots, G_j$, according to the order in which they are generated by VSA. Similarly, we denote the individual specific hypotheses as $S_1, S_2, \ldots, S_k$. For example, in Figure 5, the initial G set contains one hy-

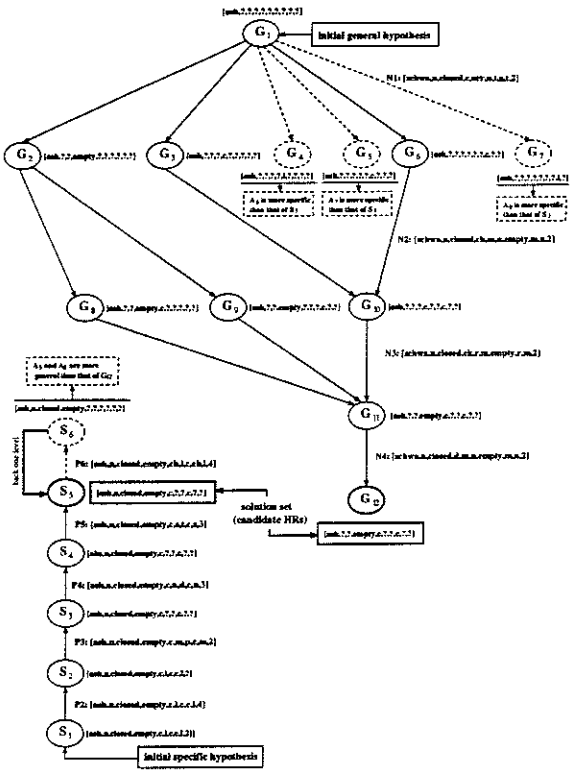


Figure 5: Hypothesis Space after One Iteration

pothesis G_1 = (ash, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?), and the initial S set contains S_1 = (ash, n, closed, au, r, l, empty, a, l, 2). Positive instances are denoted as $P_1, P_2, \ldots, P_p$, and negative ones as $N_1, N_2, \ldots, N_n$.

The Generator does not include all positive instances and exclude all negative instances in one step. Instead, an alternating sequence of positive and negative instances is examined by the MVSA algorithm. Figure 5 shows an example of one iteration of the inner loop of the Generator. When positive instance P_6 = (ash, n, closed, empty, ch, l, c, ch, l, 4) is examined, MVSA produces a new specific hypothesis S_6, (ash, n, closed, empty, ?, ?, ?, ?, ?, ?, ?). Since the current general hypothesis G_{12} is (ash, ?, ?, empty, c, ?, ?, c, ?, ?), S_6 is overly generalized in A_5 and A_8. By the definition of MVSA, S_6 is pruned, which produces an empty S set. An empty hypothesis set indicates an inconsistency in the instances and MVSA (or VSA) ordinarily would find no solution. The key idea in IVSA is that instead of failing, we take the previous S set, which here is S_5, as a solution. Hypotheses S_5 and G_{12} correctly describe all input instances before P_6. Thus, S_5 and G_{12} are candidate RHs.

In Table 2, we show selected iterations of the IVSA algorithm for this problem. During the first iteration, 54 candidate RHs are generated. When the Assembler is applied to these RHs, the first candidate RH h_1 in RH results in an accuracy of 0.88% (not shown). Next h_2 is tested both before and after h_1. Since the classification accuracy does not increase, h_2 is discarded. When all 54 candidate RHs have been examined, the classification accuracy is 70.25% with 7 selected RHs in H. After the second iteration of IVSA, the accuracy is 72.46% and so forth as shown in Table 2. When no change in the accuracy of several iterations is detected, the learning process stops after 296 iterations with 112 RHs in H and a final accuracy of 99.56%.

i	$\|CH\|$	Rejected	Accepted	$\|H\|$	Acc_i
1	54	47	7	7	70.25%
2	9	5	4	11	72.46%
3	12	8	4	15	74.23%
4	16	14	2	17	74.52%
5	17	16	1	18	76.29%
6	16	15	1	19	76.44%
7	26	23	3	22	77.76%
8	13	11	2	24	85.13%
9	7	6	1	25	85.27%
10	7	7	0	25	85.27%
50	7	7	0	37	87.04%
100	7	6	1	54	90.72%
200	7	6	1	87	95.58%
296	7	7	0	112	99.56%

Table 2: Accuracy for Grapheme <a>

Rules Learned from 90% Input Instances		
Run	Correct Words	Correct Graphemes
1	96.51%	98.86%
2	95.16%	98.74%
3	97.85%	99.43%
4	95.16%	98.71%
5	96.24%	98.85%
6	93.01%	98.12%
7	96.24%	98.99%
8	94.64%	98.50%
9	95.71%	98.83%
10	95.98%	98.87%
Average	95.65%	98.79%

Table 3: Test Results on 10% Unseen Instances

5 Results

We implemented IVSA in Prolog and C and applied it to learn pronunciation rules for all graphemes in English. In our experiments for this paper, the input was 3,724 one-syllable words (from the NETtalk Corpus), which include 17,723 grapheme examples, 98 different graphemes and 46 different sound units. Results are shown in Table 3.

IVSA was applied to 90% of the input instances (positive and negative) and tested on the unseen 10% instances (positive and negative). For the experimental run k, the unseen instances were $\{x_j \mid j \bmod 10 = k\}$, i.e., $x_1, x_{11}, x_{21}, \ldots$ for run 1. Correct IPA pronunciations were produced for 93.01% to 97.85% of the unseen words and 98.12% to 99.43% of the graphemes in the unseen words. For example, if the word *abacus* is translated as /æ, b, ei, k, ∂, s/ instead of /æ, b, ∂, k, ∂, s/, then 5 out of 6 graphemes are translated correctly, but 0 words are translated correctly. On average, 269 rules were used more than once, and an additional 665 rules (perhaps representing exceptions) were used only once.

6 Related Research

Other version-space based approaches are given in [CR94] and [VB87]. Mitchell's *extended version space* approach first sorts the instances into groups, each consistent with a single disjunct in a disjunctive hypothesis, and then applies VSA to each group individually [Mit78]. That is, if the concept *parent* is disjunctively defined as *male and has_a_child* ∨ *female and has_a_child*, the first group of instances produces the hypothesis *male and has_a_child*, while the second group of instances produces the hypothesis *female and has_a_child*. Unfortunately, no method is provided for determining in advance which attribute should be used to preclassify the instances. This makes the approach impractical for problems with many attributes.

Mitchell also suggests a method for parallel learning with version spaces [Mit78], which learns from smaller subsets of input instances on several processors and then merges the results by intersecting the version spaces. Based on this approach, Hirsh developed a version space intersection approach called *Incremental Version Space Merging* (IVSA) [Hir94]. Given two sets of specific hypotheses S_i and S_{i+1}, IVSA repeatedly selects a pair (s_j, s_k) of hypotheses with s_j from S_i and s_k from S_{i+1}, then generalizes each pair to get the intersection set: $S_{i \cap i+1} = \{s_{j \cap k}, s_{j+1 \cap k+1}, \ldots, s_{j+n \cap k+n}\}$ This intersection process is repeated for both G and S until the last G and S hypotheses which were produced from individual version spaces. The IVSA approach removes the assumption of strict consistency between a version space and its input instances. Unfortunately, this approach cannot handle cases of true inconsistency, which are misclassified according to "near by" consistent instances.

7 Conclusion

We have presented the IVSA algorithm for concept learning. Generally speaking, IVSA inherits the advantages and reduces the major weaknesses of VSA. It selects regional hypotheses that cover groups of instances and uses them to form a concept hypothesis composed of a disjunctive list of regional hypotheses. IVSA has good immunity to noise, and test results show that IVSA can obtain high accuracy if applied to unseen instances.

References

[CF82] P.R. Cohen and E.A. Feigenbaum, editors. *The Handbook of Artificial Intelligence, Volume 3.* William Kaufmann, Los Altos, CA, 1982.

[CR94] M. J. Cavaretta and R. G. Reynolds. Discovering search heuristics for concept learning using version-space guided genetic algorithms. In *Proc. of the 7th Florida Artificial Intelligence Research Symposium (FLAIRS-94)*, pages 71–75, May, 1994.

[Hir94] Haym Hirsh. Generalizing version spaces. *Machine Learning*, 17:5–46, 1994.

[HZ94] H.J. Hamilton and J. Zhang. Learning pronunciation rules for English graphemes using the Version Space Algorithm. In *Proc. of Seventh Florida Artificial Intelligence Research Symposium (FLAIRS-94)*, pages 76–80, Pensacola Beach, FL, May 1994.

[JG81] Daniel Jones and A.C. Gimson. *Everyman's English Pronouncing Dictionary.* J.M. Dent, London, 1981.

[Mit78] T.M. Mitchell. *Version Spaces: An Approach to Concept Learning.* PhD thesis, Stanford University, CA, 1978.

[Mit82] T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[SR88] T. Sejnowski and C. Rosenberg. NETtalk corpus, (am6.tar.z). ftp.cognet.ucla.edu in pub/alexis, 1988.

[VB87] K. Vanlehn and W. Ball. A version space approach to learning context-free grammars. *Machine Learning*, 2:39–74, 1987.

[Win92] P.H. Winston. *Artificial Intelligence, Third Edition.* Addison-Wesley, Reading, MA, 1992.

SHARING DOMAIN KNOWLEDGE THROUGH DERIVED CONCEPT HIERARCHIES

Colin L. Carter
Dept. of Computer Science
University of Regina, Regina,
SK, Canada, S4S 0A2
carter@cs.uregina.ca

Gary W. Hall
School of Computing Science
Simon Fraser University
Vancouver, BC, Canada, V5A 1S6
hall@cs.sfu.ca

Howard J. Hamilton
Dept. of Computer Science
University of Regina, Regina,
SK, Canada, S4S 0A2
hamilton@cs.uregina.ca

Abstract

We describe the derivation of concept hierarchies for one attribute from the values of a second attribute. These derived concept hierarchies allow the simultaneous classification of data values for a given attribute according to multiple schemes. While concept hierarchies can be derived from general world knowledge, they may also be automatically generated from a combination of previously defined hierarchies and information contained in other databases. The derived hierarchies may then be transported to multiple databases which have attributes of the same domain. This allows the exploration of data in the other databases according to information derived from the first, without having to import the tables themselves or to perform costly joins. Derived concept hierarchies can be used to interrelate data, such as census data, to a business database to profile customers for market targeting campaigns.

Keywords: concept hierarchies, attribute-oriented generalization

1. Introduction

In the exploration of databases using known data mining methods, it is often useful to have access to domain knowledge that does not exist within the data being explored. For example, a market researcher may want to develop a marketing strategy based on income and household size, but may not have such data in the company's database. One option the researcher has is to import this data from a census database into the company's database. This requires both extra storage in the database for the imported tables and extra processing time to join the table with company data.

We present an alternative method of transmitting this domain knowledge through *derived concept hierarchies.* These are concept hierarchies extracted from the information available in one database and used for data selection and generalization in other databases without changes to the schema or data of either. Derived hierarchies can be used to expand the usefulness of known data mining methods by the addition of this domain knowledge.

2. Attribute Oriented Generalization

Attribute oriented generalization (AOG) [1] is a knowledge discovery from databases technique which takes as input a table retrieved from a relational database and generalizes the data values according to a set of concept hierarchies. An example concept hierarchy is shown in Figure 1. To generalize a table retrieved from a database, data values are iteratively replaced with more general concepts until a sufficient level of generality is achieved. Duplicate tuples from the table are then combined, and various statistics are compiled about the tuples which contribute to each generalized tuple. The resulting table summarizes the original data so that the user can easily detect useful patterns.

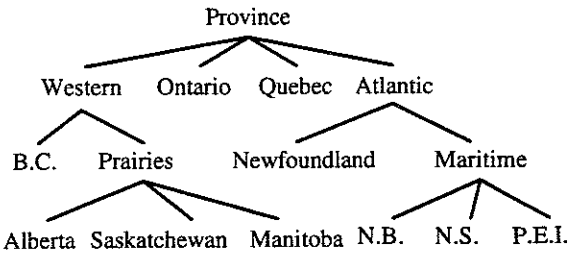


Figure 1. Concept Hierarchy for *Province* Attribute

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/214 ©1996 FLAIRS

Several fast algorithms for AOG have been developed [2,3,4,5].

3. Derived Concept Hierarchies

Concept hierarchies may be derived from data stored in a database. For example, given the 1991 census data in Table 1, Canadian provinces can be classified in terms of per capita *Average Income* or *Education Spending*. If we impose a high level structure on the *Average Income* and *Education Spending* attributes with the concept hierarchies shown in Figure 2, we can partition the provinces according to these concepts. Rather than have this information in separate table and hierarchy formats, we modify the hierarchies to include the province information by associating lists of provinces with each leaf node in the hierarchy. The derived hierarchies appear as in Figure 3.

Province	Average Income	Education Spending
Newfoundland	$15,300	$1,960
PEI	$15,300	$1,662
NS	$19,000	$1,709
NB	$18,400	$1,421
Quebec	$22,800	$1,950
Ontario	$28,400	$2,031
Manitoba	$21,700	$1,972
Saskatchewan	$20,300	$1,978
Alberta	$28,500	$1,941
BC	$25,800	$1,885

Table 1. Per Capita Income and Education Spending for Canadian Provinces

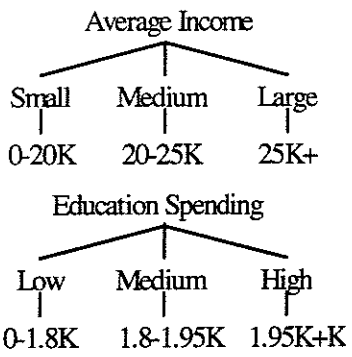


Figure 2. Concept Hierarchies to *Average Income* and *Education Spending*

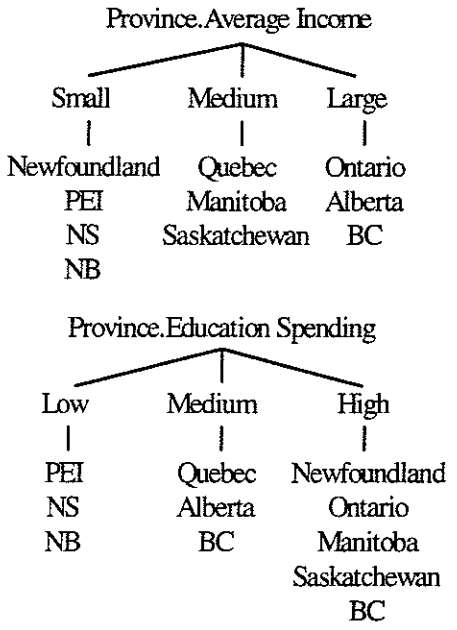


Figure 3. Expanded Hierarchies for the Province Attribute

When the number of attribute values being mapped grows large, automation is desirable. Algorithm 1 shows a simple *build_derived_hierarchy* algorithm. The result of this process is one or more hierarchies derived from one database which can be applied to any database with attributes of the same domain.

Algorithm: build_derived_hierarchy - maps the values of a primary attribute onto a concept hierarchy based on a secondary attribute.

Input: secondary_hierarchy - a hierarchy for the secondary attribute
Output: the derived hierarchy

```
procedure build_derived_attribute
        ( secondary_hierarchy : ConceptHierarchy )
begin
    var  tuple : Tuple;
         cpt : Concept;

    while ( tuple := get_next_tuple() )
        cpt = convert_secondary_value_to_concept(
                secondary_hierarchy, tuple[2] );
        append_primary_value_to_concept
                ( cpt, tuple[1] );
    end { while }
end
```

Algorithm 1. Build a Derived Hierarchy

4. Operations Using Derived Concept Hierarchies

The primary capabilities gained with derived hierarchies are more powerful high level querying and broader conceptual summarization.

Given a single geographical concept hierarchy such as in Figure 1, a database with a *Province* attribute can be queried according to any concept within the hierarchy. For example, an SQL like predicate such as:

```
where province = "Western"
```

is translated by the system to:

```
where province in ( 'B.C.', 'Alberta',
                    'Saskatchewan',
                    'Manitoba' )
```

so that the user is freed from specifying a predicate in low level terms.

Given the additional derived hierarchies in Figure 3, we may now issue a compound query containing a predicate such as:

```
where province.Region = Western and
      province.AverageIncome = high and
      province.EducationSpending = medium
```

The system retrieves the Western provinces, (B.C., Alberta, Saskatchewan, Manitoba), the high income provinces (Ontario, Alberta, B.C.), and the provinces whose education spending is in the medium range, (Quebec, Alberta, B.C.). Since the predicate is conjunctive, the intersection of these three sets is taken, resulting in a low level query of:

```
where province in (B.C., Alberta)
```

Thus, the user has the flexibility to construct complex queries according to multiple high level concepts defined on the single attribute *Province.*

Derived hierarchies may also be used to create derived attributes in data for more powerful data summarization. A *derived attribute* is an attribute dynamically added to a retrieved relation which reflects the information gained through a derived concept hierarchy. For example, given the *Education Spending* derived hierarchy for the *Province* attribute from Figure 3, the table shown in Table 2 can be automatically expanded to that in Table 3 while only retrieving two attributes from the database. Using the hierarchies from Figure 1 and Figure 4, the table can then be generalized to that shown in Table 4. The generalized result shows a predominant number of grants going to those areas which invest medium or high amounts of tax dollars in education.

Province	Grant Amount
1. Quebec	166,500
2. Quebec	15,285
3. Ontario	42,500
4. Manitoba	8,500
5. Ontario	58,000
6. B.C.	17,000
7. Alberta	15,285
8. N.S.	27,000
9. B.C.	18,700
10. Ontario	11,200
11. Newfoundland	18,700
12. Ontario	17,400
13. B.C.	15,400
14. Quebec	17,285
15. B.C.	67,000
16. Quebec	10,000
17. Ontario	20,000
18. N.S.	84,000

Table 2. Basic Research Grant Data

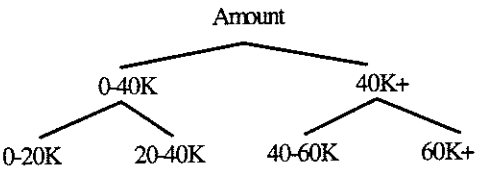


Figure 4. Hierarchy for *Amount* Attribute

5. Advantages of Derived Hierarchies

Transporting domain knowledge through derived attributes is not the only way to add domain knowledge to a database. Data such as that in Table 1 could be imported into a database, and the concept hierarchies shown in Figure 2 could be used to generalize the data. However, this means that every database in which demographic information is desired

Province	Education Spending	Grant Amount
1. Quebec	Medium	166,500
2. Quebec	Medium	15,285
3. Ontario	High	42,500
4. Manitoba	High	8,500
5. Ontario	High	58,000
6. B.C.	Medium	17,000
7. Alberta	Medium	15,285
8. N.S.	Low	27,000
9. B.C.	Medium	18,700
10. Ontario	High	11,200
11. Newfoundland	High	18,700
12. Ontario	High	17,400
13. B.C.	Medium	15,400
14. Quebec	Medium	17,285
15. B.C.	Medium	67,000
16. Quebec	Medium	10,000
17. Ontario	High	20,000
18. N.S.	Low	84,000

Table 3. Data Expanded with a Derived Attribute

Province Education Spending	Province Region	Grant Amount	Number of Grants
Low	Atlantic	0-40K	1
Low	Atlantic	40K+	1
Medium	Quebec	0-40K	3
Medium	Quebec	40K+	1
Medium	Western	0-40K	4
Medium	Western	40K+	1
High	Atlantic	0-40K	1
High	Ontario	0-40K	3
High	Ontario	40K+	2
High	Western	0-40K	1

Table 4. Generalized Relation

would have to be modified. A derived hierarchy, however, as part of the AOG system, is database independent and can be used with any database with attributes of the same domain. Furthermore, adding secondary data to several databases would cause data to be duplicated in many places. Changes to the secondary data on which the derived attributes are based would be difficult to maintain. Keeping this information in transportable hierarchies, however, would eliminate this problem.

Storing the information in derived hierarchies rather than importing secondary tables into databases as needed also avoids costly table joins. For example, suppose we have a census database which classifies various geographical areas according to enumeration area or postal / zip codes. The data will include many values which may be related to the buying habits of a company's clientele. A table of this census data for each relevant area could be imported into the company database which also has postal or zip code fields. Accessing the census data would require a join of two tables on the *Postal / Zip Code* attribute, resulting in extra time in the database query preparation and in retrieval time. High level concept hierarchies are then still needed to classify and generalize this data into high level concepts. Alternatively, derived hierarchies can be constructed from the relevant attributes of the census database. These can subsequently be used with the standard AOG knowledge discovery system without importing any data from the census database and without the need for a join across an additional table.

Since a derived attribute may be automatically generated, the secondary hierarchy on which it is based can be restructured as needed and the derived hierarchy regenerated to give a different high level view of the information. For example, suppose we have census data detailing the average income of various areas of a city. A cosmetic company may consider a medium income to be $20,000 to $40,000, while a company selling swimming pools might consider a middle income bracket to be $60,000 to $100,000. In either case the hierarchy for the secondary attribute *Income* can simply be redefined to reflect the needs of the user and the derived attributes regenerated as needed.

6. Conclusion

We have presented a method of transporting domain knowledge between databases through derived concept hierarchies. A derived hierarchy is created by mapping the values of a primary attribute onto a concept hierarchy for a secondary attribute. By this mapping, the values of the primary attribute are classified by the high level concepts of the secondary attribute. The information contained in the secondary attribute may then be transported to other databases through these derived hierarchies. This makes general domain knowledge available in an efficient form to any database with the same domain as the primary attribute. This domain knowledge is made available to standard

knowledge discovery algorithms without changing the schema of a database or modifying its data in any way.

Derived hierarchies can be used to add derived attributes to relations being retrieved from a database. These derived attributes provide additional domain knowledge from other databases, increasing the knowledge discovery potential by increasing the information content of a data query.

Derived hierarchies offer improved efficiency by avoiding the necessity of database imports of domain knowledge. They also improve query efficiency by avoiding the need of costly database joins.

References:

[1] Y. Cai, N. Cercone and J. Han, Attribute-Oriented Induction in Relational Databases, in: G. Piatetsky-Shapiro and W. J. Frawley, eds., Knowledge Discovery in Databases, AAAI/MIT Press, Menlo Park, CA, 1991, 213-228.

[2] C. L. Carter and H. J. Hamilton, A Fast, On-line Generalization Algorithm for Knowledge Discovery, *Applied Mathematics Letters,* 8:2, 1995, 5-11.

[3] C. L. Carter and H. J. Hamilton, Fast, Incremental Generalization and Regeneralization for Knowledge Discovery from Databases, in *Proceedings of the Eighth Florida Artificial Intelligence Research Symposium (FLAIRS-95),* Melbourne, FL., April, 1995, 319-323.

[4] J. Han, Towards Efficient Induction Mechanism in Database Systems, *Theoretical Computing Science,* Vol. 133, 1994, 361-385.

[5] Hwang, H.Y. and Fu, W.C., Efficient Algorithms for Attribute-Oriented Induction, in *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining (KDD-95),* Montreal, Aug., 1995, 168-173.

Defining Fuzzy User Models Using the GoM Methodology

Kathryn Gist Farinholt
School of Business
Howard University
email: farinhol@umbc8.umbc.edu

A. F. Norcio
Department of Information Systems
University of Maryland Baltimore County
email: norcio@umbc.edu

ABSTRACT

This paper presents a fuzzy representation methodology for classifying user characteristics. The Grade of Membership (GoM) methodology determines pure types for the user population. Each user is then assigned a GoM score for each pure type. It is suggested that this approach captures a clearer picture of the user's characteristics then traditional approaches.

1. INTRODUCTION

As computer systems become more complex, interfaces that adapt to the user become increasingly desirable. Rather than the user adapting to the system, an adaptive interface adapts the system to the user. This frees the user from system-specific details, (Norcio and Stanley, 1989) thus improving the efficiency of the interaction between users and computers (Hancock and Chignell 1988).

Individualizing the interaction requires the interface to have information about the user (Norcio, 1989). This knowledge allows the interface to predict the user's behavior, thus ensuring appropriate adaptation (Rich, 1983).

Techniques and strategies for predicting the user's behavior are referred to as user modeling. A user model 'is the knowledge and inference mechanism which differentiates the interaction across individuals' (Allen, 1990). Since the user model is essential to the adaptation, it is important to consider what user characteristics should comprise the user model and what inferential techniques can be used to construct the model (Chiu, Norcio, and Petrucci, 1991). This paper studies the use of a fuzzy representation for representing the user model.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/219 ©1996 FLAIRS

2. USER CLASSIFICATION

2.1 Stereotypes

User characteristics are classified using different methodologies. Stereotyping (Finin and Drager, 1986) and fuzzy representations (Chiu, Norcio, and Hsu, 1994) are two classification methodologies. A stereotype is a 'descriptive enumeration of a set of traits that often occur together' (Rich, 1983). The user is classified to one stereotype by some easily observable characteristics. These characteristics serve as a trigger for the entire stereotype (Morik and Rollinger, 1985). However, users display similar behaviors within the same stereotype; and different behaviors are displayed among the stereotypes.

However, assignment to only one stereotype is a disadvantage. A user is assigned to the stereotype that most closely compares to the user's set of traits. However, the user may only demonstrate 60% of the traits of a stereotype, yet interact with an interface developed for 100% of the traits. Further more, that user may demonstrate 40% of stereotype 2. In this case the user needs an interface that can meet 60% of the traits for stereotype 1 and 40% of the traits of stereotype 2. Using stereotypes to develop adaptive interfaces fails to handle the robustness of individual users. A fuzzy representation where users are assigned a grade of memberships for each stereotype would be preferable (Mitchell, 1992).

2.2 Fuzzy Representation (Grade of Membership)

By applying fuzzy set theory to user modeling, a user can have a grade of membership in different categories. One such fuzzy classification approach is the Grade of Membership (GoM) methodology used at Duke University's Center for Demographics (Manton, Vertrees, and Woodbury, 1990). The GoM methodology is a multivariate pattern recognition model, developed to describe medical diagnosis and symptom patterns, that applies grades of memberships to assign users to pure types (Manton et al., 1990). Data for each person is represented as a set of binary variables. Pure types are determined from the user population. The number of pure types varies, depending on the smallest number necessary to describe all nonrandom variations (Manton et al., 1990). Using the same factors for one population may elicit four pure types and another population six pure types.

Users are assigned a grade of membership for the pure types that describe their characteristics (Mitchell et al., 1992). Through the use of GoM it is possible to model the user accurately, by assigning grade of membership for the pure types. The GoM scores for each individual will be different. Each user is assigned to a dominant pure type, with other pure types listed. If there are five pure types for a specific user population, a user may demonstrate 70% of characteristics in pure type I and 20% in pure type III and 10% in pure type V. This user's dominant pure type is type I. Another user may be 100% pure type II, with pure type II the dominant pure type.

Though the assignment of a dominant pure type is similar to stereotypes the application of the grade of membership is a major difference between GoM and stereotypes. This methodology differs from stereotypes since users are not assigned to one type. It is noted that the dominant pure type does not fully describe the user's diverse characteristics. Therefore, the assignment of additional pure types captures a clearer picture of the user's diverse characteristics.

2.3 GoM Model

The model consists of four parts: raw data, lambda coefficients, grade of membership scores for each users, and number of pure types (Manton et al., 1990). The lambda coefficients predict the probability that a person who is exactly one of the pure types will have a certain response to a certain variable. For example, a person who is exactly Pure Type 2 will answer to the statement "I like my instructor(s) or supervisor(s) to recognize my efforts" with a 'strongly agree'. This coefficient is used to identify the pure types.

The Grade of Membership scores for a person determine the degree to which a particular pure type describes a person. The GoM scores may vary between .01 and 1 but the sum must equal 1. For example, a person may have a Grade of Membership score of .59 for Pure Type I. That person who has a GoM score of .59 for Pure Type I may also have a GoM score of .30 for Pure Type III and a GoM score of .11 for Pure Type IV. These scores total 1 and describe the person as having a majority of characteristics that are identified with Pure Type I but also include characteristics found in Pure Types III and IV. This differs from stereotyping where the person would be assigned to stereotype one.

The estimation of the lambda and the GoM scores is completed by maximum likelihood procedures. The chi squared statistic is generated from twice the log of the ratio of likelihood values for the models with K pure types and K+1 pure types, where K is an integer starting with two. This statistic is used to describe the input data. The model executes K pure types and then K-1 pure types until the likelihood is not increased significantly by adding another pure type. The number of pure types is different for each data set (Mitchell et al., 1992).

3. RESEARCH

3.1 Defining User Characteristics

The Productivity Environmental Preference Survey (PEPS) instrument was used to determine user's preferred preference for structuring their learning and work environment. PEPS determines the user's preference in four areas: immediate environment, emotionality, sociological needs, and physical needs (PEPS, 1992). Responses are recorded as either a strong preference for the element, a strong preference for lack of the element or no preference at all.

The environmental elements include noise, light, temperature, and design of the environment (either a formal setting of tables and chairs or an informal setting of couches or the floor). The emotionality elements include motivation, persistence, responsibility, and structure. The sociological elements include alone/peers, authority figures, or both ways. The physical elements include auditory, visual, tactile, kinethestic, time of day, and preferences for eating and drinking.

Although these are important factors to consider when structuring a learning or work environment, not all of them pertain to the development of computer interfaces. The areas of PEPS that were included in this research are: motivation, persistence, responsible, structure, alone/peers, authority figures, auditory, visual, tactile and kinesthetic. The factors of motivation, persistence and responsible would influence the amount of feedback and reminders the interface provides. A person who is not highly motivated may prefer more help comments while learning new software.

The preference for structure would influence the amount of guidance and direction the interface provides. Direct manipulation interfaces provide more directions than command line interfaces. Alone/peers and authority figures would influence the type of help comments provided. If a person wants to work with someone or someone in authority, comments can be programmed into the interface. This way the user feels that attention is being paid to their interaction. Auditory would require voice output, visual would require screen displays, and tactile/kinesthetic would require an interface that allows the user to work through tasks.

3.2 Subjects

The PEPS was administered to 98 students enrolled in an introductory computer analysis and design class to determine user characteristics. The students were enrolled at University of Maryland Baltimore County and Anne Arundel Community College. Thirty-four percent were female and sixty-six percent were male. Thirty-two percent were under the age of twenty-two, thirty-three percent were between the ages of twenty-two and thirty, nineteen percent were between the ages of thirty-one and forty, eight percent were between the ages of forty-one and fifty, and eight percent were above fifty.

3.3 Pure Types Defined

The scored PEPS results were coded and used as input for the Grade of Membership methodology program. Six pure types were defined.

Pure Type One

These users have no preferences for auditory, visual, persistence, alone or authority.

Pure Type Two

These users have a strong preference for praise, structure and authority. An interface that structures the task, leading the user through the steps and providing plenty of praise matches this pure type.

Pure Type Three

These users lack persistence and motivation, requiring visual cues and like to work with someone in authority. An interface that reminds the user what needs to be done, giving plenty of visual instructions matches this pure type.

Pure Type Four

These users are highly motivated, persistent, visual, tactile, and prefer to work alone without structure. An interface that allows the user to explore, without going through computer determined steps matches this user.

Pure Type Five

These users are auditory and kinesthetic, motivated and do not like constant checks. An interface that includes voice output, direct manipulation and lets them explore without reminders matches this user.

Pure Type Six

These users are extremely motivated, persistent and prefer to work alone, without assistance.

Pure Type Six users have a stronger preference for working along than Pure Type Four users. Likewise Pure Type Two users have a stronger preference for reminders and help than Pure Type Three. The difference lies in the strength of the preference.

3.4 GoM Scores

Each user is assigned a GoM Score for each pure type. A user may have characteristics of three pure types or only one pure type. Only fourteen percent of the users scored 100 percent of one pure type. This differs from stereotyping where one hundred percent of the users are assigned to one stereotype. Twenty-nine percent of the users received GoM scores in two pure types, twenty-nine percent received GoM scroes in three pure types, and twenty-one percent received GoM scores in four pure

types. Only six percent of the users received GoM scores for 5 pure types and no one received GoM scores in each pure type. The GoM scores indicate that users need to interact with different interfaces. Only fourteen percent of the users have all of their characteristics met with one interface.

Table 1 displays the percentage of users that have their highest and second highest GoM scores for that pure type.

Table 1
Percentage Assigned to Pure Type

Pure Types					
I	II	III	IV	V	VI
Highest					
21	10	15	23	19	12
Second					
13	15	12	17	28	15

Pure Type Four, One, and Five are the most preferred interface types. Pure Type One has no preferences for auditory, visual or authority. There is no preference for amount of structure. Pure Type Four prefers to work alone without structure and with visual displays of information. These users need an interface that allows them to explore. Pure Type Five prefers to work in groups with sound and movement, preferring no authority intervention.

The interface a user interacts with is determined by the user's GoM scores. A person with a GoM score of .59 for Pure Type One needs to interact with an interface with no structure. If that same person has a GoM score of .41 for Pure Type Four that interface should also include visual displays of information. A person with different GoM scores would require a different interface.

4. CONCLUSION

This paper presents a fuzzy approach to classifying user characteristics. The PEPS data, collected from users, was used to identify six pure types. Each user was then assigned a GoM score for each pure type. User's could be assigned a GoM score for one pure type or many pure types, depending on the user's specific characteristics. Fourteen percent of the users classified displayed one hundred percent of one pure type. The rest of the users displayed characteristics in more than one pure type. However, in stereotyping the users with characteristics in more than one pure type would be assigned to one stereotype. The GoM approach captures a clearer representation of the individual user.

REFERENCES

Allen, R.B., "User models: Theory, method, and practice," *International Journal of Man-Machine Studies* **32**, pp. 511-543 (1990).

Chiu, C., Norcio, A. F., and Hsu, C., "Reasoning on Domain Knowledge Level in Human-Computer Interaction", *Information Sciences*, **1**(1), pp. 31-46 (1994).

Chiu, C., Norcio, A. F., and Petrucci, K. E., "Using neural networks and expert systems to model users in an object-oriented environment," *Proceedings of the IEEE 1991 International Conference on Systems, Man, and Cybernetics*, pp. 1943-1948 (1991).

Finin, T. W. and Drager, D., "GUMS - A general user modeling shell," *University of Pennsylvania School of Eng. and Appl. Sci. Tech. Rep. MS-CIS-86-35*, May (1986).

Hancock, P. and Chignell, M., "Mental workload dynamics in adaptive interface design," *IEEE Transaction on Systems, Man and Cybernetics* **18**(4), pp. 647-458 (1988).

Manton, K. G., Vertrees, J. C., and Woodbury, M. A., "Functionally and medically defined subgroups of nursing home populations," *Health Care Financing Review* **12**(10), pp. 47-62 (1990).

Mitchell, K.C., "Fuzzy users: An exploratory study of the application of the grade of membership model for development of computer user types", *Masters Thesis, University of Maryland* (1992).

Mitchell, K. C., Woodbury, M. A., and Norcio, A. F., "Individualizing user interfaces: Application of the grade of membership (GoM) model for development of fuzzy user classes," *Proceedings of the First International Conference on Fuzzy Theory and Technology* (1992).

Morik, K., and Rollinger, C., "The real estate agent - Modeling users by uncertain reasoning," *The AI Magazine*, pp. 44-52, Summer (1985).

Norcio, A. F., "The Software Engineering of adaptive human-computer interfaces," *Proceedings of the 1989 IEEE International Conference on Systems, Man, and Cybernetics*, Boston, (1989).

Norcio , A. F., and Stanley, J., "Adaptive human-computer interfaces: A literature survey and Perspective," *IEEE Transactions on Systems, Man, and Cybernetics* **19**(2), pp. 399-408 (1989).

Productivity Environmental Preference Survey, 1992 Price Systems, Inc., Box 1818 Lawrence, KS 66044-1818.

Rich, E., "Users are individuals: Individualizing user models, " *International Journal of Man-Machine Studies* **18**, pp. 199-214 (1983).

Fuzzy-based Dynamic Program Reconfiguration in Distributed Systems

Nataraj Nagaratnam, Dept. of ECE, Syracuse University
Gary L. Craig, CASE Center, Syracuse University
email: { *nataraj, craig*}*@cat.syr.edu*

Abstract

In this paper, we propose a fuzzy-logic based approach for application reconfiguration in an object-oriented distributed system. In the system under consideration, each application consists of cluster of objects, grouped into ensembles. The dynamic application metrics and the system metrics together project a global state information. An efficient dynamic program configuration algorithm is required for achieving better performance, given the current state of the available resources (CPU time, memory, number of nodes etc.) and the requirements of the running application. The inherent vagueness and uncertainty in the characterization of the application and system metrics calls for the usage of linguistic variables. Hence our approach uses fuzzy rules for load characterization and performance evaluation. The load on the nodes are evaluated to assess the need to improve performance of the application and fuzzy control is employed to reconfigure the application components among the nodes. We discuss the implementation of the proposed approach in an existing prototype environment and present the obtained results.

1 INTRODUCTION

Managing resources in a object-oriented distributed environment is critical to the performance of the distributed applications executed on them. The set of resources available to any application fluctuate due to the variation in the computational and communicational load within the system. The goal of resource management is to achieve better load balancing (both communication and computation) and overall performance, based on the active application. Load-balancing policies differ from system to system, each one implementing a policy relevant to their environment. Lin and Raghavendra proposed a dynamic load balancer with central job dispatcher for distributed systems[1]. Kumar et al.[3] described a fuzzy-based expert system for load balancing. A distributed laod balancing algorithm is proposed by Park et al. [2]. Collecting the application related metrics and the systems metrics will provide a global state information about the available resources and the application performance. Decisions that are critical for improving performance are concerned about what metrics have to be collected, when and how; when should we redistribute the load among nodes and what nodes should shed load and to which of the nodes should that load be transferred to. There is inherent uncertainty in the available metric information across the application and the system. This paper characterizes uncertainty in the global state information using fuzzy set theory and presents a set of fuzzy rules to achieve efficient resource management.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/224 ©1996 FLAIRS

2 FRAMEWORK

The distributed system under consideration is the Diamonds system[4, 5], dedicated to producing a methodology and a corresponding tool-set to assist in the development of heterogenous distributed fine-grained active object-based applications. Diamonds environment consists of the static analysis tool, the run-time, central resource manager and the information repository of the system and the application. The components of the Diamonds environment executes the application code with available static information and dynamically decides run-time actions using the available system and application information. The run-time groups the objects to individual *clusters* and clusters into *ensembles*. To achieve better application performance and system throughput, the run-time and resource manager co-operate in mapping these ensembles to physical nodes as illustrated in Figure 1. Run-time reconfigures the application whenever a load imbalance is detected in the system. Reconfiguration might effectively involve migration of cluster(s) from one ensemble to another. For both initial ensemble placement and reconfiguration, the global state information is required. Fuzzy rules are used to detect the imbalance in the system and to select least 'loaded' node among them for load re-distribution, effectively achieving efficient resource management.

The characterization of the values of the system parameters and the AnT (Anticipated Throughput, an indicator of the node's capacity to process tasks at a given time instant) of the nodes as *High*, *Low* and *Medium* is both natural and appropriate. In this way, a network of nodes can be coarsely partitioned to analyze the overall load across the systems and to use them for load balancing. These three characterization will form the basic primitive classifications of a given parameter-value tuple and their membership among these sets. Further finer classifications satisfying the need of a particular application and environment (*very* High, *some-*

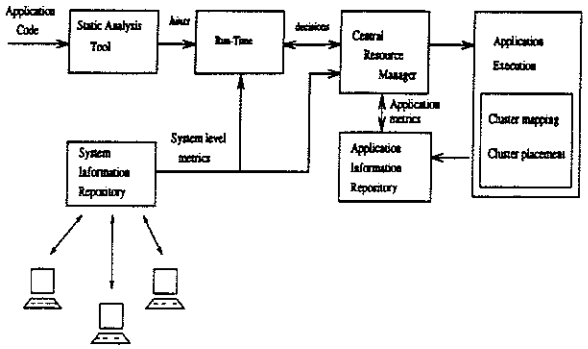


Figure 1: Diamonds Framework

what Medium etc.) are plausible by minor modifications to the obtained membership values. So in our prototype, the applied set of rules use fuzzy linguistic variables.

2.1 Static Analysis

The runtime creates ensembles for the program to execute. An ensemble manages clusters of objects. It is responsible for the creation of clusters and handling their messages. The number of ensembles created for an application is a measure of inherent parallelism in the program. It also depends on the availability of physical nodes for the program to run. Once the decision about the number of ensembles to create initially is made, the resource manager (RM) has to make a decision on where to place the ensembles, ie. it is responsible for mapping the ensembles to physical nodes. Initial ensemble placement is the mapping of ensemble to physical nodes at the program start-up. The number of cpus in a node, speed of the cpu, physical memory size and the load on the node at a given instant are some of the indicators of the node's throughput capacity at that time. So initial ensemble placement involves the following:

- Collection of relevant static (eg., number of cpus, speed of the cpus) and dynamic metrics (eg., cpu load) of all the participating nodes.
- Forming a set of rules using these metrics to measure the residual power of each node.
- Applying the set of rules on the participating nodes to evaluate their AnT.
- Distributing the ensembles among these nodes - the distribution ratio being proportional to their evaluated AnT.

The performance of the system for a given application depends on the efficient initial ensemble placement. Ideally there is one ensemble mapped onto one node for maximum concurrency. But in the case of multi-processor machine, it can handle more ensembles compared to a uniprocessor machine. So determining the distribution ratio depending on the number of available cpus and their state information is important.

2.2 Runtime Reconfiguration

Initial ensemble placement is accomplished during the program start-up, given the initial state of the environment. If the environment remains the same then the application can efficiently run to completion with the initial configuration. But as the dynamics of the physical node and the application affect the performance, the runtime need to keep track of the changes and reconfigure the application to suit the new environment. As objects get created in the application during runtime, they may be clustered into new clusters. Placement of these clusters into exisiting ensembles is an important placement decision as it will be reflected in the future behavior of the system. If the placement is not ideal, then objects across different clusters in different ensembles might have many messages passed between them. This in turn might result in imbalance in the system. At such times or during initial placement, the first task is to sense the need for reconfiguration. The need may arise because of overloading of certain nodes, whereas others may be less loaded. This calls for continuous collection of system and application metrics, and periodic evaluation of system state. This involves the following to be accomplished:

- Collection of application metrics (both computational metrics and communicational metrics)
- Collection of system metrics (static and dynamic metrics of the physical nodes)
- Applying a set of rules to dynamically evaluate the AnT of a node depending on the system metrics and the application metrics pertaining to the ensemble(s) mapped on to it.
- Decision concerning migration of clusters from one ensemble (mapped on a node with lesser AnT) to another ensemble (with higher AnT).

Initial placement of clusters within ensembles and mapping of ensembles to a set of nodes require static and dynamic metrics of the system. For the dynamic program reconfiguration, application metrics are also needed. A set of rules is formed using these metrics to assist the decision making during these phases. To efficiently achieve these mappings, a better understanding of the application metrics and how they affect its performance, system performance metrics and their role in the evaluation of node's AnT, and the rules that are used in these decisions, is required. As the collection of metrics and the formation and evaluation of fuzzy rules are similar for both initial ensemble placement and cluster migration, the application of fuzzy logic to initial ensemble placement is implemented and the results are presented. The results obtained by using fuzzy logic for initial ensemble placement exhibits the potential in this approach for handling the decision making problems in dynamic program reconfiguration. The following sections discuss the initial ensemble placement followed by obtained results.

3 INITIAL ENSEMBLE PLACEMENT

In this section we describe the metrics and the fuzzy rules concerning the initial ensemble placement among the available nodes. The system metrics namely, the number of cpus in a node, the speed of the cpu, the physical memory size and the load on the node at a given instant are used to evaluate a node's throughput capacity at a given time. Each node is assigned a AnT, which depends on the these metrics. Figure 2 illustrates ho w cpu load and cpu speed are represented using fuzzy logic. Similarly figure 3 illustrates the fuzzy representation of physical memory and number of cpus.

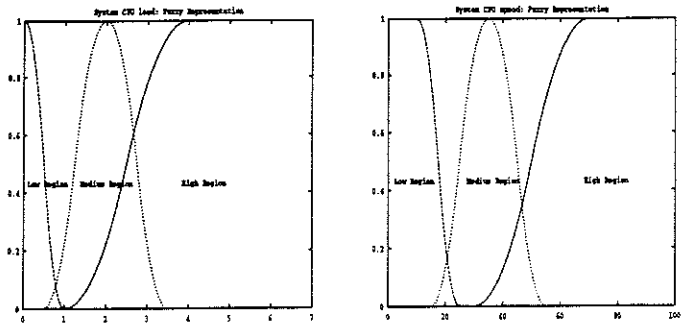


Figure 2: CPU Load and CPU Speed: Fuzzy representation

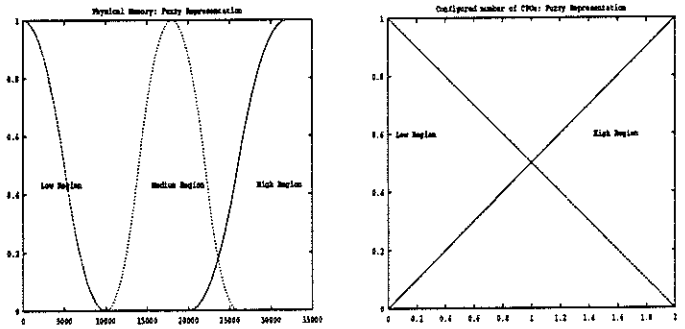


Figure 3: Physical Memory and Number of CPUs : Fuzzy representation

A node receives a low AnT if it has a high load average at that moment. The AnT is also considered low if the number of cpus is low and the cpu speed is not high. Whereas if the cpu speed is medium and the configured number of cpus is high, the node is expected to perform better. If the available memory is neither high nor low, then the node can support processes that are less memory-consuming. Hence such nodes are assigned medium AnT. A node is considered very powerful if it has higher number of cpus configured or also the cpu speed is very high. Either of these characteristics is bound to make the node perform better and have a increased throughput. Such nodes are assigned higher AnT value compared to others with less number of cpus or lower cpu speed, which receive lower AnT value. Using these criteria in mind, a set of rules are formed to evaluate the AnT of a node, using the system parameters as linguistic variables.

A set of fuzzy rules is as follows:

If *cpu_load_per_processor* is High then *AnT* is Low
If *no_of_cpus* is Low AND *cpu_speed* is not High then *AnT* is Low
If *cpu_speed* is Medium AND *no_of_cpus* is High then *AnT* is Medium
If *avail_memory* is Medium then *AnT* is Medium
If *cpu_speed* is VeryHigh OR *no_of_cpus* is VeryHigh then *AnT* is High

Applying the fuzzy rules to various machines within an environment, we can obtain different throughput capacities for those machines. The ensembles placed on a machine is proportional to the number of CPUs in that machine. After deciding and placing the ensembles, clusters are distributed among these ensembles (and hence the machines) proportional to the assigned AnT of the machine. In achieving random placement, there is one ensemble per machine and clusters are distributed randomly among these ensembles. This reflects a random placement with no knowledge of machine's static and dynamic metrics. The comparison of the completion time of an application indicates that the use of fuzzy logic has improved the performance by minimizing the completion time. The results reflect a high potential to improve system performance by evaluating machines using the fuzzy rules.

4 RESULTS

Applying the fuzzy rules to various machines under consideration, we can obtain different AnT values for those machines. The ensembles are proportionally placed in different machines depending on the number of CPUs and the clusters are distributed among ensembles depending upon their assigned AnT. The experimental network contained a mix of machines with varying CPUs, speed and memory. It helped to evaluate the assignment of AnT to all these machines and the task distribution was well balanced when fuzzy rules are used, resulting in best possible execution time. The AnT for various machines are given in Table 1.

CPUs	CPU speed in MHz	Memory in KB	CPU Load	AnT
2	60	32	1.5	8.7
1	110	8	0.7	8.65
1	50	12	0.5	4.3

Table 1: Results of Fuzzy-based System Evaluation

The performance of the two approaches were tested by solving a 8-queens problem written in the odl[6] language and executed using the Diamonds framework. The time taken by the system when the fuzzy rules are employed is compared to the time taken when random placement is achieved. The results clearly indicate that the use of fuzzy logic has improved the performance by minimizing the completion time. The comparison of results between these two approaches are presented in Table 2.

Policy	Avg. Completion Time in micro-seconds
Using the fuzzy rules	43107208
Random placement	69295331

Table 2: Comparison of the two approaches

The results reflect a 37.8% improvement in the total execution time, when fuzzy rules are used instead of random placement. The best performance while using fuzzy rules was 40203707 micro-secs and the worst was 44449944 micro-seconds. Whereas when random placement was used, the best was 59024880 micro-seconds and worst was 83359239 micro-seconds, reflecting a widely varying behavior of the random placement approach. The standard deviation for the approach using fuzzy rules is approximately 1596600 micro-seconds and that of random placement is approximately 9911000 micro-seconds. This shows a high potential to improve system performance by evaluating machines using fuzzy rules. Extension of this approach to dynamic re-configuration is presented in the next section.

5 DYNAMIC APPLICATION RECONFIGURATION

Given the Diamonds framework, it is immediately appropriate to extend these principles for dynamically reconfiguring the program. The application metrics have to be well understood to use them to take decisions on cluster migration. Considering both the computational and communicational load on an ensemble, their load at a given time instant is to be predicted and decisions have to be taken concerning the load transfer.

The loads of an ensemble can be expressed as High, Medium or Low depending on their values in the fuzzy domain. The computational load of the node (L_{phy}) and that of an application (L_{app}) determine the corresponding load on an ensemble (L_{comp}). An ensemble can be considered over-loaded if load of the underlying physical node is high. Irrespective of the application's load on that ensemble, the physical node will not be able to process more due to the restriction on its residual capacity. Whereas, if either the node has a medium load or if the application has contributed a higher load to the ensemble(s) on that node, the node can be considered to be medium loaded and can probably process more tasks. Low load on the node or low computational demand on those ensembles classify the node to be least loaded.

The inter-ensemble synchronous communication rate (L_{IEES}) and the intra-ensemble asynchronous communication rate(L_{IAEA}) are good indicators of communication load on that ensemble (L_{comm}). If either of them are high, the communicational load is considered to be on the higher side. It is considered medium if either L_{IEES} or L_{IAEA} is medium. If both communication rates are low, then the communicational load on the ensemble is also considered low.

By studying the degree of dependency of these loads on the corresponding metrics, a set of fuzzy rules are being formulated and tuned to suit the existing environment. Each rule contributes to a corresponding region in the resultant fuzzy set. We use the fuzzy min=max method as our fuzzy inference scheme to reduce the truth of a consequent fuzzy region together with the correlation minimum method in restricting the height. The defuzzification of the resulting fuzzy regions are accomplished using the method of composite moments[7]. By defuzzification of the resultant fuzzy region of the ensembles, the least to heavily loaded ensembles can be determined. Using the available information, decisions concerning cluster migration are taken and load is balanced as evenly as possible across the system, thus improving system performance. The work is under progress.

6 CONCLUSION

In this paper, we have proposed a fuzzy logic-based approach for program reconfiguration in object-oriented distributed system. The resource manager uses global state information in making decisions. The inherent uncertainity in the metrics are taken into account in forming the fuzzy rules. It is important for a load-balancing policy to adapt to the load of the system. This is taken into account in forming the decision rules. The advantages of using fuzzy set theory in resource management and dynamic reconfiguration has been experimentally verified. The performance can be further improved by by taking into account more system and application metrics while making decisions. The potential of fuzzy-based approach proved fruitful and work is in progress to incorporate this in the decision making of dynamic program reconfiguration.

7 Bibliography

References

[1] H. Lin and C. S. Raghavendra, "A Dynamic Load-Balancing Policy With a Central Job Dispatcher(LBC)", *IEEE Transactions on Software Engineering, p148-158,* Feb 1992.

[2] C. Park and J. G. Kuhl, "A Fuzzy-Based Distributed Load Balancing Algorithm for Large Distributed Systems", *Proceedings of Second Inlt. Synposium on Autonomous Decentralized Systems, p266-273.* Apr. 1995.

[3] A. Kumar, M. Singhal, and M. T Liu, "A model for distributed decision making: An expert system for load balancing", *COMPSAC, p507-513,* 1993.

[4] U. Bellur, G. Craig, K. Shank, and D. Lea, "DIAMONDS: Principles and Philosophy", *Technical Report 9313, CASE Center, Syracuse University,* June 1993.

[5] G. Craig, "Resource Management with Application Information Exchange", *Proceeding of the 4th Annual IEEE Dual-Use Technology and Applications Conference,* May 1994.

[6] D. Lea, "ODL: Language Report," *Technical Report, CASE Center, Syracuse University,*1994

[7] Earl Cox, "The fuzzy systems handbook: a practitioner's guide to building and maintaining fuzzy systems", *Boston: AP Professional,* 1994.

PROPOSITIONAL NONMONTONIC REASONING WITH EQUALITY USING THE MODAL LOGIC Z

David E. Leasure
Texas A&M University-Corpus Christi
leasure@tamucc.edu

Abstract

Circumscription has been shown to be inadequate for nonmonotonic reasoning involving circumscription of equality, even in the propositional case. We demonstrate finite domain nonmonotonic reasoning with equality using the modal logic Z. The unique names assumption is formalized in a machine solvable form for Z and an example is given. The example shows the ability of Z to nonmonotonically reason with equality, the first such formal system at least as powerful as circumscription shown to do so.

1. INTRODUCTION

Equality presents problems for nonmonotonic reasoning. The work [EMR85] demonstrates that circumscription is unable to solve problems involving the closed world assumption applied to equality. This problem exists even when the domain is finite. The problem is cited as well by [McC86] and [Lif89]. The representation of the unique names assumption (UNA) is essentially the closed world assumption applied to equality between names. In English it is stated, "different names normally denote different objects." We describe in this paper a formalization of the UNA in Z and show that Z is able to decidably solve finite UNA problems.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/228 ©1996 FLAIRS

Section 2 presents the modal logic Z and how it is used for nonmonotonic reasoning. Section 3 presents the application of Z to the UNA. Section 3.1 presents a general form of UNA. Section 3.2 presents an example of using Z to handle the UNA. Section 4 discusses conclusions and future work.

2. THE MODAL LOGIC Z

2.1 Brief Overview of Z

Z is a fragment of second order modal quantificational logic involving quantification over propositions but not over properties. It includes and extends the axioms of classical first-order logic and the modal logic S5. A widely available description of Z is [Bro91]. An extensive presentation of nonmonotonic reasoning with Z including numerous worked examples is [Lea93]. The symbols of Z are those of classical logic extended with the modal symbols shown in Table 1.

Two theorems from [Bro91] for performing possibility testing are ZP1 and ZP3. ZP1 is used for non-equality reasoning, while ZP3 is the key to reasoning modally with equality.

THEOREM ZP1: The possibility of a disjoint predicate definition

If Γ, α, and β are sentences of Z containing no unmodalized occurrence of the predicate π, =, or a propositional variable, π is of arity n, and x=x1...xn is a tuple of n variables then:

$$<>(\wedge\ \Gamma\ \forall x(\rightarrow\ \alpha\ \pi_x)\ \forall x(\rightarrow\ \beta\ \neg\pi_x))$$
iff $<>(\wedge\ \Gamma\ \neg\exists x(\wedge\ \alpha\ \beta))$

Theorem ZP1 is applicable to any theory whose prenex conjunctive normal form is such that no disjunct contains more than one unmodalized occurrence of π, since by the laws of classical logic such theories are equivalent to expressions of the form: $(\wedge\ \Gamma\ \forall x(\rightarrow\ \alpha\ \pi_x)\ \forall x(\rightarrow\ \beta\ \neg\pi_x))$ where π does not occur unmodalized in α, β, and Γ.

THEOREM ZP3: Definability of a function.

If Γ is a sentence containing no unmodalized occurrence of the function *f* or of a variable, δ is a term of Z containing no unmodalized occurrences of the function *f*, and *f* is of arity n, and x=x1...xn is a tuple of n variables then:

$$<>(\wedge\ \Gamma\ \forall x(=\ f_x\ \delta))$$
iff $<>\Gamma$

2.2 Nonmonotonic Reasoning in Z

Nonmonotonic reasoning in Z uses a consistency based approach. A default α is true if it is consistent (i.e. possible) with what is known. We may use the connector $\equiv$ to express that all and only what is known is equivalent to the propositional variable k using what we call a reflective equation: $(\equiv k\ (\Phi\ k))$, where $(\Phi\ k)$ is meta-notation for a formula of Z possibly involving k, . If $(\Phi\ k)$ does not contain k, then what is known is simply $(\Phi\ k)$. On the other hand, if $(\Phi\ k)$ does contain k (as it would if there were default expressions referring to consistency with k), then we must solve the equation (i.e. necessary equivalence) for k, that is, find its fixed points. To do this, we deduce from the initial equation a disjunction, $(v\ (\equiv k\ \beta_1)(\equiv k\ \beta_2)...(\equiv k\ \beta_n))$ where each β_i is free of k. If $(\equiv k\ \beta_i)$ implies the original equation and is free for k in β_i, then $(\equiv k\ \beta_i)$ is a solution to the original equation.

If we do not wish to allow multiple individual fixed-points representing the disjunction of solutions, we may use the propositional existential quantifier to

Table 1. Symbols of Z.

Expression	Definition	Intuitive Meaning
$([]\alpha)$	primitive	α is necessary
$(<>\ \alpha)$	$(\neg[](\neg\alpha))$	α is logically possible
$(\equiv\ \alpha\ \beta)$	$([](\leftrightarrow\ \alpha\ \beta))$	α is synonymous to β
$([\beta]\alpha)$	$([](\rightarrow\ \beta\ \alpha))$	β entails α
$(<\beta>\alpha)$	$(<>(\wedge\ \beta\ \alpha))$	α is possible(with β)
$([=]\ \delta1\ \delta2)$	$([](=\ \delta_1\ \delta_2))$	δ_1 necessarily equals δ_2

disjunctively collect the beliefs. The expression $\exists k(\wedge k\ (\equiv\ k\ \alpha))$ serves as the disjunction of the solutions to $(\equiv\ k\ \alpha)$. When we want to collect the solutions to a reflexive equation in k in this manner, we write $(\equiv j\ \exists k(\wedge\ k\ (\equiv\ k\ \alpha)))$ and the resulting solution is of the form $(\equiv j\ (\vee\ \beta_1 ... \beta_n))$, where j is all and only the facts we know.

3. NONMONOTONIC REASONING WITH EQUALITY: THE UNIQUE NAME ASSUMPTION.

3.1 Unique Name Assumption in Z

The Unique Name Assumption is the Closed World Assumption applied to equality. In other words, we may assume that unless it can be proven to the contrary, any two names represent different objects. In [*EMR85*], it is shown that UNA cannot be achieved using predicate circumscription under a traditional formalization of equality.

To represent this formally for names (constants) a and b, we write $(\equiv\ k\ (\wedge\ \Gamma\ (\rightarrow\ \langle k\rangle(\neq\ a\ b)\ (\neq\ a\ b))))$ where Γ is any other facts in the theory. If it is consistent to assume $(\neq\ a\ b)$ with what is known (i.e. there is not a proof to the contrary), then what is known contains $(\neq\ a\ b)$, written $[k](\neq\ a\ b)$. A general schematic definition of the Unique Names Assumption is

$$\begin{aligned}&UNA(\Gamma,names)\ def=\\&\exists k(\wedge k(\equiv k\ (\wedge\ \Gamma\\&\qquad \forall xy(\rightarrow\ \ (\wedge(\in x\ names)(\in y\ names))\\&\qquad\qquad (\rightarrow\ \langle k\rangle(\neq\ x\ y)\ (\neq\ x\ y)))))\end{aligned}$$

where Γ is the other knowledge known, names is the set of constants in Γ, and $(\in x\ S)$ is true if S is a finite set and x is identical to one of the elements of S, i.e. $(\in x\ S)\ def=\ (\vee\ ([=]\ x\ s_1)([=]\ x\ s_2)...([=]\ x\ s_n))$ where s_i is one of the n elements of S.

3.2 Example: Unique Names Assumption

In this example, we use a proof style which connects each step with a connector and a justification for the proof step, where the connector is one of = (equivalent), <= (implied by), and => (implies). A machine proof using the implementation described in [Lea93] was obtained in 43 seconds on a Macintosh II FX. The proof system was developed for much more general Z reasoning than is shown here.

Assumptions:

Normally, things are not equal to each other.
Ray and Reiter are the same person.
Drew and McDermott are the same person.

Conclusion:

Ray and Drew are different people.

We represent this problem using the definitions from Section 3.1. We use r as the constant for Ray, t as the constant for Reiter, d as the constant for Drew, and m as the constant for McDermott. The example is motivated by a similar problem in [Lif89].

$UNA(\ (\wedge\ (=\ r\ t)\ (=\ d\ m)),\ \{\ r\ t\ d\ m\}\ )$

= by definition expansion and minor simplification

$\exists k(\wedge\ k\ (\equiv\ k\ \Gamma))$

where Γ is

```
(∧  (= r t) (= d m)
    ∀xy(→  (∧  (∨  ([=] x r) ([=] x t)
                   ([=] x d) ([=] x m))
               (∨  ([=] x r) ([=] x t)
                   ([=] x d) ([=] x m)))
           (→ <k>(≠ x y) (≠ x y)))
```

= by Proposition 1 below.

$\exists k(\wedge\ k\ (\equiv\ k\ (\wedge\ (\neq\ r\ d)\ (=\ r\ t)\ (=\ d\ m))))$

= existential propositional quantifier simplification

$(\wedge\ (\neq\ r\ d)\ (=\ r\ t)\ (=\ d\ m))$

=>

$(\neq\ r\ d)$

qed.

Proposition 1.

```
(≡ k (∧  (= r t) (= d m)
         ∀xy(→  (∧ (∨  ([=] x r)([=] x t)
                       ([=] x d)([=] x m))
                   (∨  ([=] x r)([=] x t)
                       ([=] x d)([=] x m)))
                (→ <k>(≠ x y) (≠ x y)))))
```

iff

$(\equiv\ k\ (\wedge\ (\neq\ r\ d)\ (=\ r\ t)\ (=\ d\ m)))$

Proof omitted, but relies on Proposition 2 below.

Proposition 2.

$\langle\rangle(\wedge\ (=\ r\ t)\ (=\ d\ m)\ (\neq\ d\ r))$

Proof.

$\langle\rangle(\wedge\ (=\ r\ t)\ (=\ d\ m)\ (\neq\ d\ r))$

<= Axiom MA6 from [Bro91].

$\langle\rangle(\wedge\ (=\ r\ t)\ (=\ d\ m)\ Pd\ \neg Pr)$

= Theorem ZP3 twice

$\langle\rangle(\wedge\ Pd\ \neg Pr)$

= Theorem ZP1

$\langle\rangle\neg([=]\ d\ r)$

= Axiom A8

$\langle\rangle\#t$

=

#t (i.e. true)

qed.

4. CONCLUSIONS AND FUTURE WORK

A formalization for finite UNA is given in Z. Z is powerful enough to allow hand and machine proofs. The general approach currently taken in the proof system is exponential. To the best of our knowledge, this is the first demonstration of the solution of the unique names assumption using a formal system of nonmonotonic reasoning. Since Z is at least as strong as circumscription (as shown in [Bro91]) this result advances the capabilities of formal nonmonotonic reasoning. In general, this approach to defaults (nonmonotonic reasoning) with equality is not restricted to UNA. This opens up possibilities for handling many different kinds of equality defaults in theories of action and representation.

The machine implementation is quite general and has only weak equality reasoning. If the system were augmented with current constraint processing algorithms for equality and general possibility testing, we suspect the system would run much faster and would avoid much of the case analysis, and hence reduce the run time cost.

The biggest limitation to this approach is the restriction of equality reasoning to finite (non-quantified) domains. So far we have had limited success with hand proofs when quantified defaults are used. We continue to work in this area.

5. REFERENCES

[Bro91] Brown, Frank M., "The Modal Quantificational Logic Z Applied to the Frame Problem", *International Journal of Expert Systems Research and Applications, Special Issue: The Frame Problem. Part A.* eds. Kenneth Ford and Patrick Hayes vol. 3 number 3, pp. 169-206 JAI Press 1991.

[EMR85] D. W. Etherington, Robert E. Mercer, and R. Reiter, "On the adequacy of predicate circumscription for closed-world reasoning," *Computational Intelligence 1*, pp. 11-15.

[Lea93] David E. Leasure, *The Modal Logic Z Applied To Lifschitz's Benchmark Problems For Formal Nonmonotonic Reasoning*, Dissertation, Computer Science, The University of Kansas, April 1993.

[Lif89] Vladimir Lifschitz, "Benchmark Problems for Formal Nonmonotonic Reasoning, Version 2.00," in M. Reinfrank, J. de Kleer, M. L. Ginsberg, and E. Sandewall (Eds.), Non-Monotonic Reasoning -- Proceedings of the 2nd. International Workshop, Grassau, FRG, June 1988, published as J. Siekmann (Ed.), *Lecture Notes in Artificial Intelligence 346* (1989) Springer Verlag, pp. 202-219.

[McC86] J. McCarthy, "Applications of Circumscription to Formalizing Common-Sense Reasoning", *Artificial Intelligence*, 28. (1986).

SALO: COMBINING SIMULATED ANNEALING AND LOCAL OPTIMIZATION FOR EFFICIENT GLOBAL OPTIMIZATION

Rutvik Desai
Indiana University
rudesai@indiana.edu

Rajendra Patil
Los Alamos National Laboratory
rbp1@lanl.gov

Abstract

Simulated annealing is an established method for global optimization. Perhaps its most salient feature is the statistical promise to deliver a globally optimal solution. In this work, we propose a technique which attempts to combine the robustness of annealing in rugged terrain with the efficiency of local optimization methods in simple search spaces. On a variety of benchmark functions, the proposed method seems to clearly outperform a parallel genetic algorithm and adaptive simulated annealing, two popular and powerful optimization techniques.

1. INTRODUCTION

The goal of optimization is, given a system, to find the setting of its parameters so as to obtain the optimal performance. The performance of the system is given by an evaluation function. Optimization problems are commonly found in a wide range of fields, and it is also of central concern to many problems in artificial intelligence and machine learning. In situations where the space of parameters cannot be searched exhaustively and the evaluation function cannot be subjected to analytical methods, as is often the case in real world problems, heuristic methods have to be used. We sketch two such methods below.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/233 ©1996 FLAIRS

1.1 Simulated Annealing

Simulated annealing (SA) [1] is an optimization technique inspired from Monte Carlo methods in statistical mechanics. It attempts to avoid local optima by probabilistically taking non-locally optimal steps in the search space. The probability of taking such steps decreases with the "temperature" of the system, which in turn decreases with time. SA is able deal with evaluation functions with quite arbitrary degrees of nonlinearities, discontinuities and stochasticity and can process quite arbitrary boundary conditions and constraints imposed on these evaluation functions [1,2]. It has been shown [3] that with a "large enough" initial temperature and a proper temperature schedule, SA guarantees a globally optimal solution.

We use Adaptive Simulated Annealing (ASA) [4], an implementation of a method known as Very Fast Simulated Re-annealing (VFSR) [5] which permits a very fast temperature annealing schedule, as our basic SA algorithm.

1.2 Local Optimization

A local optimizer or a hill climber is very efficient method for optimization in simple, unimodal spaces. But in most real-world problems, it easily gets trapped in non-optimal regions, mainly because of (1) local optima; (2) flat surfaces or (3) ridges [6]. The local optimization algorithm proposed in [7] attempts to tackle some of these problems while trying to maintain the efficiency by employing following ideas: (1) Adjust the size of the probing steps to suit the nature of the terrain, shrinking when probes do poorly and growing when probes do well. (2) Keep track of the directions of recent successes, so as to probe preferentially in the direction of most rapid descent. We choose this local optimizer for combining with ASA. The outline of this algorithm for finding a locally

```
LOCAL-OPTIMIZE(f, x⃗) {
 INITIALIZE(v⃗); u⃗ ← 0;
 while |v⃗| ≥ THRESHOLD
  iter ← 0
  while f(x⃗ + v⃗) > f(x⃗) and iter < MAXITER
   v⃗ ← RANDOM-VECTOR(v⃗)
   iter ← iter + 1
  if f(x⃗ + v⃗) > f(x⃗)
   v⃗ ← v⃗ / 2
  else if iter = 0
   x⃗ ← x⃗ + v⃗; u⃗ ← u⃗ + v⃗; v⃗ ← 2u⃗;
  else if f(x⃗ + u⃗ + v⃗) < f(x⃗)
   x⃗ ← x⃗ + u⃗ + v⃗;
   u⃗ ← u⃗ + v⃗; v⃗ ← 2u⃗;
  else
   x⃗ ← x⃗ + v⃗; u⃗ ← v⃗; v⃗ ← 2v⃗;
 return x⃗;
}
```

Figure 1. An outline of the local optimization algorithm

optimal point of function f starting from point $\vec{x}$ is given in Figure 1. $\vec{v}$ is the step vector which grows and shrinks and $\vec{u}$ is used to keep track of the direction of recent success. RANDOM-VECTOR($\vec{v}$) simply returns a random vector with the same size as $\vec{v}$. MAXITER is the number of iterations before shrinking of the vector is done, and THRESHOLD is the smallest step size allowed.

2. COMBINING SIMULATED ANNEALING AND LOCAL OPTIMIZATION

The strengths and weaknesses of the two methods outlined above are complementary: SA avoids local optima by jumping away from them, but it sacrifices efficiency by doing so; hill climbers employ a "greedy" strategy and quickly reach to the nearest local optimum, but have no way of getting out of a local optimum if it is not indeed the global optimum. We attempt to combine the ability of SA to get out of local optima and the efficiency of the local optimizer in relatively "simple" regions of the space. SA helps in locating good regions of the search space, while the local optimizer is used to rapidly hit the optimum. We call this hybrid method

```
SALO(f) {
 INITIALIZE(x⃗, t);
 x⃗bsf ← x⃗;
 do
  do
   y⃗ ← PERTURB(x⃗);
   z⃗ ← LOCAL-OPTIMIZE(f, y⃗);
   Δfxy ← f(z⃗) - f(x⃗);
   if (Δfxy ≤ 0) or
     (RANDOM() < EXP(-Δfxy / T(t)))
     x⃗ ← y⃗;
     if (f(x⃗) < f(x⃗bsf)) x⃗bsf ← x⃗;
  while(equilibrium has not been reached);
  INCREMENT(t);
 while(stop criterion has not been met);
 return x⃗bsf;
}
```

Figure 2. An outline of the SALO algorithm

SALO (Simulated Annealing with Local Optimization).

From every point in the space generated by the SA process, we start the local optimizer and allow it to converge to the nearest local optimum. The value of the function at the local optimum is used as the evaluation value for the original point, and an acceptance or rejection decision is made according to the metropolis criterion. The basic high-level outline of SALO for minimization of a function $f(\vec{x})$ is given in Figure 2. $\vec{x}_{bsf}$ is the best $\vec{x}$ encountered so far, PERTURB($\vec{x}$) returns a point in the neighborhood of $\vec{x}$, and LOCAL-OPTIMIZE($f, \vec{x}$) returns a locally optimal point in the neighborhood of $\vec{x}$ for function f.

Statistical guarantee and SALO

The statistical promise of finding the globally optimal solution is considered an important feature of SA. This ensures a uniform sampling of the search space, which is reassuring when little is known about the nature of the space. Attempts to speed up SA, such as simulated quenching (SQ), usually trade this promise off with the gain in efficiency [2]. Below we argue that SALO maintains the statistical promise of SA.

Employing SALO for finding the optimal point of a function f can be viewed as using ordinary

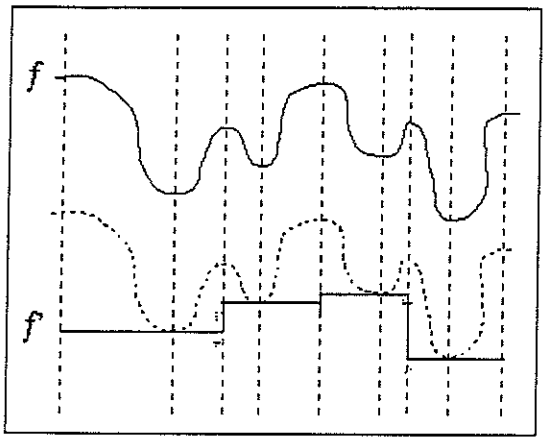


Figure 3. A sample function f and its transformation f' after applying a local optimizer

SA on a transformed function f'. The exact nature of f' depends on f and the characteristics of the local optimizer used. A transformation of a function f for a typical local optimizer is shown in Figure 3. Typical local optimizers have a "flattening" effect on the function being optimized. However, regardless of the behavior of the local optimizer (excluding some trivial cases, *e.g.*, a "local optimizer" which always returns a fixed point), f' will contain all the local optima of f, including the global optimum. In other words, using SALO on a function f is the same — including the cost and parameter cooling schedules and space sampling characteristics — as using SA on the transformed function f'. Thus SALO(f) ≡ SA(f') and if *optimum(g)* is a function returning the globally optimal point of the function g, then *optimum(f)* = *optimum(f')*. Since SA(f') is statistically guaranteed to find the global optimum of f', which is the same as that of f, we are statistically guaranteed of finding the global optimum of f. Note that SALO does not perform "true" annealing of f, but it effectively performs true annealing of f'.

If the cost of performing the transformation $f \rightarrow f'$ turns out to be less than the efficiency gained by annealing on a (hopefully) simpler surface (f'), SALO will be more efficient than SA. Otherwise, using SA will be more beneficial than using SALO. In the following section, we try to empirically determine which one of the above two cases is likely to be encountered more often in real-world situations by comparing SALO with SA and a parallel genetic algorithm, on several "interesting" functions.

3. EXPERIMENTAL RESULTS

For comparing the performance of SALO with other methods, we use several well-known benchmark problems, listed below. These problems represent various characteristic terrain found in real-world problems, *e.g.*, unimodal/multi-modal, with/without plateaus and ridges, high/low dimensional.

f1: Sphere model [8]

$$f(\vec{x}) = \sum_{i=1}^{n} x_i^2$$
$$-5.12 \le x_i \le 5.12; \min(f) = f(0,\ldots,0) = 0$$

This is a smooth, unimodal, symmetric function and does not have any problems of ridges, plateaus or foothills. The performance on this function is a measure of the general efficiency of the optimization algorithm.

f2: Rosenbrock's function [8]

$$f(\vec{x}) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)$$
$$-5.12 \le x_i \le 5.12; \min(f) = f(1,\ldots,1) = 0$$

This is a unimodal and bi-quadratic function with a very narrow ridge which runs around a parabola and has a very sharp tip. The progress of many algorithms is very slow because they are unable to discover a good search direction.

f3: Step function [8]

$$f(\vec{x}) = 6n + \sum_{i=1}^{n} \lfloor x_i \rfloor$$
$$-5.12 \le x_i \le 5.12;$$
$$\min(f) = f([-5.12,-5),\ldots,[-5.12,-5)) = 0$$

This function contains flat regions joined by steep slopes. It is a representative of plateau problems with linear and discontinuous properties. Flat regions do not give any information as to which direction is favorable.

f4: Plateau function [9]

$$f(\vec{x}) = \sum_{j=1}^{4} 2500 \cdot \max_i \left\lfloor 1000 \cdot |x_i| \right\rfloor$$
$$(j-1)h < i \le jh; h = n/4;$$
$$\min(f) = f(\max_{1 \le i \le n} |x_i| < 10^{-3}) = 0$$

This function has a large number of flat regions whose value gradually decreases towards the global minimum near the origin.

f5: Sines function

$$f(\vec{x}) = 1 + \sin^2(x_1) + \sin^2(x_2) + -0.1\exp(-x_1^2 - x_2^2)$$
$$-10 \le x_1, x_2 \le 10; \min(f) = f(0,0) = 0.9$$

This is a multi-modal function with a large number of local minima which have very similar values.

f6: Goldstein and Price

$$f(\vec{x}) = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2$$
$$-14x_2 + 6x_1x_2 + 3x_2^2)] \cdot [30 + (2x_1 - 3x_2)^2$$
$$(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$$
$$-2 \le x_1, x_2 \le 2; \min(f) = f(0,-1) = 3.0$$

This is a multi-modal and nonlinear function. The problem optimization algorithms face with this function is the peak response of about five orders of magnitude greater than the minimum in the neighborhood of the minimum.

f7: Rastrigin's function

$$f(\vec{x}) = 10n + \sum_{i=1}^{n} x_i^2 - A\cos(2\pi x_i)$$
$$-5.12 \le x_i \le 5.12; \min(f) = f(0,\ldots,0) = 0$$

This is a scaleable, multi-modal function made from the *sphere model* by modulating it with $A\cos(2\pi x_i)$ [10]. Far away from the origin this functions looks like the sphere model, but with smaller x_i the effect of the modulation grows and dominates the shape. The multi-modality here presents substantial difficulty to many optimization algorithms.

f8: Griewank 1

$$f(\vec{x}) = \frac{1}{d}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$
$$d = 2; -100 \le x_i < 100; \min(f) = f(0,\ldots,0) = 0$$

Table I shows the results in terms of the number of function evaluations required to find the global minimum with the precision of 10^{-5} for a parallel genetic algorithm (PGA), ASA and SALO. The number of function evaluations reported are the averages of 10 independent runs for each method. Lack of consistent convergence is indicated by a "?". n indicates the number of dimensions of the function.

The multi-population genetic algorithm used was PGA v2.7 [11]. The following parameters were used in PGA: number of populations 5, number of individuals in each population 20, number of bit per variable 16, selection type rank selection, mutation rate 0.005, crossover rate 1.0, crossover type two-point crossover, and migration interval 10. The same set of parameters were used in optimizing all test functions.

For ASA and SALO, a couple of parameters, which control the cost and parameter temperature annealing schedules, need to be adjusted for some functions to achieve consistent convergence. These are determined by a few trials, without fine-tuning them or exploiting the

Table I. The number of function evaluations required by various methods for finding the global optimum with the accuracy of 10^{-5}. n is the number of dimensions.

$f(\vec{x})$	n	PGA	SA	SALO
f1	2	2212	292	81
	15	17525	5925	575
f2	2	2550	3073	343
	4	197574	229390	35172
f3	5	1987	603	477
f4	2	?	488	142
	4	7935	1384	245
	8	17240	10114	2829
f5	2	2312	2968	2413
f6	2	4355	387	103
f7	2	3811	474	95
	4	?	1597	229
	8	?	7718	5199
f8	2	4762	2184	297
	10	?	28656	480

knowledge of the function terrain.

It should be noted that these results are obtained for particular implementations of the algorithms. The performance of the PGA can probably be improved marginally by experimenting with various parameters such as population size, crossover type, crossover and mutation rates, etc.

4. CONCLUSIONS AND FUTURE DIRECTIONS

Results on a variety of functions with different characteristics suggest that SALO provides a powerful optimization method. While experimentation on many real-world problems is desirable, this method clearly compares favorably to ASA and PGA, two popular optimization methods on most test functions. Once in a region near the global optimum, SALO is able to reach it rapidly due to the presence of a good local optimizer which has a variable step size and can handle ridges effectively. Annealing enables it to get out of local optima and allows it to sample different regions of the search space. From another viewpoint, the cost of "flattening" the function terrain is outweighed by the efficiency gained by doing annealing on a simpler terrain. This is achieved by what is essentially a simple addition of a function call to the local optimizer in an existing SA algorithm.

As a potential improvement to SALO, the local optimizer could be run with a probability inversely proportional to the current temperature. This could save some effort spent in initial random regions of the space. The THRESHOLD parameter in the local optimizer can also be made adaptive, keeping it high initially so as to stop the local optimizer from spending effort in trying to fine-tune its solution, and lowering it gradually so that it can reach the global optimum with the required precision, when good regions of the space are found.

References

[1] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, 220, pp. 671-680, 1983.

[2] L. Ingber, "Simulated Annealing: Practice versus Theory," *J. Mathl. Comput. Modelling*, 8, pp. 29-57, 1993.

[3] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distribution an the Bayesian Restoration in Images," *IEEE Trans. Patt. Anal. Mac. Int.*, 6, pp. 721-741, 1984.

[4] L. Ingber, *Adaptive Simulated Annealing (ASA)* [ftp.caltech.edu:/pub/ingber/asa.Z], Software package documentation, 1995.

[5] L. Ingber, "Very Fast Simulated Re-Annealing," *J. Mathl. Comput. Modelling*, 12, pp. 967-973, 1989.

[6] P. Winston, *Artificial Intelligence*, Addison-Wesley Publishing Company, Reading, MA, third edition, 1992.

[7] D. Yuret, "From Genetic Algorithms to Efficient Optimization," *MS Thesis, Dept. of Electrical Engineering and Computer Science, MIT*, 1994.

[8] K. De Jong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems," *PhD Thesis, University of Michigan, Diss. Abstr. Int. 36(10), 5140B*, University Microfilms No 76-9381, 1975.

[9] D. J. Ackley, "An Empirical Study of Bit Vector Function Optimization," in *Genetic Algorithms and Simulated Annealing*, edited by L. Davis, London, Pitman, pp. 194-200, 1987.

[10] T. Bäck, F. Hoffmeister, and H.-P. Schwefel, "Applications of Evolutionary Algorithms," *Report of the Systems Analysis Research Group SYS-2/92, University of Dortmund, Department of Computer Science*, 1992.

[11] P. Ross, "About PGA v2.7," *Dept. of AI, University of Edinburgh*, 1994.

An Environment for the Construction and Sharing of Knowledge

Alberto J. Cañas,* Kenneth M. Ford,* Patrick J. Hayes,†
John Brennan,* Thomas Reichherzer,* and Niranjan Suri*

* Institute for Human and Machine Cognition
University of West Florida, Pensacola FL 32514

† Beckman Institute
University of Illinois, Urbana IL 61801

Abstract

The purpose of this paper is to report on a continuing research effort aimed at the design and development of large-scale international computer network connecting schools in at least ten countries throughout Latin America. Toward this end, the University of West Florida and IBM Latin America have initiated a joint partnership called "Quorum: Collaboration without Boundaries." The project is large and multifaceted. In addition to providing physical connectivity, it includes teacher training and development of the curriculum material and building advanced software tools necessary to support student collaboration across classrooms and countries. This paper is focused on Project Quorum's support for collaboration at the knowledge level across classrooms and continents.

Introduction

As part of Project Quorum, we are developing a system (provisionally called Pangea) to empower students to collaborate in the construction and sharing of models of their beliefs about specific domains. Project Quorum is a three-year cooperative partnership between the University of West Florida and IBM Latin America to implement a telecommunications network connecting Latin American schools participating in Project Genesis (Cañas & Schaffer, 1990; Escorcia et al., 1991), an IBM initiative that has grown to over a thousand schools spread throughout more than ten Latin American countries. Genesis distinguishes itself by a fundamental emphasis on training, preparation, and support of teachers. Building on this strong foundation, Quorum involves the creation of an infrastructure for collaboration, including the implementation of an international computer network, development of the curriculum material necessary to support collaborative work among schools, teacher-training workshops throughout the countries, coordination in the design and development of projects, access to experts in the various areas of the curriculum, the development of advanced software tools to support more extensive collaboration as new technology becomes available in the schools, and most importantly, a momentum of interaction between schools that will result in a self-nourishing project.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/238

Collaborative learning (Ausubel, Novak & Hanesian, 1978) is an enterprise in which the learners, and perhaps their teachers, cooperatively build an explicit knowledge model which gives coherent expression to their understanding. From a constructivist perspective, the most important outcome of the modeling process is less the model itself than the insight we gain as we struggle to articulate, organize, critically evaluate and assent to it (Cañas & Ford, 1992). Likewise, the collaborative process we envision will derive its value chiefly from our success in framing the activity as a self-correcting enterprise in which learners can subject any part of the model — including their own background assumptions — to critical scrutiny. From this standpoint, the crucial question for us is "how useful is the modeling process as a means of facilitating the learner's understanding?" rather than "is the model correct?" Our research agenda, therefore, is oriented towards the development of tools and methods to aid learners and teachers to express, elaborate, share, improve, and understand their constructions of their world.

In this paper we provide a summary of the current status of Quorum and then describe the software tools being developed in Pangea to support collaborative knowledge building.

Quorum Overview: Current Status

Quorum is a means for enabling students to collaborate, to help them understand their cultural and social differences, as well as their commonalities. Students collaborate on issues that are global in nature, involving not only the problems of their communities, but also those of neighboring communities and the world as a whole. We anticipate that by their participation in the Quorum Project, students are gaining an enhanced appreciation of their social responsibilities and a deeper appreciation of the world at large.

Telecommunications initiatives between schools often limit the interaction to trivial topics that have little to do with the student's learning. In Quorum, care has been taken in developing collaborative projects that take full advantage of the participation of students from different countries and that could not be carried out without telecommunications.

In Quorum, telecommunications is incorporated into the programs that students use daily. LogoWriter and MicroMundos (MicroWorlds), the main software tools used throughout Genesis, have been extended to be both the mail tool and the transmission medium; in this way, electronic communication becomes a part of the student's work very naturally, without the need to learn a separate, dedicated

software package. We also wanted the mail tool program to be accessible to students of all ages. Quorum students and teachers are able to send not only text, but graphics, animation and sound, and more importantly, executable programs Most e-mail users in the world are limited to text messages. Very few, if any, e-mail systems allow users to execute the mail message they receive.

Quorum became operational in May of 1994 and currently includes schools from Rio de Janeiro and Sao Paulo in Brazil; Caracas and other cities throughout Venezuela; Mexico City and Aguascalientes, Mexico; and many cities throughout Costa Rica. Other countries will be joining soon. A more extensive description of the implementation can be found in Cañas (1993).

Concept Maps

Concept maps, developed in an educational setting by Novak (1977), are used as the primary language for description and communication of concepts within assimilation theory, a cognitive learning theory that has had extensive application to education (Ausubel, Novak & Hanesian, 1978). It is based on a constructivist model of human cognitive processes. In particular, assimilation theory focuses on describing how concepts are acquired and organized into a learner's cognitive structure. Ausubel argues that learning is synonymous with a change in the meaning of experience. His fundamental premise is deceptively simple:

> *Meaningful learning results when new information is acquired by deliberate effort on the part of the learner to link the new information with relevant, preexisting concepts or propositions in the learner's own cognitive structure. (Ausubel et al., 1978, p. 159)*

Assimilation theory stresses that meaningful learning requires that the learner's cognitive structure contain anchoring concepts to which new material can be related or linked. For this reason, Ausubel argues that "the most important single factor influencing learning is what the learner already knows. Ascertain this and teach him accordingly."

A concept map is a graphic display of concept names connected by directed arcs encoding propositions in the form of simplified sentences. The simplest concept map would be two nodes connected by an arc representing a simple sentence such as 'grass is green,' but they can also become quite intricate. Figure 1 shows a concept map about concept maps (from Novak & Gowin, 1984. By convention, links run top-down unless annotated with an arrowhead.) When concepts and linking words are carefully chosen, these maps can be useful classroom tools for observing nuances of meaning, helping students organize their thinking, and summarizing subjects of study. Mapping techniques are employed to help students "learn how to learn" by bringing to the surface cognitive structures and self-constructed knowledge (Novak & Gowin, 1984).

From an AI perspective, concept maps seem similar to such familiar graphical knowledge representations as semantic networks and conceptual graphs. This similarity is useful and suggestive, and we employ it in our work, but it is important to realize that concept maps are not "knowledge representations" in a computational or logical sense. They are much more loosely defined, with no firm syntactic rules and no formal rules of interpretation or semantics. Nodes in a concept map are often labelled with nouns interpretable as object names and arcs labelled with verbs interpretable as relation names, but this is by no means required. Their function is pedagogical, to help students clarify and organize their own conceptual framework, rather than formal.

There are several rules and conventions for a well-designed concept map. More general, inclusive concepts should be found at the highest levels, with progressively more specific, less inclusive concepts arranged below them. Beginners sometimes produce very "linear" maps — essentially merely decorations of long sentences — which are not well constructed. However, these conventions are not strict conditions of grammatical well-formedness, and even a poorly constructed concept map is, indeed, a concept map.

A concept map is never finished. New propositions can always be added to it. Two concept maps may also be merged to form a larger, more complex and enriched view of a subject. This process of building and merging is at the heart of the use of concept maps in Quorum.

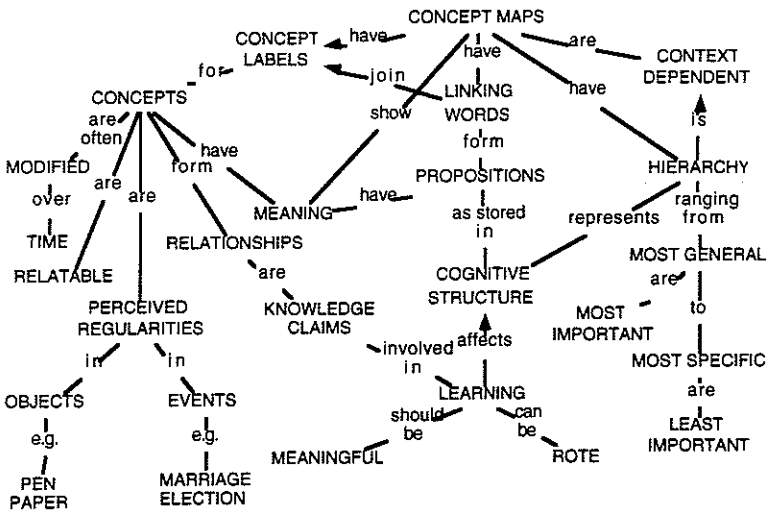


Fig. 1: A concept map about concept maps

Pangea: Collaboration at the knowledge level

Quorum affords several levels and kinds of collaboration. The simpler forms of support for collaboration include text-based mail messages and newsgroup-type discussions. Richer levels of participation include students exchanging LogoWriter pages and executable programs. For example, they can construct a simulation of the project they are studying and share that simulation with other students.

We are currently developing a collection of tools intended to take the collaboration one step further and provide a means for the students to collaborate at the "knowledge level." (Newell 1982) This system, called Pangea, allows students to build concept maps and share the knowledge expressed in their concept maps with other students.

A soup with many cooks

The system has many provisions to facilitate the process of map construction. The software allows students to build concept maps individually, under the guidance of the teacher, or as part of a group map-building project. Moreover, concept maps can be sent by electronic mail. This part of the system is fully implemented, and its use the schools will begin shortly.

Pangea will also support ways in which the knowledge encoded in concept maps can be selectively distributed among a community of pupils. This requires a rather different representation of the information in the map. As noted, a concept map can be regarded as an organized collection of propositions relating together a collection of topics. Each proposition is expressed by a simplified sentence which can be extracted from the map by following a series of arcs beginning and ending at nodes. For example, the map in Figure 1 contains the sentences 'CONCEPTS have MEANING' and 'RELATIONSHIPS are KNOWLEDGE CLAIMS involved in LEARNING.' A student creates a sentence by highlighting part of the concept map, and can edit a sentence by altering the map. The resulting sentences appear in a separate window (see Figure 2). This allows two very different (although "logically" equivalent) representations of a student's ideas; one embedded in the concept map's graphical structure and the other more textual in nature.

A student may publish a sentence, which makes it potentially visible to other students. We call this process making a claim. These published sentences — claims — become part of the class "knowledge soup," which consists of a highly organized "database" of simple sentences representing the growing knowledge of the group. It is through these knowledge soups that collaboration and sharing take place. Knowledge soups have many interpretations and can be displayed in several ways. They can be thought of as a body of text, an encoding of a large concept map, or an annotated collection of discussions between students.

Challenges and dialogs – peer review, not simple authority

Published claims can be seen by other students and can be utilized in their own map-building process, but a student can't see all claims published by the other students, as this would often be cognitively unmanageable. The system has heuristics about the relatedness of knowledge claims. The only claims that a student sees are those directly related to the ones that he contributed to the soup. As a student publishes more, a wider range of other related claims becomes visible. This strategy is intended to encourage and reward students for participation. The source of a claim remains anonymous.

A student can query a claim submitted by another student, if he disagrees with it or finds it puzzling, and the originator of the claim can respond. Querying a claim causes it to be displayed with a question mark after it to indicate to the author and any third parties that it is under dispute or discussion. The querier types a message which becomes invisibly attached to the query mark. Anyone — including, of course, the originator of the claim — can read this message and respond to it with a further comment, or an explanation or defense of the original claim. In this way, a published claim may become the locus of an extended discussion on some topic. The student's own claims are likewise subject to peer review.

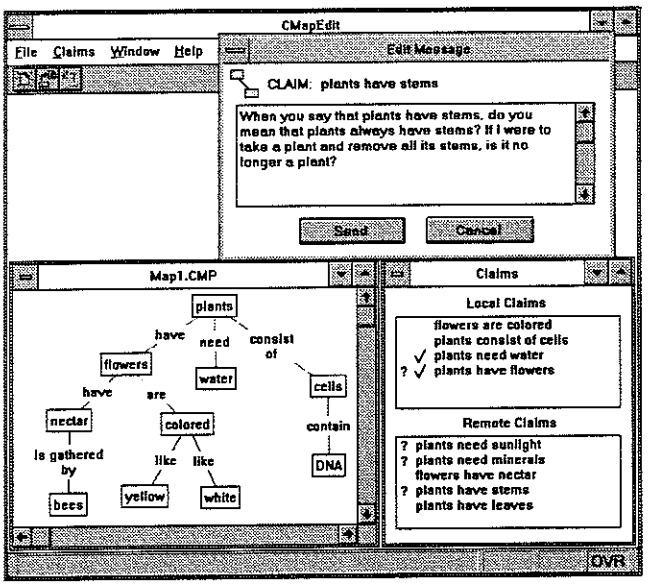


Fig. 2: Concept map, claims, and threads

The system keeps track of the authorship and time of all this publication, but it is not made public to the students. We judged that the advantages of anonymity in encouraging less socially confident students to risk publication and discouraging "flaming" outweighed the advantages of identifiability in encouraging students' pride in their work. (This strategy could be easily modified to allow signatures, identity-masking pseudonyms, etc., or even allow the teacher to decide which mode was best.) The way in which electronic communication is a "leveler" of social hierarchies is well known, and we expect to be able to take powerful advantage of this phenomenon. This mode of electronic communication and debate, leading to "threads" of related comments from several individuals, is an evolution of the widespread practice in CompuServe, Usenet, etc.

Since Pangea is not a communication channel, students cannot send personal messages to one another. The content, rather than the source, determines which parts of the soup are displayed to each student. In this way, we hope that Pangea will establish a new kind of group interaction amounting to a collaboration between students entirely in terms of what they know, a collaboration at the knowledge level.

Soups from distant lands

In addition to supporting knowledge level collaboration within a single classroom, knowledge soups can be made public for access by classrooms in other countries or continents. A teacher can import a knowledge soup from another classroom (or school) for his students to interact with. The same rich level of interaction supported for local soups is also provided for remote soups. A teacher in Caracas and another in Brazil, for example, could swap knowledge soups about the rain forest. (Figure 3) Likewise, teachers in several geographically dispersed schools could agree to have their students collaborate while developing concept maps on the same specific topic. The overall result will be a "kettle" of knowledge soups, collaboratively developed by students, and available to other students for their use in their knowledge construction.

An "artificial idiot"

We plan to make available an optional third mode of interaction, in which the system takes on the role of a friendly, helpful agent with whom the child may converse in a very simple way (see Figure 3). This agent will have access to all the information in the soup and is equipped with some heuristics for drawing plausible conclusions and asking questions.

The system will use information about a restricted sub-vocabulary of arc labels and some easily obtained grammatical information to draw tentative conclusions about relationships between the meanings of some of the assertions made in the student's growing cognitive map. These conclusions might use assertions taken from many maps. If Juan has previously claimed that 'fish have wings,' and Marta claims that 'fish can-be edible,' then the system might notice the connection, tentatively draw a (false) conclusion, and ask her if anything with wings can be edible. We hope that the friendly idiot's questions might often be stimulating to the learner's imagination, while acting in effect as an additional channel of communication between learners' understandings of the subject matter. Notice how different this system is from the overtly "debating" flavor of the assert-query-respond game in the threads associated with knowledge claims. We believe that it represents a new kind of interaction, in which a learner's ideas are contrasted or reinforced by those of other students without anything like communication, even anonymous communication, occurring between them.

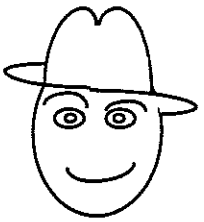


Fig. 3: The Artificial Idiot's image can be tailored by the student

We have chosen the personality of the agent with care, based in part on the recommendations of Novak and Gowin, (1984) for how teachers should interview students about their concept maps. It will be a kind of idiot savant who is nonjudgmental, nonintrusive, friendly, and hopefully amusing. (For example, we are developing a simple graphical interface using a cartoon face to convey puzzlement, interest, happiness, etc.) It uses the student's own language as much as possible and always tries to be helpful. It always leaves the student alone if told to. It knows a lot but sometimes reveals surprising ignorance. It never takes offense at being told it is a fool. In fact, it may ask the student for help or clarification when puzzled and always enjoys being taught new things by the student. The student may mislead it, of course, but to do so deliberately would be irresponsible, and such behavior is easily detectable since all transactions with the agent are internally recorded.

This agent gives a persona to the growing body of information concerning the domain of study. Just as this body grows as the class project advances, the knowledgeable fool can be expected to become more knowledgeable and his opinion more reliable as time advances. However, he should never be considered to be an authority figure, as he will almost certainly make silly mistakes from time to time no matter how much he comes to know. His role is not that of a Socratic source of wisdom, but that of a friendly colleague designed to encourage the students to reflect upon their knowledge claims.

The system has only a very weak grasp of what the various claims mean and does not have most of the everyday knowledge which humans use to help them interpret simple sentences. We cannot afford to put a large-scale knowledge base of commonsense information into the system, and in any case available evidence suggests strongly that this would not work. The conclusions it could support would almost certainly be too fragile for the often rather imprecisely expressed claims, and too idiosyncratic for the wide range of topics with which the system must be able to function. Rather, we plan to look for rather general patterns in the groups' own knowledge (as represented in the growing knowledge soup) which suggest gaps, misunderstandings, or inconsistencies in an individual student's claims.

The system will make inferences from the answers it receives, and it can sometimes check its tentative conclusions by simply asking the learner. This is the sense in which we say that the system is always willing to learn from the student. (The answers can of course be confirmed by going directly to a more authoritative source such as the teacher.) The agent can also sometimes learn a great deal by being told that its question

was silly. For example, suppose Marta replies that it is silly — as opposed to simply wrong or right — to ask whether all things with wings are edible, when prompted by the claim that fish are edible. The system can infer that something is suspicious about Juan's assertion concerning winged fish, information which might be otherwise difficult to obtain.

We are investigating several ways in which the system can put together information from several claims to draw tentative conclusions about the coherence or completeness of one learner's partial concept map. These ways involve the ideas of similarity, possibility, causal relations between types of events, and 'using' (for example, 'Plants use Sunlight for Photosynthesis'). We find that many of the arc labels used in classroom concept maps fall into one of these categories. Another key observation is that in a well-constructed concept map, the nodes are usually labelled with nouns, and the arcs with verbs or supporting vocabulary such as prepositions, and we expect to be able to recognize this distinction automatically. (The system currently uses it to initially translate maps to sentences.)

The system understands the basic notion of a concept hierarchy and recognizes some words which are commonly used as arc labels to express hierarchical relationships. These include 'is,' 'are,' 'is-usually,' 'may-be' and 'e.g.' It also understands the notion of a classification, in which one concept is divided into subclasses according to some property. The arc labels 'are-usually,' 'can-be,' and 'some-are' often indicate such a classification, so that we might find the claims 'Leaves can-be Deciduous,' 'Leaves can-be Evergreen,' etc.

One common pattern is a series of claims of the form 'A can-be B1' with an associated hierarchical series 'B1 is-a C,' which names the classification. (For example, 'Cows can-be White,' 'Cows can-be Black,' etc.; 'White is-a Color,' 'Black is-a Color,' etc. Notice however, 'Cows can-be Thin' makes sense, but 'Thin is-a Color' is a conceptual error.) The system can draw several useful tentative conclusions even from only a fragment of such information. For example, a single 'A can-be B' claim indicates fairly clearly that some A's are not B. This observation might prompt a question "Which A's cannot be B?" for example, and a positive answer (or a suitable adjustment to the concept map) of the form 'A can-be D' would support the tentative conclusion that B and D are of the same classificatory type. If learners have made several such claims, a question like "What are B, D, ... all called?" asking directly for the name of the type, can provide useful information. Once the type name is known, the system can ask directly whether a new concept appearing in a claim like 'A can-be X' is one of that type. This could have led to the question 'Is Thin a Color?' in the example above, and again, a negative answer allows the system to draw useful conclusions.

For example, a concept map containing any of 'White can-be Color,' 'White is Color,' or 'White are Cows' can now be immediately recognized as faulty; and more interestingly, a repair can be suggested ("Do you mean that white is a color, and some cows are white?" — [yes] — "You could say: 'White is-a Color' and 'Cows can-be White'." More subtly, a concept map which contains the arcs 'Cows can-be White' and 'Cows can-be Thin', but no other arcs of the form 'Cows can-be...', can now also be recognized as possibly indicating a conceptual confusion in the mind of the learner, and suitable questions could be designed to probe it, or the situation could simply be recorded for the teacher's benefit.

We should emphasize that much of the information in the soup will be meaningless to the agent. Failures which might be unacceptable for a natural-language understanding system, such as being unable to resolve ambiguities or to notice alternative lexical usages, are not a disaster here. On the other hand, new heuristic rules can be smoothly incorporated into the agent's repertoire of inferential abilities without affecting its overall classroom situation. We expect to treat this part of the system as an on going experiment.

This part of the system is still under development and has not yet been implemented, but we are excited by its potential. For example, while the "fool" is in many ways very limited in its abilities, it might access information in distant knowledge soups to bring ideas from other cultures, saying things like: "In Peru I heard that people eat guinea pigs."

Summary

Quorum is not a project that will vanish after three years. Our objective is to establish an infrastructure that includes hardware and software connectivity, technical and pedagogical support, and a momentum of interaction between schools that results in a self-perpetuating project which continues to grow. The continuing fundamental theme is to foster communication and collaboration at many levels, from simple exchange of electronic messages to collaboration across classrooms and continents at the knowledge level. We anticipate something which has never happened before. Whole generations of people in more than ten countries will be able to understand and empathize with one another in a way hitherto impossible, because they will all have gone to school together.

References

Ausubel, D. P., Novak, J. D. & H. Hanesian, H. (1978). Educational Psychology: A Cognitive View (2nd ed.). New York: Holt, Rinehart& Winston. Reprinted, 1986. New York: Warbel & Peck.

Cañas, A. J. (1993). Niños Colaborando a Travis de Fronteras: Logo y Telecomunicaciones, Proceedings of the VI Int. Logo Congress, Caracas, Venezuela.

Cañas, A. J., Schaffer, M. (1990). Project Genesis: Technology and Innovation in Education, First Int. Seminar: the Implementation of Computers in Education, Int. Society for Technology in Education (ISTE), Guatemala City, Guatemala.

Cañas, A. J. & Ford, K. (1992). An Environment for Collaborative Knowledge Building, presented at the Workshop on the Technology and Pedagogy for Collaborative Problem Solving as a Context for Learning, Toronto.

Escorcia, G., Cañas, A. J., & Schaffer, M. (1991). The Genesis Concept, Second Int. Seminar: the Implementation of Computers in Education, Int. Society for Technology in Education (ISTE), Mexico City, Mexico.

Newell, A. (1992) The Knowledge Level. Artificial Intelligence 18, 87-127

Novak, J. D. (1977). A Theory of Education. Ithaca, NY: Cornell University Press.

Novak, J. D. & Gowin, D. B. (1984). Learning How to Learn. New York: Cambridge University Press.

CONSTANT TIME INHERITANCE WITH PARALLEL TREE COVERS

Eunice (Yugyung) Lee and James Geller
Department of Computer and Information Sciences
New Jersey Institute of Technology
Newark, NJ 07102

Abstract

Inheritance along transitive relationships such as IS-A and transitive closure of such relationships play a significant role in knowledge representation research. One line of research by Schubert *et al.* [SPT87], continued by Agrawal, Borgida, and Jagadish [ABJ89, Jag88], has represented such relationships by graphs annotated with number pairs. In previous work Agrawal, Borgida, and Jagadish have shown how to minimize the use of memory by number pairs. In this paper we show that Agrawal *et al.*'s optimal encoding can be further improved by using parallelism. We present a *Maximally Reduced Tree Cover* with fewer number pairs than Agrawal *et al.*'s. Nevertheless, we still achieve constant time transitive closure queries by using a massively parallel implementation. Experimental results on a CM-5 Connection Machine are reported.

1 INTRODUCTION

In the past 12 years, AI has taken a turn towards proposing solutions in knowledge representation and reasoning, and then successively proving that most of those solutions result in intractable algorithms. This has advanced the state of the art of the field, but not the state of implemented knowledge representation and reasoning systems. However, a number of researchers have worked under a different paradigm [Kit91, EHA93, Sha93]. In this paradigm an improvement in speed of one order of magnitude is considered a qualitative change, as opposed to a quantitative change, especially if this improvement is scalable to large knowledge bases. The primary tool for achieving such scalable speed up has been the (massively) parallel computer.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/243 ©1996 FLAIRS

In the interest of returning to the original AI goal of computational (implemented) intelligence this paper presents an approach towards special purpose reasoning [Sha88, Sha89, SA90, Sha93], based on a parallel implementation [Sha93]. The special purpose reasoning that we are addressing deals with transitive relationships such as IS-A and PART-OF. Such transitive relationships play a significant role in knowledge representation research. This is especially the case for the IS-A relationship, which forms the backbone of many knowledge bases. The need to perform fast inheritance and answer queries involving large transitive relationships has led researchers to precompute the transitive closure of relationships such as IS-A and to store the result as materialized transitive closure.

Such a materialized transitive closure must permit fast look-up, and reasonably fast incremental update, without requiring much more storage than the original (IS-A) relationship. This excludes many naive approaches which would require $O(n^2)$ space for a relationship graph of $O(n)$ [ABJ89]. One way to improve the speed of look-up in a materialized transitive closure is to use parallelism [GY92, LG93].

In [LG94] a massively parallel transitive closure reasoner, called Hydra,[1] that can dynamically assimilate *any* transitive, binary relationship and efficiently answer queries using the transitive closure of all those relationships was developed.

For example, Hydra[2] can respond to questions using transitive PART-OF relationships [SPT87], or to questions of the kind "Is an elephant bigger than a can opener"? if it knows that an elephant is

[1]This research was (partially) done under a cooperative agreement between the National Institute of Standards and Technology Advanced Technology Program (under the HIIT contract, number 70NANB5H1011) and the Healthcare Open Systems and Trials, Inc consortium.

[2]This work was conducted using the computational resources of the Northeast Parallel Architectures Center (NPAC) at Syracuse University, which is funded by and operates under contract to DARPA and the Air Force Systems Command, Rome Air Development Center (RADC), Griffiss Air Force Base, NY, under contract# F306002-88-C-0031.

bigger than a person, and a person is bigger than a can opener. The efficiency of this reasoner is achieved by combining massively parallel hardware [Hil85, WS88, Wal90, Kit91] with special-purpose reasoning.

The Hydra reasoner is based on Schubert *et al.*'s [SPT83, SPT87] special-purpose reasoner for subclass verification. In his model, a class tree is represented by a numeric coding schema, assigning one number pair per node. This makes it possible to decide in constant time whether a class B is a subclass of a class A. In [GD91], Schubert *et al.*'s model was extended to include an efficient parallel update operation. Extensions of Schubert *et al.*'s work towards directed acyclic graphs (DAGs) have been presented in [ABJ89]. There Agrawal *et al.* permitted more than one number pair at each node. Therefore, it is possible to verify the existence of IS-A relationships between two nodes in time proportional to the number of number pairs at the upper node, even for graphs.

Even though Agrawal *et al.*'s main result is to prove that a spanning tree can be constructed that propagates a minimal number of number pairs, quite a large amount of space is still required to store all propagated pairs. In this paper, we will show that the number of stored number pairs can be further reduced by delaying, in effect, the propagation of some pairs to query time. This appears, at first, to contradict the main goal of this research, namely to speed up transitive closure queries by storing additional number pairs. However, luckily, by adopting the massively parallel representation of [LG93], the missing pairs do not lead to a query time penalty! In this way, we can actually reduce the storage requirements for number pairs beyond Agrawal *et al.*'s optimality result. We call this new representation a *Maximally Reduced Tree Cover.* The heart of this paper is to show (1) the structure of the Maximally Reduced Tree Cover and (2) how to maintain constant time transitive closure queries with the maximally reduced set of propagated number pairs.

2 PREVIOUS WORK

We now summarize Schubert *et al.*'s [SPT87] and Agrawal *et al.*'s [ABJ89] approaches. A spanning tree is designed by selecting parents with maximal numbers of predecessors, like in Figure 1. (Arcs that are not part of the spanning tree are shown by dashed lines.) This spanning tree can then be assigned number pairs, like in [SPT87]. (The method is simple, but will be omitted here for space resons.) These number pairs [π μ] are called *tree pairs*. Then *graph arcs* are used to propagate tree pairs upward. Such propagated pairs are called *graph pairs* and we use the notation (π μ) for them. In Figure 1 the node H IS-A F, because [4 5] is a subinterval of [2 5]. In the same figure E has the tree pair [6 6] and the graph pair (8 9) which was propagated to it from the node C. Using the graph pair, it is now possible to verify that D IS-A E because [9 9] is a subinterval of (8 9). Agrawal *et al.*'s optimality result guarantees that the number of graph pairs in the whole graph is minimal for a serial implementation with runtime proportional to the number of number pairs at each node.

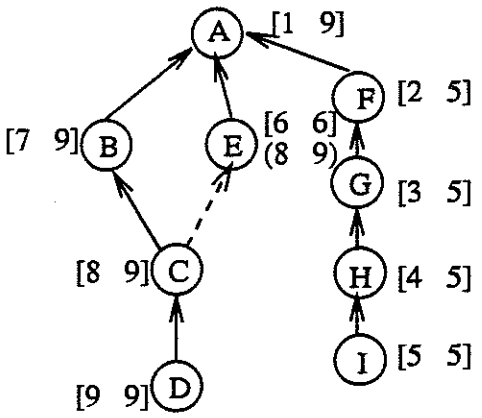


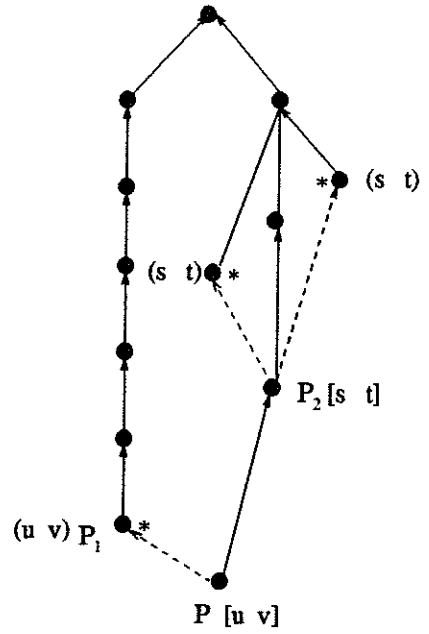
Figure 1: Agrawal's Encoding Graph

In this paper we will show how, with some limitations, a parallel implementation of Agrawal *et al.*'s approach is possible, that permits constant time subclass verification, independent of the number of number pairs at each node. This implementation is based on what we call the Maximally Reduced Tree Cover. The Maximally Reduced Tree Cover needs only in the unlikely worst case as many number pairs as Agrawal *et al.*'s optimal tree cover.

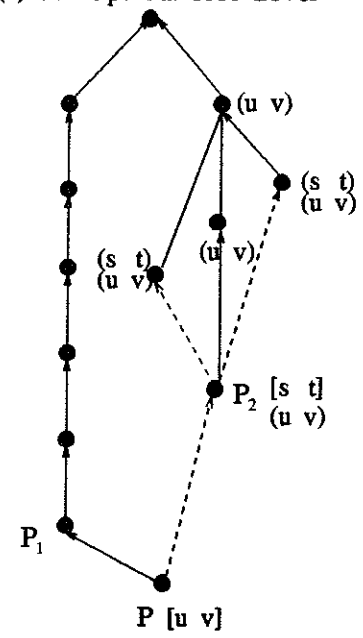
3 SPANNING TREE CONSTRUCTION

Our improvements are possible by changing the way the initial spanning tree is designed. We define the notion of *weak predecessor*. Informally, a weak predecessor of a node X is a predecessor which is reachable from X by an upward path containing at least one graph arc. A formal definition of "weak predecessor" is somewhat lengthy and will be omitted for space reasons.

Based on this definition we can prove the following



(a) Our Optimal Tree Cover



(b) Agrawal's Optimal Tree Cover

Figure 2: Examples of Optimal Tree Cover

main result:

Theorem 1: Assume a node P with parents P_1, P_2, ..., P_m. During construction of a spanning tree one of the P_is is chosen as the tree parent. If the P_i with the maximum number of weak predecessors is chosen, then the resulting tree cover will have the minimum number of propagated pairs. (The number of weak predecessors is computed assuming that the weak predecessor is not also a tree predecessor.) If several P_is have the same maximum number we choose one randomly.

Proof: Let's assume, without loss of generality, that there are m parents, P_1, P_2, ..., P_m. Let's say that P_1 has k_1 weak predecessors, P_2 has k_2, ..., and P_m has k_m weak predecessors. Let's say that $k_1 > k_i$ where $i > 1$ and $i \leq m$.

If we choose P_1 as part of the spanning tree (Figure 2(b)), then P will be connected to P_i, $2 \leq i \leq m$, by graph arcs. That makes P_2, P_3, ..., P_m weak predecessors of P. In addition, all weak predecessor of P_i, $i > 1$ and $i \leq m$, are also weak predecessors of P. Let's assume, without loss of generality, that P has only a tree pair. Then we need to propagate this tree pair to all k_i weak predecessors of P_i, and to P_i where $i > 1$ and $i \leq m$. In total,

$$\sum_{i=2}^{m} k_i + m - 1$$

pairs are propagated. For every i, because $k_i < k_1$, $k_i + m - 1 < k_1 + m - 1$, and therefore choosing P_1 propagates fewer pairs than choosing any other P_i ($i > 1$ and $i \leq m$). ■

If we compare with Agrawal *et al.*'s tree cover, pairs are propagated to all predecessors where they are not redundant. In our tree cover, pairs are propagated only to *weak* predecessors where they are not redundant. In Figure 2, (u v) and (s t) are graph pairs propagated through graph arcs. The *weak* predecessors are represented in Figure 2(a) with the symbol "*". As Figure 2(b) shows, Agrawal *et al.*'s tree cover has 7 graph pairs while in (a) only 3 graph pairs are generated by our method.

4 PROPAGATION OF GRAPH PAIRS

Let "B IS-A* A" mean that there is a path of IS-A arcs from B to A. If B IS-A* A, all successors of B are successors of A. Whenever there is a tree path from B to A, we claim that we can achieve the effect of having all graph pairs of B at A, without actually propagating these pairs to A, resulting in an additional

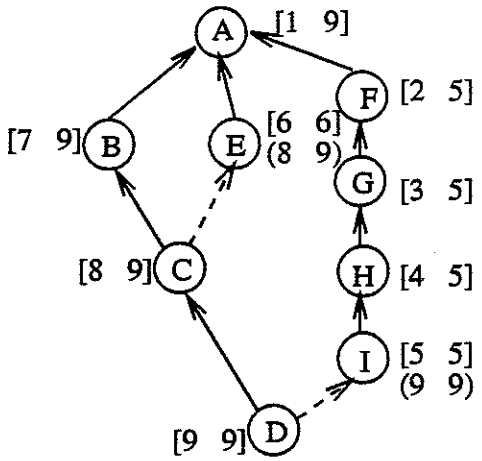


Figure 3: Our Maximally Reduced Tree Cover After Inserting a Link into Figure 1

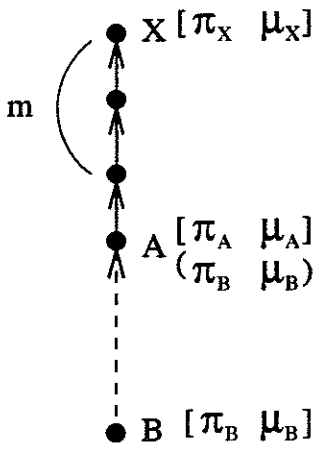


Figure 4: Maximally Reduced Propagation

saving of space. Above "achieve the effect of having all graph pairs" means that we can perform constant time subclass verification and all operations that rely on subclass verification, including propagation itself.

Lemma 1: There is a tree path from B to A iff the tree pair $[\pi_B\ \mu_B]$ forms a subinterval of the tree pair $[\pi_A\ \mu_A]$.

Proof: Trivial, by the definition of the tree numbering. ■

Now let's think about the other case, in which B points to A with a graph arc.

Lemma 2: If there is a graph, labeled according to [ABJ89], with a graph arc from B to A and B has a pair $[\pi_B\ \mu_B]$ and A has a pair $[\pi_A\ \mu_A]$ and A has m tree predecessors then (1) there exists a pair $(\pi_B\ \mu_B)$ at A and (2) there exists a pair $[\pi_X\ \mu_X]$ at each predecessor of A, such that $\pi_X < \pi_A$ and $\mu_A \leq \mu_X$.

Proof: (1) follows from the need to propagate $(\pi_B\ \mu_B)$ according to [ABJ89]. (2) follows by applying Lemma 1 m times. ■

In effect (Figure 3), Lemma 2 will permit us to verify that D is a successor of any tree predecessor of I, without propagating D's pair [9 9] to I's tree predecessors (H, G, F, A).

Referring to Figure 4, our claim is that we need to propagate all pairs of B and all graph pairs of B's tree successors to weak predecessors of B only, to maintain constant time queries. How do we find all weak predecessors of a node B?

Lemma 3: Iff G is a graph labeled according to the Maximally Reduced Tree Cover algorithm and I is a weak predecessor of D, then I will have at least one graph pair propagated from D or from a tree predecessor of D, namely C.

As an example of Lemma 3, in Figure 3, E and I are both weak predecessors of D. I receives the pair (9 9) from D, while E must have a graph pair from C, namely (8 9). The proof relies on Figure 3, but the arguments given are perfectly general.

Proof:

Case 1: Let's assume that we insert a graph arc from D to I. All pairs of D need to be propagated to I, which is a weak predecessor. This directly follows from the Maximally Reduced Tree Cover algorithm.

Case 2: Now, let's assume that there are several tree predecessors of D. Let C be one of the tree predecessors of D. By inserting a graph arc from C to E, all pairs of C should be propagated to E, following again the Maximally Reduced Tree Cover algorithm. Because E is a weak predecessor of D, and E receives at least one pair from C, a tree predecessor of D, the Lemma is true in this case too. ■

5 PROCESSING TRANSITIVE CLOSURE QUERIES

Now, we will describe how to achieve constant time transitive closure queries with the maximally reduced set of propagated graph pairs. Suppose that we want to verify whether B is a subclass of A in a graph G. However, the number pair which verifies the relationship between the two nodes might not be available at A, because pairs are propagated only to the "weak predecessors" by our Maximally Reduced Propagation algorithm, when an arc is inserted into a graph.

Therefore, we need to collect propagated graph pairs from all tree successors of A (including A it-

self) that are also predecessors of B. But because of parallel processing, the verification step can be done in constant time.

Parallel Algorithm: Verification of B IS-A* A.

```
; B is a A iff IS-A-VERIFY returns T

IS-A-VERIFY (B, A: Node): BOOLEAN
   RETURN(IS-A-VERIFY-1(B, A) OR
          IS-A-VERIFY-2(B, A));

IS-A-VERIFY-1 (B, A: Node): BOOLEAN
; If B is a tree successor of A,
; then the tree pair of A subsumes the tree
; pair of B.
    ACTIVATE-PROCESSORS-WITH
       PRENUM(B) >=!! PRENUM(A) AND!!
       MAXNUM(B) <=!! MAXNUM(A)
    DO BEGIN
      IF any processor is still active THEN
        RETURN T;
    END

IS-A-VERIFY-2 (B, A: node): BOOLEAN
; Activate A and all tree successors of A.
; (This can be done, by using Lemma 2.)
1   ACTIVATE-PROCESSORS-WITH
2      PRE!! >=!! PRENUM(A) AND!!
3      MAX!! <=!! MAXNUM(A) AND!!
4      PROCESSOR STORES TREEPAIR
5   DO BEGIN
6      SET-MARK!!
7   END

; Test whether any marked processor has the
; tree pair from B or from a tree predecessor
; of B, as a graph pair. If this is the case,
; return T. (Lemma 3)
8   ACTIVATE-PROCESSORS-WITH
9      PRE!! <=!! PRENUM(B) AND!!
10     MAX!! >=!! MAXNUM(B) AND!!
11     PROCESSOR STORES GRAPHPAIR
12     IS-MARK-SET-P!!
13  DO BEGIN
14     IF any processor is still ACTIVE THEN
15       RETURN T;
16  END
```

A *parallel variable* can be thought of as an array where every element is accessible in parallel on its own processor [Thi88]. In the algorithm, the expression PRE!! stands for a parallel variable that contains

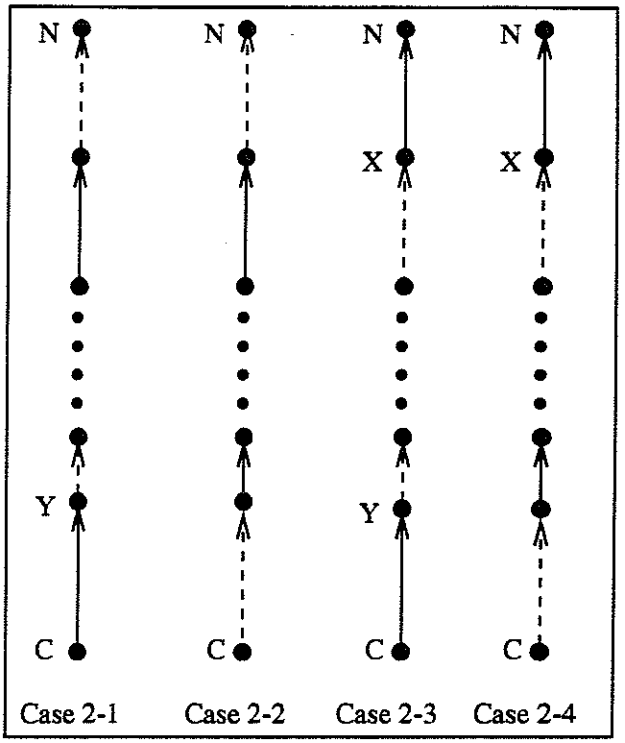


Figure 5: Possible Kinds of Propagation Paths

for every node (processor) its preorder number, and the expression MAX!! stands for a parallel variable that contains for every node its maximum number. PRENUM(A) and MAXNUM(A) retrieve the number pair for the given node A. SET-MARK!! is a function that sets a parallel flag on every active processor. IS-MARK-SET-P!! is a boolean function that returns TRUE on processors if the flag SET-MARK!! is set on them.

Conceptually speaking, what happens in IS-A-VERIFY-2 is that we are searching upwards from B for predecessors, and downwards from A for tree successors. Because some tree successor of A is guaranteed to have pairs that should be propagated to A, but are not, we have the same effect as if we would have propagated these pairs.

What we have to show now is, that subclass verification in constant time is possible for the Maximally Reduced Tree Cover. Suppose that we want to verify whether C is a subclass of a distant node N in a graph G. Let *weakly terminated path* be a path that consists of a tree path of length n, $n \geq 0$ followed by a single graph arc.

Theorem 2: Subclass verification with the Maximally Reduced Tree Cover can be performed in con-

stant time.

Proof: Typically, we may have two possible forms of path from a child node C to a predecessor N. First, N is reachable from C only through a tree path and there is no weakly terminated path from C to N. Second, N is reachable from C through at least one weakly terminated path. We want to show that the subclass verification can be done in constant time, no matter how many number pairs are at N or at C.

Case 1: Tree path only: According to Lemma 1, we can easily verify C IS-A* N by testing whether the tree pair $[\pi_C\ \mu_C]$ is a subinterval of the tree pair $[\pi_N\ \mu_N]$.

Case 2: At least one weakly terminated path: Assume that X is a tree successor of N and Y is a tree predecessor of C. We may have four possible subcases (Figure 5):

Case 2-1: The tree pair of Y is propagated to N, because N is a weak predecessor of C (Lemma 3). N is marked by lines 1 to 6 of IS-A-VERIFY-2. By lines 8 to 15, IS-A-VERIFY-2 returns T, by Lemma 3, because N is marked and because N has a pair from Y which is a tree predecessor of C.

Case 2-2: The tree pair of C is propagated to N, because N is a weak predecessor of C. N is marked by lines 1 to 6. IS-A-VERIFY-2 returns T because N is marked and, again, by Lemma 3, has a pair from C.

Case 2-3: The tree pair of Y is propagated to X because of Lemma 3, because X is a weak predecessor of Y. All processors on the tree path from X to N are marked by lines 1 to 6 in IS-A-VERIFY-2. (Because number pairs which are propagated to tree successors of N have the effect of being propagated to N by Lemma 2.) The check whether any marked processors have pairs propagated along the tree path from C to Y is done by lines 8 to 15. IS-A-VERIFY-2 returns T because X is marked and has a pair from Y, and Y is a tree predecessor of C.

Case 2-4: The tree pair of C is propagated to X, by Lemma 3, because X is a weak predecessor of C. All processors on the tree path from X to N are marked by lines 1 to 6 by Lemma 2. IS-A-VERIFY-2 returns T because X is marked and has a pair from C.

What is missing is an argument that no other kind of path can exist between C and N. Said in another way, we need to show that every possible succession of arcs can be generated from our cases.

A pure tree arc or tree path can be generated by Case 1. Because a single graph arc defines a weak predecessor, we can generate a single graph arc wherever we like by the basic form of Case 2-2. Because a tree arc or path followed by a single graph arc defines a weak predecessor, we can generate a single tree arc wherever we like, except at a place where it has no graph arc above it. Because of that limitation we have to separately consider a path that is terminated by a tree path (Cases 2-3 and 2-4). Because a weak predecessor might have a path that starts with a tree path or not, we also have two cases for the initial segment (Cases 2-1 and 2-3 as opposed to Cases 2-2 and 2-4). In summary, with our 5 cases we can generate every possible path between two nodes. As it was shown for every case that constant time verification is possible, we have shown that this algorithm performs constant time subclass verification for every pair of nodes. ■

If we compare with Agrawal *et al.*'s tree cover, they are propagating pairs to all predecessors where they are not redundant. In our tree cover, pairs are propagated only to *weak* predecessors where they are not redundant.

6 EXPERIMENTAL RESULTS

In this section we present experimental results using an existing large medical vocabulary, the INTERMED (INTERnet version of the Medical Entities Dictionary) system of CPMC (Columbia Presbyterian Medical Center) [CHJC89, CEB92, CAJP93, CCHJ94]. The INTERMED system currently has about 2,500 medical terms. These terms are related by general relations such as IS-A, but also by domain specific relations such as PHARMACEUTIC-COMPONENT-OF. The experiments were done on a Connection Machine CM-5, which makes use of groups of virtual processors executing serially on real processors. There are 32 real processors on the NPAC CM-5. Every processor emulates the activities of at least 32 virtual processors. The CM-5 [Thi88] is programmed in *LISP.

The purpose for this experiment is to show that our Maximally Reduced Tree Cover is better than Agrawal *et al.*'s tree cover in terms of the number of graph pairs generated. We have performed a set of two experiments that analyze the space and run-time in the Maximally Reduced Tree Cover and Agrawal *et al.*'s tree cover.

First, we extracted information from the INTERMED and then translated it into a format fit for our system. Each term in the INTERMED is treated as a node and we have constructed a hierarchy for each experiment. Each hierarchy was built as follows: (1) We created a class hierarchy following Agrawal *et al.*'s schema (AST) (2) We created a class hierarchy according to our Maximally Reduced Tree Cover (MRT).

Second, we stored the node set of each hierarchy in processor space on the CM-5 using our node set representations. Finally, we tested our verification and link insertion operations on both hierarchies. Refer to [LG93] for details of the link insertion operations. The results in Table 1 show that our Maximally Reduced Tree Cover is better than the other tree cover. The first row and second row show the type of tree cover and the height of the trees. The third, forth, and fifth rows show the number of tree pairs, the number of graph pairs, and the number of virtual processors (4K or 8K), respectively. The last three rows show the average run-times for the subclass verification algorithm and the link insertion algorithm. Specifically, IS-A-Verify-1 and IS-A-Verify-2 represent the first and the second cases of the parallel verification Algorithm in Section 5. All times are in seconds.

Type	*AST*	*MRT*
Level#	10	10
Tree Pair#	2495	2495
Graph Pair#	1442	744
Processor#	8K	4K
IS-A-Verify-1 (sec)	0.00042	0.00035
IS-A-Verify-2 (sec)	0.0085	0.0082
Insertion (sec)	0.139	0.087

Table 1: Experimetal Results for Three Approaches

The experiments show clearly that our Maximally Reduced Tree Covers are better than Agrawal *et al.*'s in terms of the number of graph pairs and the average run-times for the subclass verification and the link insertion algorithms. Specifically, our tree cover schema generates fewer graph pairs than Agrawal *et al.*'s, i.e., 744 versus 1442. Additionally, the average run-times for the subclass verification and the link insertion algorithms with our Maximally Reduced Tree Cover is slightly better than for Agrawal *et al.*'s tree covers.

7 CONCLUSION

In this paper we have discussed techniques for achieving fast transitive queries in IS-A hierarchies, by using parallel processing and a number pair encoding of the hierarchies. We have introduced a representation that improves Agrawal *et al.*'s optimal tree cover, called the Maximally Reduced Tree Cover. By improving Agrawal *et al.*'s tree cover we achieved an additional reduction in the storage necessary for number pairs. In addition, we achieved transitive closure queries in constant time by using parallel processing, even with several number pairs at each node. Experimental results show that our Maximally Reduced Tree Cover achieves higher performance compared with Agrawal *et al.*'s in terms of the memory which is required for number pairs and the average run-time for subclass verification and insertion algorithms.

References

[ABJ89] R. Agrawal, A. Borgida, and H. V. Jagadish. Efficient management of transitive relationships in large data and knowledge bases. In *Proceedings of the 1989 ACM SIGMOD International Conference on the Management of Data*, pages 253–262, Portland, OR, 1989.

[CAJP93] J. J. Cimino, A. A. Aguirre, S. B. Johnson, and P. Peng. Generic queries for meeting clinical information needs. *Bulletin of the Medical Library Association*, 81(2):195–206, 1993.

[CCHJ94] J. J. Cimino, P. D. Clayton, G. Hripcsak, and S. B. Johnson. Knowledge-based approaches to the maintenance of a large controlled medical terminology. *Journal of the American Medical Informatics Association*, 1(1):35–50, 1994.

[CEB92] J. J. Cimino, P. L. Elkin, and G. O. Barnett. As we may think: The concept space and medical hypertext. *Computers and Biomedical Research*, 25:238–263, 1992.

[CHJC89] J. J. Cimino, G. Hripcsak, S. B. Johnson, and P. D. Clayton. Designing an introspective, multipurpose, controlled medical vocabulary. In *Proceedings of the Thirteenth Annual Symposium on Computer Applications in Medical Care*, pages 513–518. IEEE Computer Society Press, Los Alamitos, CA, 1989.

[EHA93] M. P. Evett, J. A. Hendler, and W. A. Andersen. Massively parallel support for computationally effective recognition queries. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 297–302. MIT Press, Cambridge, MA, 1993.

[GD91] J. Geller and C. Y. Du. Parallel implementation of a class reasoner. *Journal of Experimental and Theoretical Artificial Intelligence*, 3:109–127, 1991.

[GY92] K. Guh and C. Yu. Efficient management of materialized generalized transitive closure in centralized and parallel environments. *IEEE Transactions on Knowledge and Data Engineering*, 4(4):371–381, 1992.

[Hil85] W. D. Hillis. *The Connection Machine.* MIT Press, Cambridge, MA, 1985.

[Jag88] H. V. Jagadish. A compressed transitive closure technique for efficient fixed-point query processing. pages 209–223, San Francisco, CA, 1988.

[Kit91] H. Kitano. Massively parallel AI and its application to natural language processing. In *Proceedings of the Workshop on Parallel Processing for AI at IJCAI 1991*, pages 99–105, Sydney, Australia, 1991.

[LG93] E. Y. Lee and J. Geller. Representing transitive relationships with parallel node sets. In Bharat Bhargava, editor, *Proceedings of the IEEE Workshop on Advances in Parallel and Distributed Systems*, pages 140–145. IEEE Computer Society Press, Los Alamitos, CA, 1993.

[LG94] E. Y. Lee and J. Geller. Constant time verification of transitive query with parallel tree covers. Technical Report Technical Report CIS94-53, Department of Computer and Information Science, 1994.

[SA90] L. Shastri and V. Ajjanagadde. An optimally efficient limited inference system. In *Proceedings of IJCAI-90*, pages 563–570, Boston, MA, 1990.

[Sha88] L. Shastri. *Semantic Networks: an Evidential Formalization and its Connectionist Realization.* Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[Sha89] L. Shastri. Default reasoning in semantic networks: a formalization of recognition and inheritance. *Artificial Intelligence*, 39(3):283–356, 1989.

[Sha93] L. Shastri. A computational model of tractable reasoning – taking inspiration from cognition. In *Proc. of the 13th Int. Joint Conference on Artificial Intelligence*, pages 202–207. Morgan Kaufmann, San Mateo, CA, 1993.

[SPT83] L. K. Schubert, M. A. Papalaskaris, and J. Taugher. Determining type part, color, and time relationships. *Computer*, 16(10):53–60, 1983.

[SPT87] L. K. Schubert, M. A. Papalaskaris, and J. Taugher. Accelerating deductive inference: special methods for taxonomies colors and times. In N. Cercone and G. McCalla, editors, *The Knowledge Frontier*, pages 187–220. Springer Verlag, New York, NY, 1987.

[Thi88] Thinking Machines Corporation. **LISP Reference Manual Version 5.0 edition.* Thinking Machines Corporation, Cambridge, MA, 1988.

[Wal90] D. L. Waltz. Massively parallel AI. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 1117–1122. Morgan Kaufmann, San Mateo, CA, 1990.

[WS88] D. L. Waltz and C. Stanfill. Artificial intelligence related research on the Connection Machine. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 1010–1024, 1988.

ACQUIRING CAUSAL KNOWLEDGE BY MEANS OF LESION EXPERIMENTS

Gregory D. Weber
Indiana University East
gdweber@indiana.edu

Abstract

In this paper, I describe the Controlled Lesion Method (CLM), a set of principles for inferring the causal structure of deterministic mechanisms by experimentation. CLM formulates an important part of the "logic" of causation insofar as it relates to experiment. It requires minimal causal background knowledge. For causal chains of binary variables, a program using CLM gives more complete results than statistical algorithms, makes no spurious causal inferences, and has polynomial worst-case time. I will describe three simulation studies and suggest extensions to CLM which may make it applicable to causal trees and indeterministic systems.

1 INTRODUCTION

It is generally recognized that causal knowledge provides an intelligent agent with a deeper understanding than can be supplied by knowledge of mere correlations, such as the correlations of symptoms with diseases. Causal knowledge is the basis of many expert systems. It plays a key role in qualitative physics, understanding narrative discourse, planning, and solving problems of diagnosis, repair, and control. Causal knowledge seems to be deeply connected with the nature of human intelligence, yet its origin—how it is acquired—remains little understood.

Methods of inferring causation from statistical relations of conditional independence, such as IC [9] and SGS [3], have made remarkable progress in recent years. These methods work best when there is plenty of random error or "noise." Human intelligence has the opposite strength: we are more able to discover causal relationships in a deterministic ("noise-free") or nearly deterministic environment—so much so that until the twentieth century nearly all philosophers regarded causation as a necessarily deterministic relation [1]. Evidently this human ability has important survival value, and a robot will need to be able to discover causal relations in noise-free, and not just noisy, environments.

In the deterministic case, it is clearly not enough to observe that A's and B's always occur together. That doesn't tell us whether A causes B ($A \rightarrow B$) or B causes A ($B \rightarrow A$) or some other event causes them both ($A \leftarrow C \rightarrow B$). To discriminate between the three possibilities requires either an experiment or some other causal knowledge.

To perform an experiment which will reveal something about causal relations, one must be able to distinguish between dependent and independent variables; the independent variables can then be interpreted as causes of the dependent variables. Żytkow has made the interesting suggestion that a variable for which the agent has a "measurement procedure" (an operator to observe the value) can be regarded as dependent, and variables for which the agent has a "manipulation procedure" (an operator to set the value) can be regarded as independent. [14] However, this strategy can go wrong. Suppose the agent has learned to make a certain kind of sound, *sound1*, by pressing down on a certain piano key. This pressing is a manipulation procedure for the variable *sound1*, which therefore can be considered independent. By looking at the key, the agent can determine its position; this looking is a measurement procedure for the variable *key1-down*, which can therefore be considered dependent. If independent variables cause dependent variables, we are led to the erroneous conclusion "*sound1* causes *key1-down*."

Scriven [11] proposed a non-reductive, recursive definition of causation in which the base case requires an experiment. This definition was the ba-

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/251

sis for AITION, a program which designed, executed, and analyzed experiments to infer causal relations in a simulated world [13], avoiding several fallacies of the kind just described. But the program required strong causal background knowledge. What is wanted is a method of acquiring this background knowledge. Part of the solution is the *Controlled Lesion Method (CLM)*, the subject of this paper.

2 PRINCIPLES

CLM is based on two key ideas. Informally, the *Random Operator Principle* (**R**) states that operators, by which the experimenter controls variables, are randomly determined; it is used to infer the *direction* of causation (operators → state variables). The *Lesion Principle* (**L**) is based on the notion that by inflicting "damage" on parts of the system and observing the resulting disruptions of associations, we can infer the *order* of causation, i.e., undirected causal betweenness relations ($A - B - C$). CLM uses the order information from **L** and the directional information from **R** to construct a directed graph representing the causal structure.

More formally, the **Random Operator Principle** is:

> **R**: Let X be any observed state variable. Let O be any operator. Then, (1) X does not cause O; (2) X and O are not both caused by any (unobserved) state variable Y.

R eliminates causal structures with arrows into O (such as $X \rightarrow O \rightarrow Y$). By Occam's razor, we may normally eliminate complex structures (multiply-connected and cyclic graphs).

For example, consider a light-bulb and switch system described by two binary variables: A = the switch is on, B = the light is shining. We observe A if and only if B. As experimenters, we can perform a certain operation, or pattern of movements, O, which we can choose in two ways: $O = 1$ means, roughly, "move hand to switch, grasp, and pull up"; $O = 0$ means "move hand to switch, grasp, and push down." This operation controls both A and B, but we don't yet know in what way. There are 64 possible graphs with the variables A, B, and O as vertices. Applying **R** and Occam's razor rules out all but three of these: (1) $O \rightarrow A \rightarrow B$; (2) $O \rightarrow B \rightarrow A$; (3) $A \leftarrow O \rightarrow B$.

The Lesion Principle allows us to discriminate between these three. *Lesion* of a variable Y means an alteration of the mechanism which restricts Y to one value. For example, breaking or unscrewing the bulb lesions B; taping the switch lesions A. The **Lesion Principle** is:

> **L**: Lesion of Y in a mechanism M disrupts an association ($X \cong Z$) between variables X and Z if and only if Y lies on the causal path between X and Z in M.

Examples: in the mechanism $X \rightarrow Y \rightarrow Z$, lesion of Y disrupts ($X \cong Z$); but in $X \rightarrow Z \rightarrow Y$, lesion of Y fails to disrupt ($X \cong Z$).

In our sample problem, experimenting with lesions of both A and B enables us to eliminate all but one hypothesis, as summarized in Table 1.

L_A disrupts ($O \cong B$)?	L_B disrupts ($O \cong A$)?	Conclusion
true	*true*	Error
true	*false*	$O \rightarrow A \rightarrow B$
false	*true*	$O \rightarrow B \rightarrow A$
false	*false*	$A \leftarrow O \rightarrow B$

Table 1: Logic of Lesion Studies

The basic step in CLM is to apply the logic of this table (which follows directly from **L**) to establish the causal structure connecting any similar trio of variables (one operator and two state variables). If there is a causal chain such as $O \rightarrow A \rightarrow B \rightarrow C \ldots$, the results for each pair of state variables can be integrated to establish the total causal structure of the chain.

3 IMPLEMENTATION

SCLM-1 is a computer implementation of CLM, written in the Scheme programming language. The program interacts with a simulation model of a causal mechanism, using binary atomic variables to represent operators and state variables. It is initially given a list of operators, a list of observable variables, and the model itself; all other information is obtained by interacting with the model, i.e., applying operators and observing state variables. Its output is a graph representing the causal structure of the simulated mechanism. A link $X \rightarrow Y$ is an assertion that X *directly* causes Y. "Direct" and "indirect" must be understood as relative concepts: if U is a set of variables, then X directly causes Y (with respect to U) if and only if X causes Y and there is no other variable $Z \epsilon U$ such that X causes Z and Z causes Y. What is direct with respect to a small set of variables may be indirect with respect to a larger set. If U is understood to be a universal set of variables (such as the set of all known or observed variables), we will

say that a causal influence is *locally direct* if it is direct with respect to a proper subset of U, and we will say that a causal influence is *globally direct* if it is direct with respect to U. The links in the final output represent globally direct causation. The absence of a link is simply the non-assertion, not the denial, of a direct causal influence.

The main action of the program is:

```
For each operator o,
  find all pairs of variables a, b that
    covary when o is randomly controlled;
  and, for each such pair, explain the
    correlations between o, a, and b.
```

The explanation is obtained by performing lesion experiments and interpreting them according to the logic of Table 1; this requires three experiments for each trio (o, a, b) of correlated state variables a, b, and operator o. In addition, SCLM-1 must perform an experiment for each operator, to find the variables correlated with it; and it must perform experiments to identify additional operators to control the lesions of a and b; these are called "lesion operators." Since the same experiment may be wanted many times, experimental results are stored in a table and looked up as needed, without re-executing the experiment.

For causal structures involving more than three variables, this main action is followed by a step of integrating the partial results for each trio into a single graph representing the total causal structure. The partial results show *locally* direct causation. Consequently, the integration step requires two types of retractions. (1) If there is a link $X_i \rightarrow X_k$ and there is an alternate path $X_i \rightarrow X_j \rightarrow \cdots \rightarrow X_k$, the link $X_i \rightarrow X_k$ is retracted. The longer path explains the causal influence of X_i on X_j, and therefore overrides any local evidence of direct causation. (2) For some trios (o, a, b) the program is unable to design a lesion experiment because it cannot identify appropriate lesion operators. In this case, links between all three variables ($o \rightarrow a$, $o \rightarrow b$, $a \rightarrow b$, $b \rightarrow a$) are "prohibited." The reason for this is that if the program were able to perform the lesion experiment, it would use it to eliminate some of these links (by the first type of retraction). Without the evidence of the lesion experiment, it is not safe to infer the presence of these links. The program accumulates a list of the prohibited links and, during the integration step, retracts any of the links that were prohibited.

4 RESULTS

I will describe the results of three simulation studies which tested the correctness and completeness of SCLM-1, and an analysis of its run time. By *correctness* I mean that the program's output includes a link $X \rightarrow Y$ only if X globally directly causes Y (i.e., no links are falsely asserted). By *completeness* I mean that for every X and Y such that X globally directly causes Y, the output contains a link $X \rightarrow Y$ (i.e., no real links are omitted).

I. The first study tested SCLM's ability to distinguish the three possible causal structures of three variables (two state variables and one operator), namely, $O \rightarrow A \rightarrow B$, $O \rightarrow B \rightarrow A$, and $A \leftarrow O \rightarrow B$. SCLM invariably found the correct structure among the three principle variables O, A, B, although it did not find all of the causal structure involving the lesions of these variables and the lesion operators. E.g., Figure 1 shows the model's true causal structure and the structure found by SCLM for the case of $A \leftarrow O \rightarrow B$.

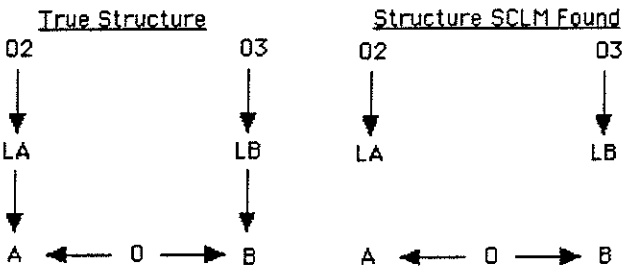


Figure 1: One of the three possible causal structures for O, A, B. L_A and L_B represent the lesions of A and B, respectively. The equations are $A = O \wedge \neg L_A$, $B = O \wedge \neg L_B$, $L_A = O_2$, and $L_B = O_3$.

II. The second study tested SCLM's ability to find the structure of causal chains of arbitrary length, i.e., $O \rightarrow X_1 \rightarrow X_2 \rightarrow \ldots \rightarrow X_n$. For lengths n from 2 to 15, SCLM invariably found the correct structure for the principle variables, although it did not find all of the causal structure involving the lesions of these variables and the lesion operators.

SCLM is therefore correct, but not complete, for the structures in the first two studies. In all cases, it successfully found the central, backbone structure of the mechanism. In some cases it also correctly identified the relation between each lesion operator and its corresponding lesion, e.g., $O_3 \rightarrow L_B$ in Figure 1. It did not find the causal relation between each lesion and the state variable of which it was the lesion, e.g., $L_B \rightarrow B$.

III. The third study applied the program to tree-structured causal mechanisms, in which a cause could have more than one direct effect. SCLM-1 could not reliably determine the structure of these trees, because it would sometimes choose an inappropriate le-

sion operator. The definition of "lesion" needs to be refined to prevent this. The problem is that "lesion" which is relevant to a variable Y must mean damaging, or at least doing something to, the thing that Y is about—the "subject" of Y. For example, pulling the plug or flipping the switch could not be lesions of the light because their subject is not the light. But it impossible to represent "the subject of Y" when the representation language has only atomic variables; one needs a subject-predicate notation. The third study shows that SCLM-1 is incorrect for causal trees, and suggests that the knowledge representation used in SCLM-1, using only graphs and propositional variables, is inadequate for trees.

Analysis shows the main action of the program has a worst-case time, measured in experiments executed and/or looked-up, of $O(N^3M^2)$, where N = number of operators and M = number of state variables. Performance measurements show that the number of experiments actually performed, for the causal chains in Study II, is just $2N$.

5 RELATED WORK

There are three possible approaches to causal knowledge acquisition: *a priori*, as in CYC [5]; passive observation, either statistical (IC, SGS, etc.) or non-statistical [8]; and experimental.

Statistical methods are useful, but not sufficient by themselves. Some readers have thought that the Lesion Principle resembled the "screening off" rule used in IC and SGS, but the two methods yield quite different results. Figure 2 compares the results of IC and SCLM-1 for a typical deterministic causal chain. The variables of primary interest are O_1, A, B, and C. Auxiliary variables L_A, L_B, and L_C represent the lesion of A, B, and C, respectively, and are controlled by operators O_2, O_3, and O_4. The equations are $A = O_1 \wedge \neg L_A$, $B = A \wedge \neg L_B$, $C = B \wedge \neg L_C$, $L_A = O_2$, $L_B = O_3$, $L_C = O_4$. The default values for all operators are zeroes. SCLM infers six of the nine directed links correctly.[1] IC infers four links but cannot decide their direction; crucially, the links between A, B, and C are missing. Neither SCLM nor IC infers any spurious links in a chain structure. The results are strikingly different. The contrast shows that any resemblance between the Lesion Principle and the screening-off rule is superficial.

The experimental work of Żytkow was discussed above. The Random Operator Principle identifies the first causes with Żytkow's "manipulation procedures" themselves, rather than the variables which they manipulate. Most other experimental approaches to causal knowledge are content to identify the effects of operators without attempting to find the order among these effects. [2, 10, 12]

An exception is May's Causal Discovery Program (CDP), which, based on Mackie's philosophical analysis of causation and Mill's Methods, uses experiment to systematically derive causal explanations [4, 6]. CDP's inputs include a phenomenon to be explained (i.e., an effect) and a list of potential causal factors. Like CLM, its experiments involve inhibiting certain variables; the variables CDP inhibits include alternative causes, as well as intervening variables (May, personal communication). CDP does not make CLM's assumption of simple causal structure; for example, it can accept causal cycles. However, it requires an assumption of "causal homogeneity," which must be justified by (usually) domain-specific knowledge. SCLM implicitly establishes homogeneity, without resorting to any domain-specific knowledge beyond the observable frequencies, by assigning default values to operators that are not being systematically varied. Another difference is that CDP requires its input to distinguish between one state variable which is an effect and others which are potential causes.

The most crucial difference between the two systems is that CDP assumes *any* of the input factors can be controlled as an independent variable. That is not always realistic; e.g., you cannot control the light's being on directly, but only by means of the switch. More importantly for the purpose of establishing the foundations of causal knowledge, CDP gives no answer to the question of how we can know which of two variables is independent, if they both vary together. It simply assumes that both may be made independent, at the experimenter's will.

These differences make CDP more robust for complex causal structures but also more knowledge-dependent. It seems appropriate for a certain stage of scientific inquiry, whereas CLM is appropriate for an earlier stage, at the level of acquiring common sense.

6 CONCLUSIONS AND FUTURE WORK

CLM is an effective method for discovering the structure of deterministic causal chains. Additional research is planned to investigate its effectiveness with indeterministic mechanisms. A subject-predicate knowledge representation is needed to extend the method to causal trees.

The current implementation, SCLM-1, has an excessively narrow criterion for deciding that two propo-

[1] If the default for O_1 is 1, SCLM is unable to infer the links $O_2 \rightarrow L_A$, $O_3 \rightarrow L_B$, and $O_3 \rightarrow L_C$, but still infers the main chain correctly.

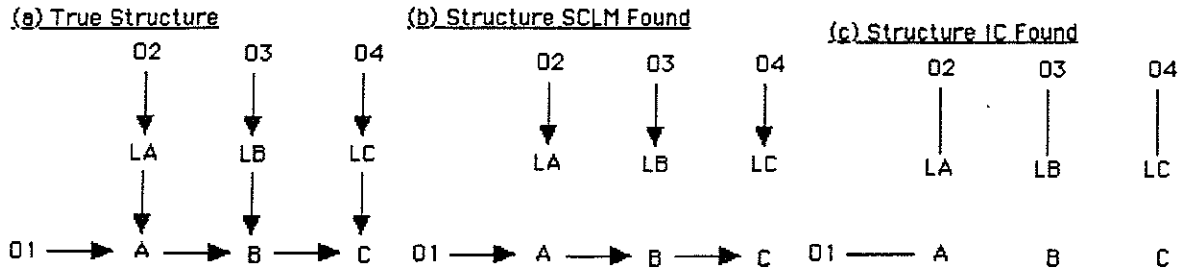


Figure 2: (a) A causal chain of length 3. (b) SCLM infers six of the nine directed links correctly. (c) IC infers only four links, without direction.

sitional variables are associated: they must either have the same value 100% of the time ($X \equiv Y$), or have opposite values 100% of the time ($X \equiv \neg Y$). This criterion can be broadened by defining X and Y to be associated if any one of the following relations holds at least p% of the time, for suitable p: $X \supset Y$, $X \supset \neg Y$, $Y \supset X$, $Y \supset \neg X$. (Cf. the use of contingency tables in 49er [7].) Further research is planned to test SCLM, with a broader criterion of association, in indeterministic environments.

As previously discussed, a subject-predicate knowledge representation is needed to properly define the concept of "lesion." Such a representation could also support the formulation of additional principles—for example, "If the experimenter is grasping X, and the experimenter's hand moves to P, and X moves to P, then the experimenter directly causes the movement of X to P." Such a rule would eliminate the need for lesions at the point of contact, e.g., taping the switch. But this rule requires subtle qualification. It is not necessarily true if X is part of the experimenter's body—the foot, for example. And if we are tempted to qualify the rule by adding "and X is not part of the experimenter," then we ought to think long and hard how the experimenter is to distinguish between itself and the external world.

Acknowledgments. I am grateful to Michael May, Tom Osgood, Vandana Rao, and several anonymous reviewers for discussion and for comments on earlier versions of this paper. Todd Shrider assisted with the experiments. This research was supported by an Indiana University East Summer Faculty Fellowship.

References

[1] G. E. M. Anscombe. Causality and determination. In Ernest Sosa and Michael Tooley, editors, *Causation.* Oxford University Press, 1993.

[2] Yolanda Gil. A domain-independent framework for effective experimentation in planning. In *Proceedings of the Eighth International Workshop on Machine Learning*, 1991.

[3] Clark Glymour, Peter Spirtes, and Richard Scheines. Causal inference. *Erkenntnis*, 35:151–189, 1991.

[4] Gerd Grasshof and Michael May. From historical case studies to systematic methods of discovery. In *Systematic Methods of Scientific Discovery: Papers from the 1995 AAAI Symposium*, pages 46–57, Stanford University, March 1995. AAAI Press. Technical Report SS-95-03.

[5] R. V. Guha and Douglas B. Lenat. CYC: a midterm report. *A. I. Magazine*, 11:33–59, 1990.

[6] J. L. Mackie. *The Cement of the Universe: A Study of Causation.* Oxford University Press, 1974.

[7] Ireneusz R. Moraczewski, Robert Zembowicz, and Jan M. Żytkow. Geobotanical database exploration. In *Systematic Methods of Scientific Discovery: Papers from the 1995 AAAI Symposium*, pages 76–80, Stanford University, March 1995. AAAI Press. Technical Report SS-95-03.

[8] Michael Pazzani. A computational theory of learning causal relationships. *Cognitive Science*, 15:401–424, 1991.

[9] Judea Pearl and T. S. Verma. A theory of inferred causation. In *Proceedings of KR-91*, pages 441–452. Morgan Kaufmann, 1991.

[10] P. D. Scott and S. Markovitch. Learning novel domains through curiosity and conjecture. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 669–674, 1989.

[11] Michael Scriven. The logic of cause. *Theory and Decision*, 2:49–66, 1971.

[12] Wei-Min Shen and Herbert A. Simon. Rule creation and rule learning through environmental exploration. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 675–680, 1989.

[13] Gregory D. Weber. Learning causal mechanisms by experiment. EECS 599 Directed Study Report, submitted to John E. Laird, The University of Michigan, July 10 1992.

[14] Jan M. Żytkow. Integration of knowledge and method in real-world discovery. *SIGART Bulletin*, 2:179–184, 1991.

Solving Knowledge Preconditions Problem in Adaptive Planning

Jen-Lung Chiu
Microsoft Corporation
jenlc@microsoft.com

Abstract

Plan feasibility has been playing a central concept in the theory of planning. The conditions under which a plan is feasible to be carried out are the preconditions of the plan. Most planning systems solve physical preconditions problem by explicitly expressing the physical preconditions axioms for each action to be executed, but few planning systems have discussed how to solve the knowledge preconditions problem.

Reasoning about the satisfiability of the knowledge preconditions of plans requires a theory that integrates temporal reasoning and reasoning about knowledge. In order to determine whether an agent knows enough to execute a plan, we must be able to characterize what he knows at the beginning of the plan execution, and how the state of the world and knowledge of the agent changes as the result of the plan execution.

This paper will discuss how the design of an adaptive planning system CADDY resolves the knowledge preconditions problem. We will present the preconditions axioms of each primitive action for CADDY and the knowledge preconditions of the control-structure operators in informal terms, then discuss how CADDY preserves these axioms during plans construction, thus solving the plan feasibility problem.

1 INTRODUCTION

This paper will discuss how the design of an adaptive planning system CADDY [4] resolves the knowledge preconditions problem. We will present the preconditions axioms of each primitive action for CADDY and the knowledge preconditions of the control-structure operators in informal terms, then discuss how CADDY preserves these axioms during plans construction, thus solving the plan feasibility problem.

One central concept in the theory of planning is the feasibility of a plan being executed. The conditions under which a plan is feasible to be carried out are the *preconditions* of the plan. Preconditions can be divided into two categories:

- *Physical preconditions* deal with what should be true in the physical world for the planning agent to execute the desired plan. If the physical preconditions of a plan is not satisfied at a situation, the plan cannot be executed.
- *Knowledge preconditions* deal with what the agent has to know in order to execute the plan. Unlike physical preconditions, a plan may still occur even if its knowledge preconditions is not satisfied. Consider that an agent is about to execute the plan "visit the biggest city of the United States". The agent can successfully execute the plan by going to New York City even if the agent does not know that New York City is the biggest city of the United States. In this case, this plan is not executed deliberately.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/256 ©1996 FLAIRS

Most AI planning systems consider only physical preconditions, not knowledge preconditions. In STRIPS [7] and TWEAK [3], each primitive action expresses explicitly the physical preconditions for the action to be executed. They are thus effectively limited to problems in which the planner has all the relevant knowledge from the start.

Reasoning about the satisfiability of the knowledge preconditions of plans requires a theory that integrates temporal reasoning and reasoning about knowledge. In order to determine whether an agent knows enough to execute a plan, we must be able to characterize what he knows at the beginning of the plan execution, and how the state of the world and knowledge of the agent changes as the result of the plan execution.

McCarthy and Hayes [8] were the first to address the Knowledge Preconditions Problem. The first extensive discussion of knowledge preconditions for actions was that of Moore [9, 10]. Moore presented a theory integrating knowledge and actions by using a modal language and a language of possible worlds. A modification and extension of Moore's theory was proposed by Morgenstern [11, 12]. Morgenstern built her theory on a first order logic in which knowledge was represented as a relation on strings. Davis [6] proposes a simple formalism to solve the knowledge preconditions problem for plans. The formalism uses the situation calculus with branching time feature as the temporal model. In [5], Davis extends the above formalism to a continuous model of time.

The paper will focus on the discussion on how CADDY solves the knowledge preconditions problem. Section 2 gives an overview of CADDY and discuss the functionalities of each module. Section 3 presents the knowledge preconditions problem in CADDY; that is, the knowledge preconditions axioms that must be preserved during plan execution. Section 4 discusses how CADDY preserves the knowledge preconditions between two consecutive steps in the constructed plan during plan generation, thus solving the problem. Finally, section 5 gives the conclusion.

2 CADDY OVERVIEW

2.1 The Golf World Testbed

The initial prototype system of CADDY operates in a testbed domain of a mediocre golfer playing golf on a foggy day. This domain was chosen to meet the following criteria:

1. Perception is partial; action is probabilistic; and the world is continuous.
2. The causal structure can be characterized by a collection of real-valued parameters. Many different types of plans are optimal under different settings of the parameters.
3. The world is designed to be simple, subject to objectives (1) and (2).

The golf course is a uniform plane; there are no roughs or traps. The golf hole, by convention, is a circle of radius ϵ around the origin, ϵ being a physical parameter. The physical state of the world at any moment consists of a triple of points: the current position of the ball, the current position of the golfer, and the position of the ball at the last stroke (used when a lost ball is replaced). Since the world is invariant under rotation about the origin, it has five real degrees of freedom.

There are five types of actions: hitting, moving, looking for the ball, scanning for the ball while moving on a path, and replacing a lost ball. The actions are characterized in terms of their preconditions, their effects on the physical world, their effect on the epistemic state of the agent, and their cost.

- (hit X θ Z ψ). X and θ are input parameters while Z and ψ are output parameters.
- (move X θ) where X and θ are input parameters.
- (look Z ψ) where Z and ψ are output parameters.
- (scan $PATH$ Z ψ) where $PATH$ is input parameter while Z and ψ are output parameters.
- (replace)

look is a pure sensing action (an action that provide information of the physical environment); *move* and *replace* are effect actions (actions that change the state of the physical environment); while *scan* and *hit* involve both sensing and physical changes. The planner can perfectly predict the effects of actions *move* and *replace*. It cannot perfectly predict the effects of *look* and *scan*, due to limited perception; nor the effect of *hit*, due to limited perception and imperfect physical control.

CADDY's knowledge of the world varies over time. It always knows the position of the golfer, and the position of the golf ball at the last hit (the position to which the ball would be replaced.) Its knowledge of the current position of the golf ball is in one of three conditions:

- Initially, and between any successful *look* or *scan* and the subsequent *hit*, CADDY knows the exact location of the ball.
- Immediately after a *hit*, CADDY knows a probability distribution for the position of the ball, which is a product of a Gaussian over distance from the hit point times a Gaussian over direction from the hit point. If the distance argument to the hit is X and the perceived distance is P, then the distribution of the true distance is a Gaussian centered at $(\beta_1^2 P + \beta_2^2 X)/(\beta_1^2 + \beta_2^2)$ with standard deviation $\beta_1\beta_2/\sqrt{(\beta_1^2 + \beta_2^2)}$
- The effect of an unsuccessful *look* or *scan* is to carve the region perceived — a circle or strip of radius λ — out of the probability distribution, and to raise the probability in the remaining region by renormalization. After a series of unsuccessful *looks* and *scans*, therefore, the probability distribution is the original Gaussian with a collection of holes poked out of it.

It is evident that any reasonable plan for CADDY has the following general form: First, you hit the ball toward the hole. Then you search for it, using some combination of looks and scans, until either you see it or you give up. If you see it, you move straight to it; if you give up, you replace it at the last hit point. Then hit it again, and iterate.

The variations in plan therefore rest in the strategies for deciding how far to hit the ball; how to search for it; and when to give up and replace it. The formulation of these strategies depends on the various world parameters. For example, if the hole is large, hitting is accurate, and replacement is cheaper than searching, then the optimal strategy could be to keep replacing the ball at the starting point and trying until you hit the ball directly from the start to the hole. If searching is expensive as compared to hits, and control is poor then it is advisable for the golfer never to hit the ball further than he can see it. If searching is expensive and control is fairly good, then the golfer should hit it far enough that the uncertainty in position is comparable to the visibility distance. If the control in direction is good but control in distance is poor, so that the region of uncertainty is long and thin, then the golfer may wish to scan on a linear path. If the region of uncertainty is both long and wide, then the golfer may wish to do a spiral search, starting from the center.

2.2 CADDY System Overview

CADDY is an adaptive planner, which works by finding a promising plan in its library, and modifying it to fit altered circumstances. The CADDY planning system is composed by the following five modules:

The Plan Library

The Plan Library (PL) consists of three tables of known plans: the Best Table (BT), which stores the best plan CADDY has actually found for particular circumstances; the Failure Table (FT), which stores the failed experiences CADDY has encountered; and the Simulation Table (ST), which records the simulation costs of each plan in PL under selected circumstances. BT and FT represent the history of CADDY's execution while ST is used as an assistance table which helps the comparison the similarity between plans and circumstances. The significance of BT, FT, and ST will be discussed later in this section.

A plan in CADDY is represented as a tree whose leaf nodes are primitive actions, and whose inner nodes are the control structures "sequence", "cond", and "while". Nodes in a plan

are annotated with comments indicating the purpose of the corresponding step.

The Estimation Module

The Estimation Module (EM) estimates the cost of a known plan under the current circumstances using information stored in ST.

In the current implementation, EM uses linear interpolation on system parameters (physical parameters α_1, β_1, λ, and cost parameter *c_replace*) and the starting situation (distance between CADDY and the hole at the beginning) to calculate the estimated cost. This is a 5-stage interpolation. In each stage, EM searches through ST, chooses two nearest values of the current attribute, and applies linear interpolation to get the estimated cost under current interpolation stage. After passing down then passing up 5 stages, EM will get the estimated cost of a specific plan under chosen circumstance. Information stored in ST is used as basis points during interpolations.

The Plan Selection Module

The objective of the Plan Selection Module (PSM) is to find the best plan in PL for the current circumstances. PSM first looks through BT to find if CADDY has met similar circumstances before. If a match is found, PSM retrieves the plan from PL. Otherwise PSM uses EM to estimate the cost of each plan in PL and passes the plan with the least estimated cost to PAM. The use of BT prevents CADDY from executing unnecessary interpolations over and over, thus helping to improve the system performance.

The Plan Adaptation Module

The Plan Adaptation Module (PAM) modifies the candidate plan output by PSM to fit the current circumstances using domain-specific heuristics. PAM begins by collecting the set of heuristics that apparently apply to the selected plan in the current circumstances. PAM then checks in FT whether this set of heuristic rules has previously been found to fail in similar circumstances. If not, PAM is safe to apply these collected heuristics to the selected plan. Otherwise PAM tries one heuristic at a time, checks whether it gives an improvement, and applies all those that give improvements. The use of FT to prevent CADDY from attempting an adaptation that has already failed is an application of failure-driven learning.

The Plan Execution Module (PEM)

PEM performs multiple (typically hundreds) of simulations of the execution of the plan generated by PAM, in order to determine its expected value. PEM includes a user interface which can demonstrate the execution of a plan on a graphic board and give helpful information and an execution log for trace purpose.

PEM stores the simulation result in PL for future uses. If the plan generated by PAM outperforms the plan selected by PSM, PEM stores the generated plan in PL, records the simulation result in BT, and invokes simulations under selected circumstances and records these results in ST so that this newly generated plan can be interpolated and selected by future runs.

But if the plan generated by PEM is outperformed by the plan selected by PSM, PEM discards the generated plan and records the failed experience in FT so that CADDY can avoid performing the same adaptation on the same selected plan under similar circumstances.

3 THE PROBLEM

For all primitive actions, we assumes that CADDY knows the values of the input parameters before executing the action, and knows the values of the output parameters after executing the action.

For (hit $X\ \theta\ Z\ \psi$), the one physical precondition required is that CADDY and the golf ball be at the same position ($(R_x, R_y) = (B_x, B_y)$). No knowledge preconditions are required. The result of executing *hit* is unpredictable by CADDY and is controlled by the underlying testbed (the values of α_1, α_2, β_1, and β_2).

For (look $Z\ \psi$) and (scan $PATH\ Z\ \psi$), no physical and knowledge preconditions are needed. The result of executing *look* or *scan* depends on the CADDY's perception capability (the value of λ).

For (move $X\ \theta$), no physical and knowledge preconditions are required and CADDY executes *move* error–free.

For (replace), no physical preconditions are required. The one knowledge precondition required is that CADDY knows the position that it most recently hits the ball (the value of (H_x, H_y)). CADDY executes *replace* error–free.

The knowledge preconditions of the control–structure operators are adopted from Morgenstern's model [11]. For (sequence $P_1\ P_2$), the knowledge preconditions require that CADDY knows how to execute sub–plan P_1 at the beginning, and knows how to execute sub–plan P_2 after P_1 is complete. For (cond $Q\ P_1\ P_2$), the knowledge preconditions require that CADDY knows the truth value of Q at the beginning, and knows either how to execute sub–plan P_1 if Q is $TRUE$ or how to execute sub–plan P_2 if Q is $FALSE$. For (while $Q\ P_1$), the knowledge preconditions are defined recursively in terms of sequences and conditionals since (while $Q\ P_1$) is equivalent to (cond Q (sequence P_1 (while $Q\ P_1$)) $\emptyset$).

It seems that the epistemic feasibility problem is rather simple in CADDY, since only *replace* requires knowledge preconditions. The problem with this "local view" is that CADDY does not consider the causal structure (hit, search, then move or replace loops) of the plans. We must extend the "local view" to a "global view" of the whole plan structure. Several conditions must be preserved during plan generation so that the generated plans are feasible for later execution.

S_1 : In order to *hit* the ball, CADDY must *move* to the the same position where the ball resides or *replace* the ball. This requires that every *hit* must be preceded by either a *move* to the position of the ball or a *replace*.

S_2 : In order to *move* to the position of the ball, CADDY must know the position. This requires that a successful *look* or *scan* must have taken place.

S_3 : In order to *replace* the ball, CADDY must know the position where it most recently *hit*s the ball. Condition S_3 can be easily achieved since CADDY can store the required position into its internal knowledge each time it *hit*s the ball.

In this "global view" of CADDY, a plan is epistemically feasible if the following two conditions are satisfied:

Q_1 : When CADDY is about to execute a primitive action, the preconditions of the primitive action is satisfied.

Q_2 : Conditions S_1, S_2, S_3 are preserved during plan execution. Since S_3 can be easily preserved with extra internal storage, we focus on S_1 and S_2.

We now discuss how CADDY preserves the feasibility conditions during plan generation (the execution of the Plan Selection Module followed by the execution of the Plan Adaptation Module).

4 THE SOLUTION

For planning systems which assume complete knowledge, the plan feasibility problem reduces to the physical preconditions problems. The agents always have complete and error–free knowledge of the underlying environment, all knowledge preconditions axioms will be satisfied. Classical planning systems like STRIPS and TWEAK are within this category.

For planning systems which interleave planning and acting, the plan feasibility problem is implicitly solved by the planning stage. At each planning stage, the planning systems only consider those actions which are feasible for the agent. That is, the planning system only considers those actions whose physical preconditions and knowledge preconditions are satisfied at the time the planning systems are about to generate next steps. POMDPs [1, 2] are within this category.

CADDY operates in an unknown environment and has limited perceptions, and CADDY will not take any execution until the plan has been generated. CADDY does not fall into either of the above categories, CADDY must have ways to solve the feasibility problem. Instead of constructing plans from scratch, CADDY adapts experiences. To solve the feasibility problem, CADDY must take care of the following two issues:

- The plans which are pre–store in the Plan Library satisfy the requirements of preconditions.
- The heuristic rules which are used to adapt the selected plans must maintain the requirements of preconditions.

In the simplest execution, CADDY just selects a plan from the Plan Library and executes it. In order to solve the plan feasibility problem, the plans which reside in the Plan Library must satisfy the requirement of physical preconditions as well as knowledge preconditions. The plans initially in the Plan Library are constructed with care so that CADDY is feasible to execute either of these plans if it is selected. That is, condition Q_1 and Q_2 are preserved during the execution of every plan stored in the Plan Library.

The plans stored by the Plan Execution Module are plans modified by the Plan Adaptation Module. After the Plan Selection Module chooses a plan, the Plan Adaptation Module modifies the plan by applying the heuristic rules. These modifications change the internal structure of the plans and have potential to destroy the feasibility of the plans, CADDY should apply these rules with care so that conditions Q_1 and Q_2 are still preserved even the plan structure is altered.

Applying the heuristic rules for adapting *hit* (**H1** and **H2**) only changes the X parameter of the primitive action *hit*, but the internal structure of the plan remains the same. *Hit* does not require knowledge preconditions and the physical preconditions are still satisfied (the adaptation rules do not alter the ball's position or CADDY's position before the *hit*), Q_1 is preserved. Q_2 is also preserved since changing the hitting distance of *hit* does not alter the causal structure of the plan, S_1 and S_2 are unaffected.

Applying the heuristic rules for adapting *replace* (**R1**, **R2**, and **R3**) changes the internal structure of the plan by adding a *cond* plan segment before searching the ball. The plan segment tests some condition and executes *replace* if the condition is $TRUE$, otherwise resumes the search execution. Action *replace* does not require physical preconditions and the knowledge required can be extracted from the instantaneous situation (the (H_x, H_y) point in each situation); the condition testing does not change anything if the condition is $FALSE$, so the preconditions needed for the first action of the search sub–plan are still satisfied. Q_1 is preserved. The insertion of the plan segment changes the causal structure of the plan. The new flow (*hit–replace–hit*) preserves condition S_1 and S_2 (*hit* is preceded by a *replace*, no *move* is involved in the causal flow), the original flows are known to preserve conditions S_1 and S_2. Q_2 is also preserved.

Applying the heuristic rules for preventing *replace* (**S1**) changes the internal structure by adding a *scan* action before *replace* action to extend the search. Since *scan* does not require physical preconditions and knowledge preconditions, *replace* does not require physical preconditions and the knowledge required can be extracted from the instantaneous situation (the (H_x, H_y) point), Q_1 is preserved. The extra search step added into the plan changes the causal structure of the plan. The added *scan* action can be viewed as s search flows extension, and the global causal structure remains the same. Q_2 is also preserved.

The plans initially in the Plan Library are feasible to be executed, and the heuristic adaptation rules do not destroy the feasibility of the plans. This concludes that the plan which CADDY is about to execute will not cause any feasibility problem.

5 CONCLUSION

The paper presents the "local view" and "global view" of the knowledge preconditions axioms for CADDY under the golf world problem domain, then shows how these knowledge preconditions axioms are preserved during the execution of the modules in CADDY. This proves that the generated plans which CADDY is about to executed are always epistemically feasible.

The study of plan feasibility problems in CADDY, a portotype planning system, gives us the understanding of one central key issue of designing a problem solver; that is, to preserve the satisfiability of the preconditions, especially knowledge preconditions. If we cannot guarantee the generated plans are feasible to be executed, the generated plans are unlikely to deliberately solve the problems.

References

[1] Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting Optimally in Partially Observable Stochastic Domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, Washington, U.S.A., July 31 - August 4 1994. American Association for Artificial Intelligence.

[2] Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Algorithms for Partially Observable Markov Decision Processes. Technical Report CS–94–14, Department of Computer Science, Brown University, 1994.

[3] David Chapman. Planning for Conjunctive Goals. *Artificial Intelligence*, 32(3):333–377, 1987.

[4] Jen-Lung Chiu. *Planning in Imperfect World Using Previous Experiences.* PhD thesis, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, May 1995.

[5] Ernest S. Davis. Branching Continuous Time and the Semantics of Continuous Action. In Kristian J. Hammond, editor, *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems*, pages 231–236, Chicago, Illinois, U.S.A., June 13 - 15 1994. Poster.

[6] Ernest S. Davis. Knowledge Preconditions for Plans. *Journal of Logic and Computation*, Forthcoming.

[7] Richard E. Fikes and Nils J. Nilsson. STRIPS : A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3):189–208, 1971.

[8] John McCarthy and Patrick J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press., Edinburgh, United Kingdom, 1969.

[9] Robert C. Moore. Reasoning about Knowledge and Action. Technical Report 191, SRI International, Menlo Park, California, U.S.A., 1980.

[10] Robert C. Moore. A Formal Theory of Knowledge and Action. In Jerry Hobbs and Robert C. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358. ABLEX Publishing, Norwood, New Jersey, U.S.A., 1985.

[11] Leora Morgenstern. Knowledge Preconditions for Actions and Plans. In *Proceedings of the Tenth Annual International Joint Conference on Artificial Intelligence*, pages 867–874, Milan, Italy, August 23 - 28 1987.

[12] Leora Morgenstern. *Foundations of a Logic of Knowledge, Action, and Communication.* PhD thesis, Courant Institute of Mathematical Sciences, New York University, 1988.

HIGH SPEED SIMILITUDE RETRIEVAL FOR A VIEWPOINT-BASED SIMILARITY DISCRIMINATION SYSTEM

Takashi Yukawa, Kaname Kasahara and Kazumitsu Matsuzawa
NTT Communication Science Laboratories
{yukawa,kaname,matsu}@nttkb.ntt.jp

Abstract

This paper proposes high-speed similitude retrieval schemes for a viewpoint-based similarity discrimination system (VB-SDS) and presents analytical and experimental performance evaluations. The VB-SDS, which contains a huge set of semantic definitions of commonly used words and computes semantic similarity between any two words under a certain viewpoint, promises to be a very important module in analogical and case-based reasoning systems that provide solutions under uncertainty. By computing and comparing similarities for all words contained in the system, the most similar word for a given word can be retrieved under a given viewpoint. However, the time this consumes makes the VB-SDS unsuitable for inference systems. The proposed schemes reduce search space based on the upper bound of a similarity calculation function to increase retrieval speed. An analytical evaluation shows the schemes can achieve a thousand-fold speedup and confirmed through experimental results for a VB-SDS containing about 40,000 words.

1 INTRODUCTION

Semantic similarity discrimination between linguistic words is important in knowledge and data processing such as conceptual information retrieval, user-friendly query interfaces on databases [1], analogical reasoning [2] and case-based reasoning [3]. Since semantic similarity varies depending on the situation and context, a similarity discrimination system must be able to recognize both. The situation and context are considered as viewpoints and a viewpoint-based similarity discrimination system (VB-SDS) has been proposed [4]. The VB-SDS measures the similarity between two linguistic words under various viewpoints. It contains a huge set of semantic definitions of daily used words and computes the similarity between a pair of words in the system under any viewpoint, which is also a word in the system.

Similitude retrieval is the retrieval of a word in the VB-SDS that is closest to a given word under a given viewpoint. In analogical reasoning, an alternative rule is determined based on similitude when an appropriate rule cannot be found, and in an intelligent database query interface, an alternative keyword is determined based on similitude when the entry matching a query cannot be found. Thus, it can be said that similitude retrieval plays an important role in these systems. Basically, computing and comparing similarity for all words in the system provides similitude retrieval. However, the large number of words contained in the VB-SDS increases the computation time. This delay must be reduced for VB-SDS to be used in database or inference systems.

This paper proposes upper-bound-based fast similitude retrieval schemes, UB-FSR and UB-FSR/R. The schemes take advantage of pre-calculated similarities and the upper bound of the similarity calculation function to reduce search space.

2 BACKGROUND

The semantic similarity between two words changes with the viewpoint. For example, "horse" is more similar to "pig" than "car" when the viewpoint is "animal" and more similar to "car" than "pig" when the viewpoint is "transportation". Thus, in addition to the words themselves, a viewpoint is required to calculate similarity between two words.

The VB-SDS represents a system that can discriminate such a similarity for commonly used words. It includes semantic definitions for several hundred

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/261 ©1996 FLAIRS

thousand of commonly used words acquired from machine-readable dictionaries [4]. On this system, the semantics of a word are defined as a set of attribute-weight pairs, where an attribute is the name of a morpheme category and each morpheme is contained in the dictionary definition of that word. The weight is the frequency that the attribute appears in the definition. Thus, a word, W_i, is represented as

$$W_i = \{(p[1], q_i[1]), \ldots, (p[n], q_i[n]), \ldots\},$$

where n is a unique number for each category, $p[n]$ is the name of a category n and $q_i[n]$ is the weight corresponding to the category.

2.1 Similarity Calculation Procedure

The calculation procedure for semantic similarity between two words, W_i and W_j, under the viewpoint W_v is as follows:

Modulation based on the viewpoint: To make the viewpoint influence similarity, each attribute of the words is emphasized based on the corresponding attribute in the viewpoint. For each attribute which is included in both word W_i and the viewpoint W_v, the weight for the attribute in the word is multiplied by C. $C(> 1)$ is the designated modulation factor and controls the strength of the influence of the viewpoint. $W_{i(v)}$ represents the representation for the modulated word.

Normalization of weights: Each weight in the modulated word representation is normalized. They are divided by the square root of the sum of the square of each weight included in the word. $W'_{i(v)}$ represents the normalized representation for the modulated word.

Calculation of similarity: The similarity $S_v(i,j)$ is defined as the inner product of the normalized representations $W'_{i(v)}$ and $W'_{j(v)}$.

Similarity calculated from the above procedures minus the modulation step means the essential similarity between the words, that is independent of viewpoints. We call this the universal similarity and represent it as $S_0(i,j)$.

2.2 Basic Similitude Retrieval

Similitude retrieval is retrieving a word that has the largest similarity with the given query word under a given viewpoint. Obviously, this can be done by computing and comparing the similarity among all words in the system. This means that a calculation time is proportional to the number of the words. Assuming the similarity calculation for one pair of words takes 1 millisecond and the system contains 40,000 words, it would take 40 seconds to find the most similar word for each query.

3 HIGH SPEED SIMILITUDE RETRIEVAL SCHEMES

We propose two high speed similitude retrieval schemes based on the upper bound of similarity. One version is called UB-FSR, which stands for upper-bound based fast similitude retrieval, and the other is UB-FSR/R where the R stands for restricted list size. These schemes reduce the search space based on the relation between the largest similarity found at a certain moment and the upper bounds.

In studying the similarity calculation function, the range of similarity variation caused by the viewpoint is found to be limited and can be determined from the universal similarity, although the exact similarity cannot be known until the words and the viewpoint are given. The upper-bound function for the similarity function is expressed as

$$U(i,j,C) = S_0(i,j)\frac{1+(C^2-1)}{1+(C^2-1)S_0(i,j)}.$$

The detailed formulation of this function is described in appendix A.

For each word, it is possible to calculate the universal similarities between that word and any other word when the VB-SDS is constructed.

3.1 UB-FSR

Assume a query word W_i and a viewpoint W_v. The scheme prepares a list of pairs of word W_j and the universal similarity $S_0(i,j)$ for any j sorted with the descending order of S_0. The list is generated when the VB-SDS is constructed.

Denote the word having the k-th largest universal similarity $W_{J_i(K)}$. Obviously, the universal similarity for the word is expressed as $S_0(i, J_i(k))$. The retrieval procedure on UB-FSR is as follows:

1. Initialize a variable $S_{\max}$ that holds the largest similarity to 0, a variable $W_{\max}$ that holds the word which provides the largest similarity to null and extraction pointer k to 1.
2. Extract a word $W_{J_i(k)}$ and the universal similarity $S_0(i, J_i(k))$.
3. Calculate the upper bound $U(i, J_i(K), C)$ based on $S_0(i, J_i(k))$.

4. If the upper bound $U(i, J_i(k), C)$ is less than $S_{\max}$, terminate the procedure and output $W_{\max}$ as the most similar word.

5. Calculate the similarity S between W_i and $W_{J_i(k)}$ under the viewpoint W_v.

6. If S is greater than the largest similarity at the moment $S_{\max}$, store S to $S_{\max}$ and $W_{J_i(K)}$ to $W_{\max}$.

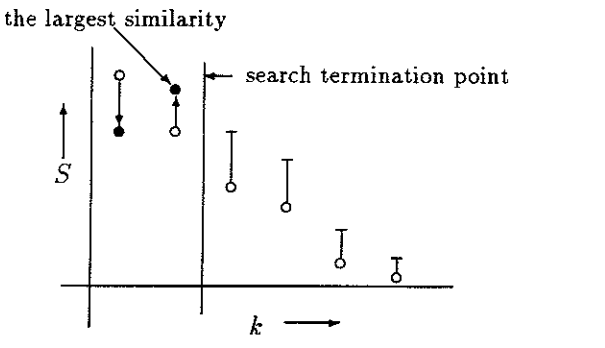
7. Increment k and go to step 2.

The execution of the procedure is illustrated in Figure 1.

3.2 UB-FSR/R

For UB-FSR, the number of elements in the universal similarity list is obviously the same as the number of words in the system and one list is required for each word. Thus, the size of the lists grows with the square of the number of words. Since the VB-SDS contains commonly used words, the number of which reaches several hundred thousand, the lists become too huge. UB-FSR/R is an identical scheme to UB-FSR except that it restricts the number of elements in the list to several hundred.

In studying similarity distribution of the VB-SDS, we found there are many words that provide the universal similarity as 0. These words can not be the most similar word because they also provide the similarity as 0 under any viewpoint. In addition, the possibility is also small that a word providing a small universal similarity is the most similar word, because of the monotony of the upper bound of the similarity. According to this discussion, UB-FSR/R can be achieved without a significant declination of the performance.



k : the rank in the universal similarity list
S : similarity • : similarity under the viewpoint
o : universal similarity ⊤ : upper bound of the similarity

Figure 1: Reduction of search space

The retrieval procedure for UB-FSR/R is almost the same as UB-FSR. The difference is that it switches over to the basic retrieval procedure if the terminative condition is not satisfied when all words in the list are extracted.

4 PERFORMANCE EVALUATION

4.1 Analytical Estimation

Since the schemes aim for fast retrieval, performance is measured by how much a speed is increased compared to the basic retrieval scheme. The speedup ratio V is defined as $V = M/M_X$, where M is the number of words in the VB-SDS and M_X is the average number of words that must be examined to determine the most similar word.

To obtain analytical results, at first, attributes which have non-zero weight are assumed to occur randomly for a word. This assumption enables us to estimate the frequency distribution of the universal similarity, which implies probability density that a word provides certain value as the universal similarity. M_X is known based on the the estimated frequency distribution. Then the speedup ratio is calculated using M_X.

Frequency distribution of the universal similarity: The universal similarity between a pair of words is according to the number of attributes, where the weight is non-zero in both words. Therefore, the frequency of the universal similarity arising from the assumption has an almost exponential distribution. The number of words that provides the universal similarity from S_a to S_b is computed by

$$F_M(S_a, S_b) = M \int_{S_a}^{S_b} \frac{1}{\lambda} e^{-\frac{1}{\lambda}x} dx,$$

where λ is determined based on the average number of attributes with non-zero weight in one of the words and is the same as the average of the universal similarity. An observation of the real VB-SDS obtained that λ is approximately 0.05.

The number of words that must be examined(M_X): Assuming the most similar word has the similarity S_x, the retrieval procedure is terminated when the upper bound for a word becomes smaller than S_x. Thus, only words that provide a larger upper bound than S_x are examined. Referencing the upper-bound function, the universal similarity that provides upper bound U is

$$S_0(U) = \frac{U}{1 + (1 - U)(C^2 - 1)}.$$

The number of examined words, $M_X(U)$, is given by integrating the distribution function form $x = S_0(U)$ to 1. That is, $M_X(U) = F_M(S_0(U), 1)$.

Speedup ratio, V: According to the above discussion, dividing M by $M_X(S_x)$ gives the speedup ratio for which similarity of the most similar word is S_x.

Figure 2 shows the estimated speedup ratio for UB-FSR with a modulation factor from 1.2 to 5. From Fig. 2, it can be seen that the scheme achieves a more than 1000-fold speedup for $S_x > 0.8$ and $C < 3$. Since the most similar word is a word that provides the largest similarity, the similarity for it is expected to be a large value, i.e., greater than 0.8. In addition, an analysis of results obtained by opinion tests concludes that an appropriate modulation factor lies in a range of 1.5 to 2.5 for the VB-SDS. Therefore, it can be said that the UB-FSR scheme achieves excellent performance for practical ranges of similarity for the most similar word and a modulation factor.

The speedup ratio for UB-FSR/R is the same as that for UB-FSR except that there is a cut-off point caused by the incompleteness of the universal similarity list. The terminative condition is not satisfied when the similarity of the most similar word, S_x, is smaller than the least upper bound among those for all words in the list, U_L. This implies that if S_x is smaller than U_L, the retrieval procedure is switched to the basic retrieval scheme and no speed increase can be obtained.

U_L is calculated from the least universal similarity in the list, S_{0L}, as

$$U_L = S_{0L} \frac{1 + (C^2 - 1)}{1 + (C^2 - 1)S_{0L}}.$$

The relation between S_{0L} and the number of elements in the list N_L is

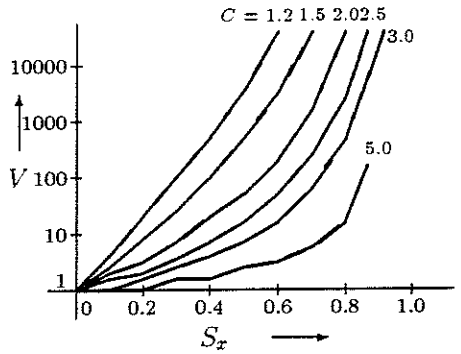
$$N_L = M \int_{S_{0L}}^{1} \frac{1}{\lambda} e^{-\frac{1}{\lambda}x} dx.$$

The latter expression enables us to calculate S_{0L} from N_L. When $S_x > U_L$, the speedup ratio is exactly the same as that for UB-FSR and when $S_x \leq U_L$, the speedup ratio is 1. Figure 3 shows the estimated speedup ratio of the UB-FSR/R for a practical range of modulation factors when $N_L = 100$. Since the cut-off point is smaller than 0.8, the restriction of the number of elements in the list causes no speed loss for the area on which we focus.

4.2 Experimental Results

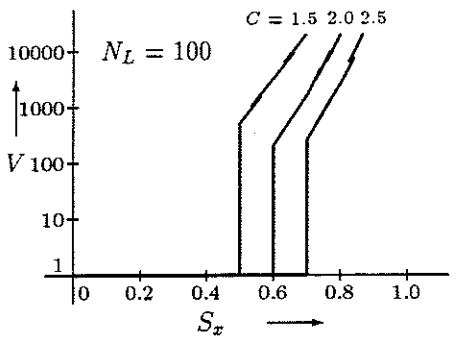
Experimental results obtained with the real VB-SDS are presented in this section. The VB-SDS contains 40,000 commonly used words and the universal similarity list comprises 100 elements for each word. The experimental system retrieves the most similar word for a given query word and viewpoint based on UB-FSR/R and reports the word itself, the similarity for it and the number of words examined until the word is found. The speedup ratio is calculated from the number of examined words.

In the experiment, retrieval was tried for 50 randomly selected words. Figure 4 shows the results. Each dot gives a speedup ratio and the similarity of the most similar word for each trial. The estimated speedup ratio is also shown by the solid line for reference. The figure shows a thousand-fold speedup for almost all trials. In addition, the cut-off point obtained experimentally agrees with the estimation.



V : Speedup ratio C : Modulation factor
S_x : Similarity for the most similar word

Figure 2: Estimated speedup ratio for UB-FSR



V : Speedup ratio C : Modulation factor
S_x : Similarity for the most similar word

Figure 3: Estimated speedup ratio for UB-FSR/R

5 CONCLUSION

This paper has proposed high speed similitude retrieval schemes, UB-FSR and UB-FSR/R, for a VB-SDS. The schemes reduce the search space by taking advantage of the property of the upper bound of the similarity, where the upper bound is calculated based on the universal similarity. Since the word with a smaller upper bound than the largest similarity already found cannot be the most similar word, it is possible to omit wasteful similarity calculations by preparing a sorted list of the universal similarity, searching from the top of the list and terminating the search when the upper bound becomes smaller than the largest current similarity. UB-FSR is the primary scheme for achieving the above principle. In UB-FSR/R, the size of the universal similarity list is restricted, but this does not significantly affect performance impairment.

Analytical and experimental performance evaluations of the schemes were also presented. The analytical results predict a more than a thousand-fold speedup for the most similar word when the similarity is more than 0.8. To prove the performance estimate, an experimental VB-SDS containing a 40,000 commonly used word system based on UB-FSR/R was constructed, and the speedup ratio was measured for 50 randomly selected words. The results show that, as estimated, the scheme achieves a thousand-fold speedup.

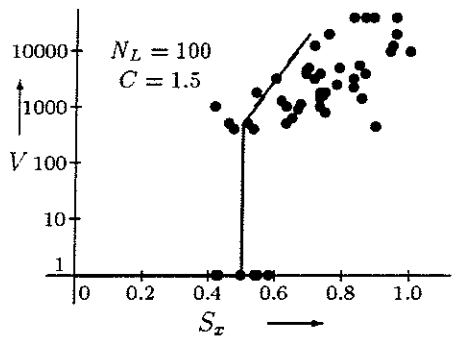
References

[1] A. Motro, "Intentional Answers to Database Queries," *IEEE Trans. Knowledge and Data Engineering*, Vol. 6 No. 3, pp. 444–454, 1994.

[2] P. H. Winston, "Learning and Reasoning by Analogy," *Communications of the ACM*, Vol. 23 No. 12, pp. 689 –703, 1980.

[3] K. J. Hammond, "CHEF: A Model of Case-Based Planning," *Proc. AAAI*, pp. 267–271, 1986.

[4] K. Kasahara, K. Matsuzawa, T. Ishikawa and T. Kawaoka, "Viewpoint-based measurement of semantic similarity between words," *Proc. Int. Workshop on AI and Stat.*, pp. 292–302, 1995.



V : Speedup ratio — : Estimated speedup ratio
S_x : Similarity for the most similar word

Figure 4: Experimental results for UB-FSR/R

A Formulation of the Upper-Bound Function

Define α as

$$\alpha = C^2 - 1,$$

a set D comprised of indexes that the weight for the viewpoint is non-zero as

$$D = \{d | q_v[d] \neq 0\},$$

and $R(x, y)$ as

$$R(x,y) = \sum_{d \in D} q_x[d] q_y[d] \Big/ \sum_n q_x[n] q_y[n] .$$

Using these notations, the similarity between words W_i and W_j under a viewpoint W_v is represented as

$$S_v(i,j) = S_0(i,j) \frac{1 + \alpha R(i,j)}{\sqrt{1 + \alpha R(i,i)}\sqrt{1 + \alpha R(j,j)}},$$

where $S_0(i,j)$ is the universal similarity between words W_i and W_j.

According to the property of the inner product, it is true that

$$\sum_{d \in D} q_i[d]^2 \sum_{d \in D} q_j[d]^2 \geq \left(\sum_{d \in D} q_i[d] q_j[d] \right)^2 ,$$

then the following inequality is carried.

$$R(j,j) \geq R(i,j)^2 S_0(i,j)^2 / R(i,i)^2 .$$

Therefore, the similarity is expressed as

$$S_v(i,j)^2 \leq \frac{(1 + \alpha R(i,j))^2 S_0(i,j)^2}{(1 + \alpha R(i,i))(1 + \alpha R(i,j)^2 S_0(i,j)^2 / R(i,i)^2)}.$$

Computing the relative maximum by differentiating the above expression by $R(i,j)$ and $R(i,i)$ gives the upper-bound function.

TOWARDS COMPETENCE ASSESSMENT FOR INTELLIGENT SYSTEMS

Klaus P. Jantke
Hochschule für Technik, Wirtschaft und Kultur Leipzig (FH)
jantke@informatik.th-leipzig.de

Abstract

The present paper deals with the problem of reflective behaviour in artificial intelligence. An intelligent system is said to enjoy the ability of reflection, to some degree, if it has some ability to evaluate the quality of its own behaviour. So far, investigations in this area were exclusively focused on metalevel architectures. In this paper, reflection is understood as a problem of a system's behaviour. This does not directly address issues of the system's architecture. Related work is surveyed. A reflective system may reason about several aspects of its own behaviour like competence, efficiency, certainty and so on. This paper puts emphasis on a system's competence, i.e. its ability to solve a problem it is faced to. The concept of reflection types is introduced. This is a key concept for formalizing a system's ability to ponder about the limits of its competence. First, several reflection types are formalised on a domain-independent level. These approaches are intuitively different. Second, these approaches are applied to the domain of learning recursive functions. There is demonstrated the impact of slightly different concepts of reflection on the power and limitations of reflective systems. Under very weak assumptions, there is no need to distinguish among several types of reflection. Under other circumstances, the choice of a reflection type may turn out to be important, as stronger reflection types restrict the system's power.

1 INTRODUCTION

The development of knowledge-based systems is an exciting area of research and applications. With the growing complexity of application domains for artificial intelligence (AI) systems, there is a growing need for control. But simultaneously, the growth in complexity makes AI systems less controlable. This bears abundant evidence of the need for computer support in system supervision and even for computer self-control.

More systematically, there are the following tendencies causing the need for reflection as a crucial property of new generation AI systems:

- Computer systems are mushrooming. They are penetrating into almost all areas of daily life. As a result, a huge number of naive users is facing seemingly intelligent systems. A remarkable percentage of these users is definitely unable to identify a system's power and limitations.
- The progress of technology is opening new areas of application. In particular, there is a relevant area of applications requiring autonomous computer systems. Repair actions under dangerous conditions (in nuclear power plants, for instance) and space missions are illustrative examples. Computer on duty under those conditions do need some ability of reflection, as they are left to their own devices.
- The appearance of computer systems is drastically improving. Based on new technologies like natural language processing and multimedial tools, there is some mismatch of appearance and ability. Numerous systems look much more impressive than they really are. Consequently, there is the seemingly unavoidable danger of overestimating a system's competence.

Reflection denotes someones activity of thinking about oneself as well as about one's relation to the outside world. In particular, reflecting means pondering about ones capabilities and limitations. Reasoning about ones competence is a central issue of reflective behaviour.

According to the importance of reflection for artificial intelligence, there had been launched an ESPRIT

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/266

Basic Research Action P3178 entitled REFLECT on Knowledge Level Reflection: Specifications and Architectures. The interested reader is directed to [vHABS+92] and [vHW92], in this respect. The state of the art is still unsatisfactory, and most AI systems are not reflecting at all. This bears witness to the need of research endeavours into the essentials of automating reflective behaviour.

2 OUTLINE OF THE APPROACH

The crux of investigations of reflection in AI is to find the right formalizations which, on the one hand, provide a sufficiently formal background for non-trivial results and, on the other hand, are sufficiently general for useful interpretations in practically relevant setting. Therefore, we put more emphasis the usual on motivations and on cruizing for appropriate concepts. This is particularly driven by the author's opinion that the ESPRIT basic research action REFLECT mentioned above has been completely misguided when concentrating on architectural problems. *Reflection is a phenomenon of a system's behaviour.*

The present paper will be organized according to the following issues:

- **Scenarios of Competence Assessment**

  A careful analysis of existing systems, in their right perspective, exhibits that a remarkable number of them enjoys a certain degree of reflective behaviour, implicitly.

- **Specifying Types of Reflection**

  From the preceding analysis, there are abstracted fundamental types of reflection.

- **Application to Learning Systems**

  A rigorous application of the novel concepts to learning systems will result in a score or more formal results illuminating the power and limitations of reflective behaviour.

For readability, technicalities are suppressed as much as possible. Compared to the present paper, [Jan95] is much more formal. However, we try to present results of a similar precision and relevance.

3 SCENARIOS OF COMPETENCE ASSESSMENT

Compared to traditional expert systems, reflective systems act more responsible. Different scenarios illustrate reflective systems' behaviour, characterize versions of competence assessment, and exhibit the practical exploitation of reflection.

Typically, AI systems are used interactively. They serve as intelligent assistants in several areas. Details of application scenarios vary with the application domain. But most application scenarios have certain features in common. These are essential for our investigations of reflective behaviour and competence assessment.

First of all, the use of an intelligent computer system usually proceeds as a sequence of man–machine–interactions. It may be abstracted as a sequence of input/output or action/response pairs. Characteristic intentions of the human input are to pose a particular problem to be solved, to provide data about the problem on hand, and to feed in background knowledge which may be useful or necessary for problem solving. Typical responses are requests for further information and pieces to build up a solution. In the case of complex problems and domains, single input data rarely suffice to describe a problem completely. In realistic situations, the number of input/output cycles necessary to find a solution is unknown in advance. Thus, interactive problem solving gets a certain touch of limiting behaviour, although the problem solution may be arrived at after finitely many steps.

As a consequence, individual inputs usually do not allow to derive definite statements about a system's ability to solve a current problem. Competence assessment is more a global than a local problem. It refers to a system's limiting behaviour, in a sense.

In contrast, all system outputs are generated in dependence on a finite number of inputs and based on a merely finite history of information processing. Particular system reactions may be interpreted as utterances about the system's reflection. Those local utterances need to be related to the system's global behaviour.

Basic scenarios are aimed at relating local utterances about a system's reflection and global attempts to evaluate its competence.

3.1 An Introductory Scenario

Assume any standard chess playing program of the simplest type. Its main activity is generating moves in response to moves put in. Those programs may be used to assist a user playing against someone else. Their outputs may be understood as the simplest type of intelligent assistance. Simple programs are optimistically generating outputs seemingly assuming to be able to win every match. In other words, every output may be interpreted as a pair of statements: (1) The system expresses its confidence in its ability to solve the problem on hand, i.e. to win the game. Therefore, (2) it generates a particular output

intended to be used as a building block for solving the overall problem. With the system's advice, a human user may sometimes win, sometimes loose. The system only admits to be unable to do its job successfully when checkmated.

More advanced chess playing programs are able to recognize particular positions in which the other player can definitely checkmate the system within a certain number of moves. Some of the currently existing systems are much more competent in this respect than the majority of human chess players. Those systems may be considered as more intelligent than those of the former type, as they enjoy a higher degree of reflection. Although they mostly behave self-confidently like the simpler systems discussed before, there are particular situations in which they are able to reflect about their own limitations.

Interestingly, there are particular situations where using a less sensitive system may turn out to be more successful in practice. This applies in cases where the other player is not able to recognize his advantage and to find his winning moves. A reflective system in correctly analyzing a position may resign earlier than necessary with respect to the partners abilities.

Another type of chess playing programs is able to evaluate positions on the chess board. On request, systems of that type are generating some score which – in a sense – expresses their confidence in being able to do their job correctly. Thus, systems of that type enjoy the ability to gradually express their competence. As there is not known any sufficiently correct and computationally feasible theory of chess, those scores do not definitely describe the potentials of a given position. Instead, these values express only the systems "belief" which may be correct to some extent.

To sum up briefly, in the simplest scenarios of using chess playing programs as intelligent assistants, systems' outputs may be interpreted as pairs consisting of a system's utterance about its competence to solve the problem on hand together with a particular contribution to problem solving.

The following chapters 4 and 5 will deal with formalizations and results. Before going into these details, we try to extract some essentials.

3.2 First Steps Towards Formalization

The minimal formalizations describe a given system's behaviour by a sequence of action/response pairs. An input is any input i provided by the user or by the environment in which the system is acting. A system's output consists of a conventional output denoted by o together with a system's utterance r expressing its confidence in being able to solve the present problem. Within a sequence of interactions, the system's reflective behaviour is represented by a sequence

$$(i_1, [o_1, r_1]) \ , \ (i_2, [o_2, r_2]) \ , \ (i_3, [o_3, r_3]) \ , \ \ldots$$

For simplicity, we assume the values r_n to belong to the real interval $[0, 1]$. The system's utterance $r_n = 1$ is interpreted as the systems strong belief to ultimately solve the problem faced to. In contrast, $r_n = 0$ expresses the system's serious doubts as to solve the current problem. Values properly between 0 and 1 are understood as indicating some degree of uncertainty. Due to the incompleteness of intermediately presented information, a reflective system may change its opinion several times even rapidly.

The crux of interpreting as system's reflective utterances is that, on the one hand, a user is interested in clear and trustable statements, i.e. (s)he will usually prefer utterances like $r_n = 1$ or $r_n = 0$ to those which are more fuzzy. On the other hand, reflective behaviour is particularly important in non-trivial situations where the difficulties of incomplete information may obscure a system's introspection.

4 TYPES OF REFLECTION

Assume two AI systems on duty in the same application domain, but with different reflective behaviour. On some sequence of input data

$$i_1 \ , \ i_2 \ , \ i_3 \ , \ i_4 \ , \ i_5 \ , \ i_6 \ , \ \ldots$$

the two machines' behaviour is represented as

$$(i_1, [o'_1, r'_1]) \ , \ (i_2, [o'_2, r'_2]) \ , \ (i_3, [o'_3, r'_3]) \ , \ \ldots$$

and

$$(i_1, [o''_1, r''_1]) \ , \ (i_2, [o''_2, r''_2]) \ , \ (i_3, [o''_3, r''_3]) \ , \ \ldots$$

respectively. The first machine is said to reflect more optimistically than the second one, exactly if

$$r'_1 \geq r''_1 \ , \ r'_2 \geq r''_2 \ , \ r'_3 \geq r''_3 \ , \ \ldots$$

holds throughout the whole problem solving process. Naturally, the user of a reflective system is not interested in optimism or pessimism per se. The ultimate goal is a reflective self-evaluation approaching a correct reflection of the system's competence. This leads to a collection of definitions listed in the sequel.

For shortness, we will use the following abbreviation: A sequence of inputs is called admissible for some given AI system, if and only if it relates to a

problem within the area of competence of the system. In other words, admissible sequences are exactly those on which the system under consideration works successfully. In non-trivial cases, this property is undecidable. Our first concept is extremely weak:

Definition 1 An AI system enjoys the ability of *weak limiting reflection*, if and only if its behaviour on all admissible sequences of inputs satisfies $\lim_{n\to\infty} r_n = 1$ and this equality does not hold for any sequence of inputs which is not admissible. □

Definition 2 An AI system enjoys the ability of *limiting reflection*, if and only if its behaviour on all admissible sequences of inputs satisfies $\lim_{n\to\infty} r_n = 1$ and it guarantees $\lim_{n\to\infty} r_n = 0$ on all other sequences of inputs. □

Obviously, the property of limiting reflection is still quite weak and difficult to exploit, as every individual utterance r_n may be completely misleading.

For introducing some stronger versions of reflection types, assume any ε reasonably chosen from the interval $[0, 0.5)$.

Definition 3 An AI system enjoys the ability of *ε–optimistic reflection*, if and only if it has the property of limiting reflection and whenever any utterance r_n satisfies $r_n \leq \varepsilon$, this indicates that the given sequence is not admissible. □

Definition 4 An AI system enjoys the ability of *ε–pessimistic reflection*, if and only if it has the property of limiting reflection and whenever any utterance r_n satisfies $r_n \geq 1 - \varepsilon$, this indicates that the given sequence is admissible. □

Definition 5 An AI system enjoys the ability of *ε–exact reflection*, if and only if it has both ε–optimistic reflection and ε–pessimistic reflection. □

The property of 0–optimistic reflection has been called self-confident reflection in [Jan95].

Definition 6 The notion of *optimistic reflection*, *pessimistic reflection*, and *exact reflection* means ε–optimistic reflection, ε–pessimistic reflection, and ε–exact reflection, respectively, where ε equals 0. □

These latter reflection types have been introduced and intensively investigated in [Gri96].

The concepts above precisely describe certain levels of reflecting a system's competence. They can be effectively exploited. For example, a user working with a system which admits ε–optimistic reflection can deduce from any system's utterance $r_n \leq \varepsilon$ that the present problem exceeds the system's competence. Vice versa, if an ε–pessimistic system claims some $r_n \geq 1 - \varepsilon$, this informs the user in advance about the coming successful result. Hence, ε–exact reflection seems particularly useful.

Intuitively, it is more desirable to implement stronger reflection types. From another perspective, one may expect systems with reflection of a greater expressiveness to be less powerful.

Is it true that the requirement of ε–exact reflection restricts a system's capabilities compared to systems with only ε–optimistic (or ε–pessimistic) reflection? What are characteristic obstacles to combine ε–optimistic and ε–pessimistic reflection? Are there circumstances under which ε–optimistic and ε–pessimistic reflection coincide?

The present paper is intended to be a launching pad for investigations in the area of well-formalized reflection in AI to answer question like the few above.

The proposed methodology is to apply the basic reflection types to particular settings and to investigate the power and limitations of reflective intelligent systems within such a particular domain. The following chapter provides a prototypical approach.

5 REFLECTIVE LEARNING

To deal with complex cycles of man-machine interaction, we choose the application area of inductive learning, more precisely, inductive inference of total recursive functions from input/output examples (see [AS83] and [KW80], for excellent surveys).

An inductive inference machine (IIM, for short) is any recursive device which gets fed in input/output examples about some target phenomenon to be learnt. It generates hypotheses in response to input data. An IIM is learning some given recursive function, if it eventually generates a correct hypothesis which will never be changed again when receiving further information about this target phenomenon already learnt successfully.

Formalizations of mathematical precision are usually based on acceptable programming systems (cf. [MY74], e.g.). Here, we suppress these details.

Definition 7 A class U of total recursive functions is *learnable in the limit*, if and only if there exists some IIM which for any function f of U excepts arbitrary finite samples of f's behaviour as inputs and generates programs as hypotheses such that after finitely many steps of interaction all hypothesized programs are identical and compute f correctly. A class U of total recursive functions is *finitely learnable*, if and only if it is learnable in the limit by some IIM such that it is uniformly decidable whether the finally correct hypothesis has already been found. □

Obviously, IIMs respond to sequences of samples $s_1, s_2, s_3, \ldots$ put in by sequences of generated hypotheses $h_1, h_2, h_3, \ldots$

Definition 8 An IIM is called *reliable*, if and only if its hypotheses never converge on input sequences which are not admissible. □

The phenomenon of reliable inductive inference is well-understood (cf. [BB75]).

Definition 9 A *reflective* IIM is an IIM which additionally generates utterances $r_1, r_2, r_3, \ldots$ expressing the results of its introspection according to one of the types of reflection introduced. □

A few results are intended, on the one hand, to exhibit that one may derive precise results about the power and limitations of reflective AI systems and, on the other hand, to illustrate that the effects may be interesting and even sometimes surprising.

Theorem 1
For every reliable IIM, there exists a version which has the property of weak limiting reflection. ♣

The inverse implication seems not to be valid.

Theorem 2
For every $\varepsilon \in [0, 0.5)$ and for every IIM which has the property of ε–optimistic (ε–pessimistic) reflection, there exists an IIM solving the same learning problems with optimistic (pessimistic) reflection. ♣

This motivates to focus on the case $\varepsilon = 0$.

Theorem 3 [Gri96]
For IIMs learning function classes in the limit and for binary reflection, i.e. $r_n \in \{0, 1\}$ (for alle n), limiting reflection, optimistic reflection, pessimistic reflection, and exact reflection coincide. ♣

This result seems quite surprising, as approaches of an intuitively distinguished strength turn out to be provably equivalent under very weak assumptions. [Jan95] contains a similar main result.

Theorem 4 [Gri96]
For finitely learning IIMs and for binary reflection, i.e. $r_n \in \{0, 1\}$ (for alle n), limiting reflection, optimistic reflection, pessimistic reflection, and exact reflection are mutually different. ♣

In contrast to the result before, this proves that under certain circumstances competence assessment is possible and the chosen type of reflection is crucial.

6 ACKNOWLEDGEMENT

Initially, this work has been partially supported by the German Federal Ministry for Research and Technology (BMFT) within the Joint Project (BMFT-Verbundprojekt) GOSLER on *Algorithmic Learning for Knowledge-Based Systems* under contract no. 413-4001-01 IW 101. Further results of this joint project are published in [JL95]. The author's conference participation has been supported by the German Research Fund (DFG) under grant 477/378/96.

The author's student Gunter Grieser has provided a remarkable number of approaches, formalizations, and results first documented in his Master's Thesis [Gri96]. First results are published in [GJ95].

References

[AS83] Dana Angluin and Carl H. Smith. A survey of inductive inference: Theory and methods. *Computing Surveys*, 15:237–269, 1983.

[BB75] Leonore Blum and Manuel Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.

[GJ95] Gunter Grieser and Klaus P. Jantke. Ansätze zur Reflexion in der Induktiven Inferenz. Studie der Forschungsgruppe Algorithmisches Lernen, HTWK Leipzig (FH), FB Informatik, Mathematik & Naturwissenschaften, Juni 1995. Studie #02/95, Version 1.0.

[Gri96] Gunter Grieser. Reflexion in der Induktiven Inferenz. Master's thesis, Leipzig University of Technology, March 1996.

[Jan95] Klaus P. Jantke. Reflecting and self-confident inductive inference machines. In Klaus P. Jantke, Takeshi Shinohara, and Thomas Zeugmann, editors, *Proc. 6th International Workshop on Algorithmic Learning Theory, (ALT'95), October 18-20, 1995, Fukuoka, Japan*, volume 997 of *LNAI*, pages 283–298. Springer-Verlag, 1995.

[JL95] Klaus P. Jantke and Steffen Lange, editors. *Algorithmic Learning for Knowledge-Based Systems*, volume 961 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1995.

[KW80] Reinhard Klette and Rolf Wiehagen. Research in the theory of inductive inference by GDR mathematicians - a survey. *Inform. Sci.*, 22:149-169, 1980.

[MY74] Michael Machtey and Paul Young. *An Introduction to the General Theory of Algorithms*. North-Holland, 1974.

[vHABS+92] Frank van Harmelen, Hans Akkermann, Brigitte Bartsch-Spörl, Bert Bredeweg, Carl-Helmut Coulon, and Uwe Drouven. Knowledge-level reflection: Specifications and architectures. Esprit Basic Research Project P3178 REFLECT Report, Univ. Amsterdam, June 1992.

[vHW92] Frank van Harmelen and Bob Wielinga. Knowledge-level reflection. In B. LePape and L. Steels, editors, *Enhancing the Knowledge Engineering Process*, pages 175–204. Elsevier Science Publ., 1992.

Explaining Anomalies as a Basis for Knowledge Base Refinement

Neli P. Zlatareva
Department of Computer Science
Central Connecticut State University
New Britain, CT 06050

Abstract

Explanations play a key role in operationalization-based anomaly detection techniques. In this paper we show that their role is not limited to anomaly detection; they can also be used for guiding automated knowledge base refinement. We introduce a refinement procedure which takes: (i) a small number of refinement rules (rather than test cases), and (ii) explanations constructed in an attempt to reveal the cause (or causes) for inconsistencies detected during the verification process, and returns rule revisions aiming to recover the consistency of the KB-theory. Inconsistencies caused by more than one anomaly are handled at the same time, which improves the efficiency of the refinement process.

1 Introduction

We consider knowledge base (KB) refinement in the context of Knowledge-Based System (KBS) validation. The greatest advantage of this approach is that we can use the outcome of other validation activities to detect anomalies in the KB-theory. Similar to some machine learning techniques [Wilkins, 1988, Ourston and Mooney, 1994, DeJong and Mooney, 1986], the one presented here uses explanations of detected anomalies in its attempt to reveal the culprit (or culprits) for each of them. However, contrary to the situation in explanation-based learning, where explanations are constructed for each of the successfully solved test cases, our procedure generates and evaluates only explanations related to detected anomalies.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/271 ©1996 FLAIRS

Compared to other refinement methods designed to support KBS validation [Mesenguer, 1993, Zlatareva, 1994, Palmer and Craw, 1995], the one presented here is different in that it does not require test cases. Instead, the refimenement procedure makes use of a small number of refinement rules for guiding the refinement process.

The paper is organized as follows. In Section 2, we review related work in KB refinement. Section 3 presents the refinement procedure in the context of other validation activities, and Section 4 illustrates it by means of an extended example.

2 Related work

In the KBS domain, knowledge base refinement is traditionally viewed as a part of the knowledge acquisition process, and it is most often implemented by means of some machine learning technique. Inductive (or empirical) learning [Quinlan, 1983] can be used if a significant number of real world test cases is available. For example, the EITHER system required 80 test cases to improve the accuracy of the knowledge base by 35 percentage point [Ourston and Mooney, 1994]. Such an extensive number of test cases is typically not available during KBS development.

Explanation-based learning (EBL) does not require an extensive number of test cases, but it assumes that the KB-theory is correct. The goal of the learning process is to build new operational rules from the explanations produced by the existing theory in its attempt to explain presented test cases. This activity can hardly be called "learning", because the system does not really acquire new knowledge from presented test cases. But it finds a more efficient way to represent the KB-theory. In this process, redundant rules (if such have existed in the initial theory) are automatically eliminated. We consider such a "compilation" of the KB-theory after it

has been proven structurally and functionally correct (by means of some other verification method) an important part of the refinement process. But it can only take place if the following two conditions are met:

- Real-world test cases with known solutions are available during KBS development.
- The efficiency of knowledge is more important that its accessibility.

Similar to EBL, apprenticeship learning [Wilkins, 1988] involves building explanations and using them as a component of learning. The difference between them is that in apprenticeship learning, learning occurs when the KB-theory is unable to correctly explain the presented test case. Because the knowledge base may contain errors, it is assumed that this is what causes the problem. To fix it, the apprenticeship learning program builds modifications of the KB-theory that allow a successful explanation of the presented test case to be constructed. A similar approach to KB refinement is implemented in the KRUST system [Palmer and Craw, 1995], which supports validation of rule-based KBSs. In both frameworks, the availability of test cases is crucial for the success of the refinement process. As it is well known, test cases are rarely available during KBS development, which limits the applicability of these methods.

The refinement procedure presented here does not require test cases. Instead, it uses a small number of refinement rules to locate the culprits for a detected problem, and offer possible refinements. It handles independent anomalies in the KB-theory causing the same problem, at the same time. This situation is not addressed by other refinement methods, which tackle each anomaly independently.

3 Outline of the refinement process

We consider KB-theories represented as propositional or quasi-first-order-logic Horn clauses. We assume that the KB-theory is a complete, but incorrect model of the domain theory. The KB-theory may contain errors, such as logical and semantic inconsistencies, redundant and circular rules, and unreachable hypotheses. The ultimate goal of the refinement process is to free the KB-theory from all of the errors detected during its validation. Next, we discuss the refinement process in its connection with other validation activities. We assume that the reader is familiar with operationalization-based verification techniques. The DIVER tool [Zlatareva and Preece, 1994], which implements such a technique, is used to detect anomalies in the KB-theory.

Anomaly detection

Using DIVER, the KB-theory is first tested for potential structural anomalies. The expected outcome of this process is:

- One or more operational descriptions of the KB-theory, called *grounded stable extensions*, where each reachable hypothesis is represented in an operational form. As an intermediate step, DIVER produces *stable extensions* of the KB-theory, where each hypothesis is represented in terms of its immediate predecessors.
- A set of potential anomalies that may occur in possible real world situations.

Each of the detected anomalies points to a real error, unless an important semantic constraint is missing from the description of the KB-theory. If this is the case, the missing constraint must be added before the refinement process is initiated.

Detected anomalies can be divided into two groups, depending on the underlying cause, as follows:

- Anomalies resulting from the *incompleteness* of the KB-theory.
- Anomalies resulting from the *inconsistency* of the KB-theory.

Each anomaly can be explained in terms of data and rules involved in its generation. For anomalies caused by the incompleteness of the KB-theory, only partial explanations can be constructed. To complete these explanations, the refinement procedure performs a search for unusable components and incomplete specifications, provided that test cases are available. We have discussed this aspect of KB refinement in [Zlatareva, 1994]. Thus we shall assume that the KB-theory is complete, and all of the detected errors are caused by data or rules inconsistencies.

Anomaly explanation

At each run, DIVER produces a set of logical and semantic inconsistencies, not all of which point to real errors (as mentioned above, some of these inconsistencies may be caused by unaccounted semantic constraints). After filtering out "pseudo" inconsistencies, the refinement procedure sorts the rest into levels depending on how early in the operationalization process the inconsistency was detected. That is, inconsistencies detected at the first step of the operationalization process comprise "level 1", inconsistencies detected at the second step comprise "level 2", etc. The refinement procedure deals with all of the inconsistencies at a given level at the same time, because they are likely to be caused by independent anomalies. Inconsistencies at higher levels may depend on inconsistencies detected earlier, which is why their resolution is postponed until inconsistencies at the lower levels are resolved.

Let $C_{F_i/F_j} : (F_i, F_j)(C_{F_i/F_j})$ be the inconsistency currently examined (for logical inconsistencies, $F_j = \neg F_i$), and let SE_k be one of the stable extensions generated by DIVER at the current step of the verification process. SE_k is searched for formulas whose heads are F_i or F_j[1]. The union of the *T-sets* (i.e. the sets containing the facts, intermediate hypotheses and rules upon which head F_i depends), constitute the explanation for F_i relative to SE_k. Similarly, the union of the *T-sets* of all of the formulas for F_j constitute the explanation for F_j. To construct the explanation for inconsistency C_{F_i/F_j}, the refinement procedure computes the intersection of the explanations for F_i and F_j. That is,

$$(E1_{F_i} \vee E2_{F_i} \vee ... \vee En_{F_i}) \wedge (E1_{F_j} \vee E2_{F_j} \vee ... \vee Em_{F_j}).$$

This can be represented in the following equivalent form:

$$E1_{F_i} \wedge E1_{F_j}, ..., E1_{F_i} \wedge Em_{F_j},$$
$$E2_{F_i} \wedge E1_{F_j}, ..., E2_{F_i} \wedge Em_{F_j},$$
$$...$$
$$En_{F_i} \wedge E1_{F_j}, ..., En_{F_i} \wedge Em_{F_j}.$$

Each of these formulas provides an independent explanation for C_{F_i/F_j}. These explanations are both useful and relevant to the refinement process, because they have the following properties:

- *Efficiency.* Each explanation involves the minimal input data set and the minimal set of rules implying a given inconsistency.
- *Completeness.* If there exists an explanation for a given inconsistency, then this explanation will be constructed.
- *Relevance.* If all of the components of constructed explanations had been related meaningfully, then KB-theory would have been valid.

Evaluation of constructed explanations and identification of possible refinements

The explanations constructed for detected inconsistencies may explicate independent anomalies in the KB-theory causing the same problem. This is why each explanation is searched for possible anomalies by means of the following tests:

Search for duplicated data

If the *T-sets* of F_i and F_j contain the same datum, then this datum will be encountered twice in the explanation for C_{F_i/F_j}. Such a datum is likely to be either redundant (and therefore, it should be removed from the affected rules), or the culprit for the inconsistency.

Consider the following two rules: $A \wedge B \rightarrow H_i$ and $C \wedge B \rightarrow \neg H_i$. There are two possible reasons for these rules to be inconsistent: (i) A and C are inconsistent, in which case B is a redundant premise, and (ii) B is the culprit for the inconsistency, in which case it must be removed from the left hand side of one of the two rules. Notice that the latter case is equivalent to specializing a predecessor rule with the negation on B (see example in Section 4). This suggests two possible revisions of the KB-theory, captured by the following refinement rules:

- Rule 1: Generalize all of the rules containing the redundant datum by removing it from their left hand sides.
- Rule 2: If the duplicated datum participates in same-level conflicting rules, then select (arbitrary) one of them for generalization, and remove this datum from its left hand side.
- Rule 3: If a duplicated datum participates in conflicting rules fired at different steps of the operationalization process, then select (according to Rule 4 below) the appropriate conflicting

[1] Stable extensions consist of two types of formulas: $F_i : (B_1, ..., B_m)()$ and $F_j : (A_1, ..., A_n)(C_{F_h}, ...C_{F_k})$, where F_i and F_j are the heads; $B_1, ..., B_m$ and $A_1, ..., A_n$ are the *T-sets*; and $C_{F_h}, ..., C_{F_k}$ is the *U-set*.

rule for specialization with the negation of a duplicated datum.

Search for conflicting rules

Rules involved in a given explanation are ordered into levels depending on how early in the operationalization process the rule has been applied. It is reasonable to assume that "level 0" rules (directly acquired from the expert) are most likely to be correct. Under this assumption, the following rule defines the possible candidate for revision among a set of conflicting rules:

- Rule 4: If conflicting rules belong to different levels, then the highest level rule is selected for specialization.

Obviously, the number of perspective refinements generated by means of these rules is very small, and each of them can be tested by verifying the revised theory.

4 Example

To illustrate the refinement procedure, consider the following example theory[2]:

R_0 : $work\_hours(Student, \leq 35) \rightarrow$
$unemployed(Student)$.
R_1 : $unemployed(Student) \rightarrow$
$financial\_deferment(Student)$.
R_2 : $filed\_for\_bancrupcy(Student) \rightarrow$
$financial\_deferment(Student)$.
R_3 : $international(Student) \rightarrow$
$unemployed(Student)$.
R_4 : $international(Student) \rightarrow$
$\neg eligible\_for\_deferment(Student)$.
R_5 : $military\_deferment(Student) \rightarrow$
$eligible\_for\_deferment(Student)$.
R_6 : $financial\_deferment(Student) \rightarrow$
$eligible\_for\_deferment(Student)$.
R_7 : $married(Student) \rightarrow$
$eligible\_for\_deferment(Student)$.
R_8 : $eligible\_for\_deferment(Student) \rightarrow$
$no\_payment\_due(Student)$.
R_9 : $credits\_taken(Student, \geq 15) \rightarrow$
$full\_time(Student)$.
R_{10} : $full\_time(Student) \wedge GPA(Student, \geq 3.5) \rightarrow$
$honors(Student)$.
R_{11} : $honors(Student) \rightarrow$
$eligible\_for\_deferment(Student)$.
R_{12} : $\neg eligible\_for\_deferment(Student) \rightarrow$
$\neg no\_payment\_due(Student)$.

To simplify the representation, assume the following abbreviations:

F_0 : $work\_hours(Student, \leq 35)$,
F_1 : $unemployed(Student)$,
F_2 : $financial\_deferment(Student)$,
F_3 : $filed\_for\_bancrupcy(Student)$,
F_4 : $international(Student)$,
F_5 : $eligible\_for\_deferment(Student)$,
F_6 : $military\_deferment(Student)$,
F_7 : $married(Student)$,
F_8 : $no\_payment\_due(Student)$,
F_9 : $credits\_taken(Student, \geq 15)$,
F_{10} : $full\_time(Student)$,
F_{11} : $GPA(Student, \geq 3.5)$,
F_{12} : $honors(Student)$.

The initial KB-theory evaluated by DIVER is the following:

$R_0 : (F_0)() \rightarrow F_1$, $R_1 : (F_1)() \rightarrow F_2$,
$R_2 : (F_3)() \rightarrow F_2$, $R_3 : (F_4)() \rightarrow F_1$,
$R_4 : (F_4)() \rightarrow \neg F_5$, $R_5 : (F_6)() \rightarrow F_5$,
$R_6 : (F_2)() \rightarrow F_5$, $R_7 : (F_7)() \rightarrow F_5$,
$R_8 : (F_5)() \rightarrow F_8$, $R_9 : (F_9)() \rightarrow F_{10}$,
$R_{10} : (F_{10}, F_{11})() \rightarrow F_{12}$, $R_{11} : (F_{12})() \rightarrow F_5$,
$R_{12} : (\neg F_5)() \rightarrow \neg F_8$.

Here $\{F_0, F_3, F_4, F_6, F_7, F_9, F_{11}\}$ are input data, $\{F_8, \neg F_8\}$ are final hypotheses, and $\{F_4 \wedge F_6, F_4 \wedge F_6\}$ are primary semantic constraints.

At the first run, DIVER produces the following stable extension:

$SE_1 = \{F_1 : (F_0, R_0)(), F_2 : (F_3, R_2)()$,
$F_1 : (F_4, R_3)()$, $\neg F_5 : (F_4, R_4)()$, $F_5 : (F_6, R_5)()$,
$F_5 : (F_7, R_7)()$, $F_{10} : (F_9, R_9)()$,
$C_{F_5} : (F_5, \neg F_5)(C_{F_5})$, $F_2 : (F_1, R_1)()$,
$F_5 : (F_2, R_6)()$, $F_8 : (F_5, R_8)(C_{F_5})$,
$F_{12} : (F_{10}, F_{11}, R_{10})()$, $\neg F_8 : (\neg F_5, R_{12})(C_{F_5})$,
$C_{F_8} : (F_8, \neg F_8)(C_{F_5}, C_{F_8})$, $F_5 : (F_{12}, R_{11})()\}$.

There are two logical inconsistencies detected at this step, C_{F_5} and C_{F_8}, where C_{F_5} is a "level 1" inconsistency, and C_{F_8} is a "level 2" inconsistency. Therefore, the refinement procedure attempts to resolve C_{F_5} first, and it constructs the following inde-

[2]The original version of this example was introduced in [Pazzani and Brunk, 1991]; here we use a modified version of it.

pendent explanations for it:

$E1_{C_{F5}} \Leftarrow (F_0, F_4, R_0, R_1, R_4, R_6)$,
$E2_{C_{F5}} \Leftarrow (F_4, F_4, R_1, R_3, R_4, R_6)$,
$E3_{C_{F5}} \Leftarrow (F_4, F_6, R_4, R_5)$,
$E4_{C_{F5}} \Leftarrow (F_4, F_7, R_4, R_7)$,
$E5_{C_{F5}} \Leftarrow (F_3, F_4, R_2, R_4, R_6)$,
$E6_{C_{F5}} \Leftarrow (F_3, F_4, R_2, R_4, R_6)$,
$E7_{C_{F5}} \Leftarrow (F_4, F_9, F_{11}, R_4, R_{10}, R_{11})$.

Each explanation is searched for violated semantic constraints to filter out possible pseudo anomalies. $E3_{C_{F5}}$, $E5_{C_{F5}}$ and $E6_{C_{F5}}$ do contain semantic constraints which is why they are removed from further consideration.

The search for duplicated data identifies $E2_{C_{F5}}$, where F_4 is encountered twice. According to Rule 3, F_4 must be used for specializing one of the conflicting rules, R_4 or R_6. Because R_6 is the highest level rule, it is the one selected for specialization according to Rule 4. As a result, the following revision of R_6 is suggested:

$R_6 : (F_2, \neg F_4)() \rightarrow F_5$.

The analysis of $E1_{C_{F5}}$, $E4_{C_{F5}}$ and $E7_{C_{F5}}$ is carried out in the same way, and it results in two more revisions:

$R_7 : (F_7, \neg F_4)() \rightarrow F_5$
$R_{11} : (F_{12}, \neg F_4)() \rightarrow F_5$

These revisions require the input data set to be extended with $\neg F_4$, which introduces a new primary semantic constraint, $F_4 \wedge \neg F_4$.

The suggested revisions are tested by initiating a new verification cycle. At this run, DIVER will produce two grounded stable extensions, and both of them will be found consistent.

5 Conclusion

In this paper, we have introduced a refimenement procedure, which takes a small number of refinement rules, and explanations constructed in an attempt to reveal the culprit (or culprits) for inconsistencies detected during the verification process, and returns rule revisions aiming to recover the consistency of the KB-theory. The motivation behind this work is that test cases are rarely available during KBS development, and therefore refinement methods that make use of them have limited application. The presented refinement procedure is intended to be used as a component of a validation framework for rule-based Knowledge-Based Systems.

Acknowledgement. Many thanks to Charles Neville for his useful comments.

References

[DeJong and Mooney, 1986] DeJong, G. and Mooney, R. (1986). Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–176.

[Mesenguer, 1993] Mesenguer, P. (1993). Expert system validation through knowledge base refinement. In *Proc. 13-th International Joint Conferences on Artificial Intelligence (IJCAI'93)*, pages 477–482. Morgan Kaufmann Publ.,Inc.

[Ourston and Mooney, 1994] Ourston, D. and Mooney, R. (1994). Theory refinement combining analytical and empirical methods. *Artificial Intelligence*, 66:273–309.

[Palmer and Craw, 1995] Palmer, D. and Craw, S. (1995). Utilising explanation to assist the refinement of knowledge-based systems. In *Proc. EUROVAV-95*, pages 201–211.

[Pazzani and Brunk, 1991] Pazzani, M. and Brunk, C. (1991). Detecting and correcting rule-based expert systems: an integration of empirical and explanation-based learning. *Knowledge Acquisition*, 3:157–173.

[Quinlan, 1983] Quinlan, J. (1983). Learning efficient classification procedures and their application to chess and games. In *Machine Learning - An Artificial Intelligence Approach*. Morgan Kaufmann.

[Wilkins, 1988] Wilkins, D. (1988). Knowledge base refinement using apprenticeship learning techniques. In *Proc. AAAI'88*, pages 646–651.

[Zlatareva, 1994] Zlatareva, N. (1994). A framework for verification, validation, and refinement of knowledge bases: The VVR system. *International Journal of Intelligent Systems*, 9(8):703–738.

[Zlatareva and Preece, 1994] Zlatareva, N. and Preece, A. (1994). An effective logical framework for knowledge-based systems verification. *International Journal of Expert Systems: Research and Applications*, 7.

INTRODUCTION OF PROCESSING MECHANISMS IN KNOWLEDGE BASED CONCEPTUAL MODELS

C. Frydman, L. Torres, N. Giambiasi
DIAM-IUSPIM
Av Escadrille Normandie Niemen
13397 Marseille Cedex 20
France
E-mail : diam_cfr@vmesa11.u-3mrs.fr

M. Le Goc
Sollac-SACHEM
LB 1
13776 Fos sur Mer Cedex
France

Abstract

In this paper, we propose an approach based on high level Petri nets, allowing us both to validate the conceptual model and to solve the problem concerning the passage of a conceptual model to an executable model.

Keywords

operationalizing, conceptual model, executable model.

1 INTRODUCTION

The specification of a knowledge based system is considered as a modelling activity that must product a conceptual model. This model describes the problem solving tasks and the various solving strategies. It must be free of implementation problems. All the difficulty lies in designing and coding the system from this conceptual model.

In this paper, we are interested in the transformation of a conceptual model into an executable model. Our approach is based on the KADS methodology used to build the conceptual model, and on the Petri net formalism used to represent and simulate the dynamic of the model.

First, we will present the problems linked to the specification of knowledge based systems using KADS. Then, we will show how to transformate a specification model into an executable one. Finally, we will illustrate the use of our approach to the Sachem system of blast furnace control.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/276 ©1996 FLAIRS

2 SPECIFYING KNOWLEDGE BASED SYSTEMS WITH KADS

The design of a computer system constitutes a complex problem. The complexity is bigger if the system needs to make heterogeneous technologies cohabit: database management (numerical and symbolic), knowledge base, signal processing, pattern recognition, CASE, human-computer interfaces, new technics emerging from research laboratories, etc. Then, there is a methodological problem: how to design such a system? In other words, it is difficult to establish rigorous and precise specifications at the architectural and functional levels.

In the SACHEM project, KADS [Breuker and Wielinga 1985; Hesketh and Barret 1989; Wielinga et al. 1989; Wielinga et al. 1991] has been chosen as the specification methodology. KADS is a structured methodology for the development of knowledge based systems. KADS approaches the development of knowledge based sytems as a modelling activity. It uses multiple models to cope with the complexity of knowledge engineering and it uses knowledge level descriptions as an intermediate model between system design and expertise data [Schreiber et al. 1993].

Unlike most other approaches, KADS distinguishes between a conceptual model of expertise independent of a particular implementation, and a design model specifying how a model of expertise is operationalized with particular computational and representational techniques. In other words, KADS distinguishes between knowledge level [Newell 1982] and computational level models.

As a consequence, the analysis model and design are completely disconnected. Therefore, it may be difficult to recognize the conceptual model in the designed system, to transfer modifications from the one to the other, and to explain the behaviour of the system in the terms of the conceptual model [Karbach et al. 1991].

On the one hand, KADS makes easier the specification of the problem-solving expertise but, on the other hand, it creates the additional work of "making it run" [Wielinga et al.]. Making the conceptual model run is the objective of our research.

2.1 The Four-Layered Conceptual Model

Four layers compose the KADS conceptual model.

The layer of the highest level is the "Strategy" one. It contains the sets of aims that the knowledge corpus can be achieved, by the activation of the problem solving procedures described in the "Task" layer. These tasks activate the operators of the "Inference Structure" layer. The operators describe the logical transformations that lead to the problem solving. The logical transformations are realized by the knowledge in the "Domain" layer. This last layer contains the totality of the truly expert knowledge, that is the set of concepts and relation needed to solve a concrete problem [van Harmelen et al. 1991].

The KADS methodology, by preconizing the distinction between conceptual model and design model, contributes to clarify the knowledge based system building process. But it does not take into account the problem of passing from specification to design. The KADS approach to design is based on a manual transformation of the knowledge model to the designed system. This process is inefficient and prone to errors [Wielinga et al. 1993].

3 PASSING FROM SPECIFICATION TO DESIGN

From the conceptual model, describing the specifications, we must design and code a system that correctly and efficiently realizes the specifications, and that satisfies the final users. To obtain such a system, we must be able to verify the validity of the system in relation to the specifications, and first, the validity of the specifications relating to the users' expectations.

Now, we are faced with the problem concerning the relation between the conceptual model and the designed system. This problem is about the passage from specification to design.

Usually, this passage is solved by the application of a homomorphy principle. This approach consists in respecting the structure of the conceptual model, and even in producing a system with the same structure. It sets us the problem of the nature of the conceptual model: is it an expert model or more generally a model of the complete system? If the second possibility is adopted, the model must contain at least some implementation constraints (or external constraints of the system) that not concern the expert. Then, building a conceptual model is no more an operation depending on the only expert analysis.

Some research works deal with this question by operationalizing conceptual models [Jacob-Delouis and Krivine 1995; van Harmelen 1991]. The aim is to give a semantics to the objects of the conceptual model that can be interpreted by a machine. This allows to define a language of high level of abstraction. This approach gives the advantage of allowing an analysis of the syntactical coherence of a conceptual model and a simulation of the dynamical behavior of the system described at a conceptual level. Simulation is made before the design and code phases of the system, allowing particularly the validation of specifications. On the other hand, we are faced with the double coding problem, carrying away the well-known problems of code validation.

The advantage of simulation is decisive for a very constrained temporally system. The drawback of double coding is redhibitory for a complex and large system.

Since we adopted the KADS methodology for our works, the transformation from specification to design consists in transforming the four-layered conceptual model. To avoid the previous problem of double coding, while providing a possibility of analyzing dynamics of the conceptual model, we decide to make only the three high layers of the conceptual model operational. Indeed, the specification of the dynamic behavior of the system is defined by these three layers, and more particularly by the "Inference Structure" layer and the "Tasks" layer. The constraints on the temporal ordering of the problem solving tasks can be analyzed in these layers, without taking into account the "Domain" layer.

4 MAKING THE CONCEPTUAL MODEL RUN

We have adopted the Petri net paradigm [Brams 1983; David and Alla 1989] to represent and simulate the dynamics contained in the KADS conceptual model. This allows us to use existent simulators.

First, we transform the "Inference Structure" and "Tasks" layers of the conceptual model, in the form of a time [Merlin 1974; Merlin et al. 1976] hierarchical [Jensen et al. 1990] and interpreted Petri net. Then, we have to complete the previous net by taking into account the control knowledge expressing the dynamics of the system.

4.1 The Operator Precedence

The "Inference Structure" layer contains the operators of the conceptual model. An operator is defined by:

- its name,
- its input data that constitute its operands,
- its output data that constitute its results,

- references to concepts of Domain layer: each data refers to some concepts of the Domain layer.

From the description of the operator set of the "Inference Structure" layer, we define the Petri net such as:
- the places are the operands and the results of the operators,
- the transitions are the operators themselves.

The same operand or result used by two different operators generates only one place in the Petri net.

The use of time Petri nets [Merlin 1974; Merlin et al. 1976] allows us to represent the time concept. Such Petri nets associate a time interval. with each transition. In our context, this time interval represents the lowest bound and the highest bound of the processing time of an operator. It is convenient to use intervals, because processing times cannot be defined exactly since the specification phase.

The Petri net directly built from the operators, is only a precedence graph between operators. This first Petri net allows yet some verifications on the operator set.

For example, we can verify that:
- the input/output of the model are really those that are defined for the system;
- there are not isolated operators, that is operators that cannot be reached during the problem solving;
- all the operators can be reached;
- etc.

4.2 The Operative Model of the Task

KADS introduces the concept of task to structure a conceptual model. In the "Tasks" layer, the user defines a set of tasks where any task contains operators and/or other tasks.

The hierarchical structure of tasks is naturally represented by hierarchical Petri nets [Jensen et al. 1990; Jensen 1992]. Each task may be viewed as a Petri net composed by one transition and input and output places. The alone transition is a substitution transition [Jensen 1992]. It corresponds to a Petri subnet, obtained from the Petri nets of its components, where each component is represented by only one transition and input and output places. The transitions associated with the components of the task are substitution transitions that correspond, in their turn, to Petri subnets, and so on.

The hierarchical Petri net of the task constitutes its operative model. It represents the task at different abstraction levels.

The whole system considered as a task, can be associated by a hierarchical Petri net, as any part of the system.

4.3 The Control Model of the Task

To make the task executable, the user has to define its control model, in addition to its operative model. This consists in describing the type of control of the components of the task. Various control types may be considered: sequence, alternative, repetition, parallelism, simultaneity, locking, conditional execution, etc.

Defining an control type may implicate the addition of control places and transitions to the operative model to constitute the executable model

Constraints defined in the precedence graph must be respected by the control model. As the user does not know the precedence graph of operators, verification procedures will be introduced to ensure the validity of the model.

For instance, in the case of a sequence between two tasks t1 and t2 (represented by two transitions in the Petri net) in the order t1 - t2, we proceed in the following manner:

- if t1 and t2 are directly or indirectly in sequence in the operator precedence graph in the same order, the executable model is the same as the operative model;

- if t1 and t2 are independent in the operator precedence graph, the executable model will contain a control place (as output of t1 and input of t2);

- otherwhise, this sequencing is not acceptable.

The conditional execution of a task can be modelled by associating a logical expression with the corresponding transition. This justifies the use of interpreted Petri net [Brams 1983; David et al. 1989].

Finally, we obtain an executable model of the task that can be simulated at different abstraction levels.

5 APPLYING OUR APPROACH TO THE SACHEM PROJECT

Our approach has been validated on the control of the blast furnaces of Usinor+Sacilor Group (the first European steel company). The project SACHEM aims at the development of an interactive knowledge based system to help the decision making to the control blast furnace activity, in real time. Begun in 1991 by Sollac which is the flat products branch of Usinor+Sacilor Group, this project is regarded as one of the most important industrial project of artificial intelligence in Europe.

The SACHEM software must provide, in real time, a continuous service to the operator, 24 hours per day, 365 days per year. It must also help the study of the control blast furnace activity, to ameliorate the quality of the knowledge patrimony.

The target of the system is the set of the blast furnace of the Usinor+Sacilor group, then the specification imposes to built a generic Sachem system (free of blast furnace), single, maintainable and adaptable with minor expenses. Therefore, the property to be generic of Sachem becomes a strategic stake.

The problem is complex. Indeed, the expert behavior is based on a large phenomenology that relieve of the chemistry and physical. It treats temporal phenomenon found in an industrial environment, with many imperfections (vagueness, failing captors, erroneous information, ...). More, there is not knowledge needed to build a generic Sachem, because there is generic expertise as generic blast furnaces. Then, it must be build from an expert college, from distant sites and different administratively.

Sachem is a complex knowledge based system that has been modelled with the KADS methodology. Therefore, our approach allows us:

- to verify the validity of the Sachem conceptual model;

- to study different software architectures of the system before design;

- to take into account the time concept and the different kinds of control needed by the truly real time systems (synchronization, interruptions, sharing, others);

- to simulate Sachem at different abstraction levels.

6 CONCLUSION

The approach we have adopted to build an executable model from a conceptual model is based on the use of Petri nets of high level.

This approach allows us both to validate the conceptual model and to make the problem concerning the passage of a conceptual model, to an executable model, clear. Moreover, we can simulate the dynamical behavior of the system before its design and coding.

Now, we are studying a ressource model. Adding this model will provide a design choice at both software and hardware architectures.

References

[Bax 1995] M. Bax : *Une approche mixte réseaux de Petri et modélisation orientée objet : le formalisme OPenRT pour la modélisation de systèmes distribués temps réel.* Thèse de l'Université de Montpellier II, December 1995.

[Brams 1983] G. Brams : *Réseaux de Petri : Théorie et pratique.*, Ed Masson Paris 1983.

[Breuker and Wielinga 1985] J. Breuker , B. Wielinga: *KADS Structured knowledge acquisition for expert systems,* Expert systems and theirs applications, Avignon 1985, p.887-900.

[David and Alla 1989] R. David, H. Alla: *Du graphcet aux réseaux de Petri*, Hermes 1989.

[Hesketh and Barret 1989] P. Hesketh, T. Barret: *An introduction to the KADS methodology,* deliverable M1, Esprit projet P1098, 12/21/89.

[Jacob-Delouis and Krivine 1995] J. Jacob-Delouis, J.-P. Krivine: *LISA : un langage réflexif pour opérationnaliser les modèles d'expertise*, Revue d'intelligence artificielle, vol. 9, n°1, 1995, p.53-88.

[Jensen et al. 1990] K. Jensen, P. Huber and R.M. Shapiro: *Hierarchies in colored petri nets.* Advances in Petri nets, Lecture Notes in Computer Science, 483: 313-341, 1990.

[Jensen 1994] K. Jensen: *An introduction to the theoretical aspects of colored petri nets.* In a Decade of concurrency, Lecture Notes in Computer Science, 803: 230-272, Springer-Verlag 1994.

[Karbach et al. 1991] W. Karbach, A. Vob and U. Drouven: *Model-K: Prototyping at the Knowledge Level*, Expert Systems and their Applications, Avignon France 1991.

[Merlin 1974] P. Merlin: *A study of recoverability of computer systems.* Ph D thesis, University of California, Irvine 1974.

[Merlin 1976] P. Merlin and al.: *Recoverability of communication protocols- implications of a theoretical study* IEEE Transactions on Communications, 24 (9) 1976.

[Newell 1982] A. Newell: *The knowledge level*, Artificial Intelligence 18, 1982, p.87-127.

[Schreiber et al. 93] G.Schreiber, B. Wielinga, J. Breuker: *KADS : A Principled Approach to Knowledge Engineering.* Academinc press, London ,1993

[van Harmelen et al. 1991] F. van Harmelen, J.R. Balder, M.W.M.M. Aben, J.M. Akkermans: *(ML)2 A formal language for KADS models of expertise,* Esprit projet P5248 KADS-II, KADS-II / T1.2 / TR / ECN / 006 / 1.0, D1.2.1, 28/10/91.

[Wielinga et al. 1989] B.J. Wielinga, G.Schreiber, P. de Greef:*KADS synthesis report*, deliverable Y3, Esprit projet P1098, UvA-Y3-PR-001, 03/21/89.

[Wielinga et al. 1991] B.J. Wielinga A. Th. Schreiber, J.A. Breuker: *KADS a modelling approach to knowledge engineering*, Esprit projet P5248 KADS-II, KADS-II / T1.1 / PP / UvA / 008 / 1.0, 11/4/91.

[Wielinga et al. 1993] B. J. Wielinga, W. Van de Velde, A. Th. Schreiber, and J. M. Akkermans. Towards a unification of knowledge modelling approaches. In Jean-Marc David, Jean-Paul Krivine, and Reid Simmons, editors, Second Generation Expert Systems, pages 299-335. Springer-Verlag, Berlin Heidelberg, Germany, 1993.

Generation of a minimal set of test cases that is functionally equivalent to an exhaustive set, for use in knowledge-based system validation

Thomas Abel & Rainer Knauf
Faculty of Computer Science and Automation
Dept. of Artificial Intelligence
Technical University of Ilmenau, Germany
Thomas.Abel@TheoInf.TU-Ilmenau.de
Rainer.Knauf@TheoInf.TU-Ilmenau.de

Avelino Gonzalez
College of Engineering
Dept. of Electrical and Computer Engineering
University of Central Florida, Orlando, FL
ajg@ece.engr.ucf.edu

Abstract

A lot of the created AI Systems find their end in the "wastebasket of academic solutions". The authors are convinced, that the main reason of this dilemma is the fact that nobody can really prove the validity of his system.

This is no surprise, because expectations of users usually can't be formalized. *One* way out of this dilemma is using a test case methodology for system validation. The paper presents a method to find a set of "good" test cases for rule–based systems using sensor data as input.

1 INTRODUCTION

One question that has always posed difficulties for knowledge-based system developers has been how intensively to test the system when validating its performance [3]. A lack of suitable standards has in the past prevented this question from being answered appropriately. One standard that does exist, however impractical as it may be in most cases, is the exhaustive testing of the knowledge-based system. That is, generate a set of test cases which covers all contingencies possible in the operation of the system.
For systems which have more than a few inputs, the combinations of values of these inputs can be very large, thus making exhaustive testing quite impractical [2]. Nevertheless, it is not necessary in most cases to have a truly exhaustive set of test cases, yet still be able to test the system in a *functionally exhaustive* fashion. As stated by Chandrasekaran [1], the test cases should reflect the problems to be seen by the system.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/280 ©1996 FLAIRS

A functionally exhaustive set of test cases can be made considerably smaller than a *naively exhaustive* set by eliminating functionally equivalent input values and combinations of input values which subsume other values. In this paper we describe an automated technique which generates a functionally exhaustive set of test cases. While we certainly do not claim that a functionally exhaustive set of test cases makes it practical to test exhaustively in most cases, it will present the highest standard of testing intensity, from which a developer can step back if necessary. Before we describe our approach, we shall include a formal definition of our terminology and algebra in order to facilitate the understanding of our approach. We will limit our discussion to rule-based systems used for diagnostic or classification tasks, and which use analog-type sensing devices as inputs (i.e., *sensors*).

2 DATA DESCRIPTION

- $S = \{s_1, s_2, \ldots, s_m\}$[1] is a set of variables s_i designating the **input sensor data** and ranging between s_i^{min} and s_i^{max} with a "normal value" s_i^{norm} (a value read by the input device and considered as unfaulty).
- $E \subset \{[t_1, r, t_2] : t_1 \in S, t_2 \in (S \cup \Re), r \in \{<, \leq, =, \neq, \geq, >\}\}$[2] is a set of single **expressions** about sensor data.
- $F = \{f_1, f_2, \ldots, f_n\}$ is a set of output diagnoses (**final conclusions**).
- $I = \{i_1, i_2, \ldots, i_k\}$ is a set of intermediate conclusions (**intermediate conclusions**).
- $H = F \cup I$ is the set of **conclusions**.

[1] Top index symbols designate values of variables; variable names itself own only a bottom index; upper letter symbols designate sets.
[2] $\Re$ is the set of real numbers.

- $R \subseteq \{[l,r] : l \in (E \cup I)^+, r \in F \cup I\}$[3] is a set of diagnostic **rules** in which
  1. the left-hand–side l designates the *if*-part of a rule (a conjunction of elements of $E \cup I$ called *premise*) and
  2. the right-hand–side r designates the *then*-part of a rule (either a final or an intermediate conclusion).

A functionally exhaustive set of test cases is defined as a set of test cases t. A test case t_j is the m-tuple $t_j = [s_1^j, s_2^j, \ldots, s_m^j]$ representable as a compound logic expression $\bigwedge_{k=1}^{m}(s_k = s_k^j)$.

The *functionally exhaustive set* of test cases FES has to meet the following requirements:

1. For each $f_i \in F$ there is at least one $t_j \in FES$.
2. The test cases t_j should be able to reflect the boundary conditions between different final conclusions f_i.
3. The cardinality of FES should be as small as possible.

3 THE APPROACH

The process is based upon the following ideas:

1. break down the range of a sensor into subranges where its values are considered to be equivalent in terms of its effects on the final conclusions,
2. compute an initial set of potential test cases P based upon combinations of values within these subranges,
3. sort these cases into several sets P_i having very distinctive features, and,
4. filter all P_i by eliminating those that are subsumed by others.

Before continuing, however, a definition of what we call *region of influence* must be discussed.

A region of influence is a region formed by the intersection of the projection of the values of the sensors which have a direct effect on a particular final conclusion. Thus, values of the related sensors, which as a group fall within this region, will be able to identify a particular final conclusion. Figure 1 shows the various regions of influence for a two input problem. The lines demarcating the regions from other regions are called the *region boundaries*. The regions are defined not by the inputs themselves, but also by the knowledge (e.g., rules).

[3] M^+ expresses $M \cup (M \times M) \cup (M \times M \times M) \ldots = \bigcup_{i=1}^{\infty} M^i$

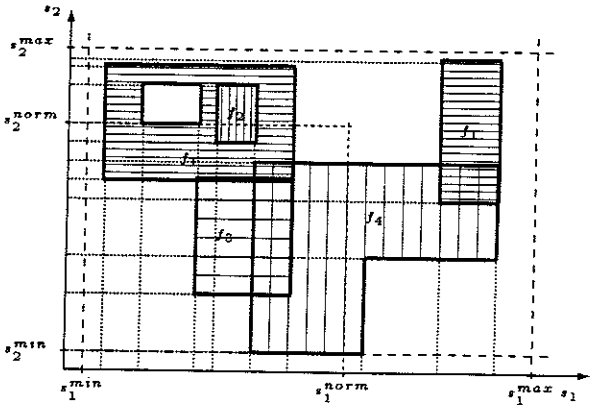


Figure 1: Regions of Influence for a 2 input problem

3.1 Computation of Dependency Sets

The first step is to compute the *rule dependency sets* $R_i \subseteq R$ and the *sensor dependency sets* $S_i \subseteq S$ for each $f_i \in F$. R_i only contains rules $r_k \in R$ and S_i only contains variables $s_k \in S$ on which f_i depends. This is easily carried out by tracing the rules backwards from the final conclusions to the sensor inputs.

3.2 Critical and Δ-values

A particular value of variable s_k is called *critical*, iff that value marks a difference in the effect of s_k on one of its dependent final conclusions.

Critical values are determined by inspection of the rule left-hand sides, where values of s_k are described in relation to either constants or other variables. If an s_k is related to another variable s_j, then all the critical values of s_j must be considered also.

All critical values of one variable s_k form the set of critical values S_k^{crit}. These critical values bound the subranges of functionally equivalent values for each variable, and they are mark out the regions of influence.

In a second step we compute a Δ-value Δs_k for each $s_k \in S$. These values will allow us to surround the boundaries of the resulting regions of influence.

Having all critical value sets S_k^{crit} we establish a Δ-value Δs_k for each s_k generated from S_k^{crit}. This is due to the fact that a small change in a sensor input may distinguish two different final conclusions. These Δ-values and critical value sets give us the first opportunity to minimize enourmously the amount of test cases because they relieve us from generating the theoretically infinite set of possible values a sensor is able to take.

One intuitive approach for computing Δs_k of each variable s_k is to ensure that Δs_k is the half of the smallest difference between any "pair of critical values" that are members of S_k^{crit}. If $S_k^{crit} = \emptyset$, then Δs_k is set to the half of the smallest difference between any pairs of $\{s_k^{min}, s_k^{norm}, s_k^{max}\}$.

We choose the distance from the critical values to be half of the smallest difference for the Δ-values because such a Δ-value makes sure that

1. for all critical values of s_k there will be exactly one upper and one lower boundary value,
2. there is at least one pair of critical values where the upper border value of the one is equal to a lower border of the other. This will allow for the elimination of at least one test case.
3. in any other case, there are exactly two *potential test case values* (*PTC-Values*) between two critical ones.

Upon completion of this, there is a set of Δ-Values $S_\Delta = \{\Delta s_1, \Delta s_2, \ldots, \Delta s_m\}$. With these Δ-values we're able to enclose all the critical values of s_k with potential test case values (*PTC-Values*) as described in the next section.

3.3 Computing the sets of potential test case values

The next step is to compute all sets V_{ij} $(0 < i \leq n, 0 < j \leq m)$ of all *PTC-Values* of s_j which contribute in any way to f_i. V_{ij} contains all *PTC-Values* of $s_j \in S$, which are in any way responsible for f_i.

We have to look for each $s_j \in S$, whether s_j is an element of S_i or not, and – if $s_j \in S_i$ – whether there is a relationship between s_j and a fixed value or another s_k. There are three cases that have to be distinguished:

1. $s_j \notin S_i$

   In that case s_j doesn't contribute to f_i, V_{ij} would be empty. Because each variable of a test case will have to be assigned with a value the *PTC-Value* of s_j is set to its normal one and therefore $V_{ij} = \{s_j^{norm}\}$.

2. $s_j \in S_i$ and s_j is compared with at least one fixed value $\langle value \rangle$

   In that case there are three *PTC-Values* for each value s_j is compared with:

   $$s_j^1 = \langle value \rangle - \Delta s_j, \quad s_j^2 = \langle value \rangle \quad \text{and} \quad s_j^3 = \langle value \rangle + \Delta s_j$$

   and which have to be added to V_{ij}.

3. $s_j \in S_i$ and s_j is compared with another variable s_k

   There are three possible cases here:

   (a) If $V_{ik} \neq \emptyset$, then define three *PTC-Values* for each $s_k^x \in V_{ik} \cap S_k^{crit}$, namely:

   $$s_j^1 = s_k^x - \Delta s_k, \quad s_j^2 = s_k^x \quad \text{and} \quad s_j^3 = s_k^x + \Delta s_k$$

   that have to be added to V_{ij}.

   (b) If $V_{ik} = \emptyset$ and there is a joint interval of s_j and s_k, namely between s^{int_min} and s^{int_max}, then create a *temporary set* TS with (note that $\Delta s_k = \Delta s_j$):

   $$\begin{aligned} TS = \{ & \quad s_j^{min}, \; s_j^{min} + \Delta s_k, \\ & s_j^{norm} - \Delta s_k, \; s_j^{norm}, \; s_j^{norm} + \Delta s_k, \\ & s_j^{max} - \Delta s_k, \; s_j^{max}, \\ & \quad s_k^{min}, \; s_k^{min} + \Delta s_k, \\ & s_k^{norm} - \Delta s_k, \; s_k^{norm}, \; s_k^{norm} + \Delta s_k, \\ & s_k^{max} - \Delta s_k, \; s_k^{max} \quad \} \end{aligned}$$

   Those elements of TS that are between $s^{int\_min} - \Delta s_k$ and $s^{int\_max} + \Delta s_k$ (inclusively) are added to both V_{ij} and V_{ik}.

   (c) If $V_{ik} = \emptyset$ and there is no joint interval of s_j and s_k[4], then both sets V_{ij} and V_{ik} will be equal to the set of their normal values:

   $$V_{ij} = \{s_j^{norm}\} \qquad V_{ik} = \{s_k^{norm}\}.$$

3.4 The set of all potential test cases

Having completed the procedure to calculate *PTC-Values* for all f_i leads us to $n * m$ sets V_{ij} which can can be represented as an $[n \times m]$-Matrix M, whose elements are the sets V_{ij}:

$$\mathbf{M} = \begin{pmatrix} V_{11} & V_{12} & \ldots & V_{1m} \\ V_{21} & V_{22} & \ldots & V_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ V_{n1} & V_{n2} & \ldots & V_{nm} \end{pmatrix}.$$

[4] In this case for each relation $(<, \leq, =, \neq, \geq, >)$ and any value of s_j and s_k we are able to decide whether the expression being treated is true or false.
By the way, such cases indicate a mistake of knowledge acquisition, because such expressions are always true or always false (not depending on the values of the two sensor data variables). The expression can be removed in case of always being true, resp. the whole rule can be removed in case of always being false.

If there is a sensor data variable $s_j \notin S_i$ then the element $V_{ij} = \{s_j^{norm}\}$, otherwise V_{ij} is the set of all *PTC-Values* of s_j.

The i-th row of M describes the dependence of f_i on each variable $s_j \in S$. It can be represented as an m-tuple $M(f_i)$. The j-th element of that tuple is the set V_{ij} of either the more interesting *PTC-Values* of $s_j \in S_i$ or the one normal value of $s_j \notin S_i$ which is needed but not as interesting as the other ones: P will be the union of all test cases $t \in M_i$ that can be created from each $M(f_i)$ by computing the cross product of the sets $V_{i1}, \ldots, V_{im}$:

$$P = \bigcup_{i=1}^{n} M_i = \bigcup_{i=1}^{n} (V_{i1} \times V_{i2} \times \ldots \times V_{im}) = \bigcup_{i=1}^{n} \prod_{j=1}^{m} V_{ij}.$$

The global aim is to create the FES by minimizing P as much as possible.

3.5 Minimizing the set of all potential test cases

The huge cardinality of P forces us to minimize the amount of test cases and to find the minimized set of necessary ones (FES). To reduce the cardinality of P we first sort P into $n+1$ different subsets P_i $(0 \leq i \leq n)$ and then we minimize all the subsets P_i $(0 < i \leq n)$ separately, through which sets $P_i^\star$ will be generated. FES will be the union of each $P_i^\star$.

The first step - Sorting P into $n+1$ subsets P_i

Due to the necessity of keeping the information of the relation between any test case t and its conclusion f_i we represent P as an $n+1$-tuple $[P_0, P_1, \ldots, P_n]$, where each P_i $(i > 0)$ contains those test cases $t \in M_i$ being positive ones for f_i and P_0 contains all test cases $t \in M_i$ $(0 < i < n)$ being negative for f_i: We have to check for each $t \in M_i$ whether the knowledge based system creates a positive solution for f_i. If f_i is true, then t will be an element of P_i of positive test cases for the proof of f_i, otherwise t will be an element of the set of negative test cases P_0[5].

Upon completion, P_0 contains all negative test cases $t \in P$, and the sets P_i $(i > 0)$ contain all positive test cases $t \in M_i$. So we have sorted the set P into $n+1$ dijunctive subsets P_i.

Of course until now the cardinality of P isn't decreased, but this procedure will help us in the second step.

[5]It's possible as well to collect all the test cases being a member of M_i which don't imply malfunction f_i but f_j in the corresponding set P_j. This would lead to a smaller set P_0 and more or less larger sets P_j. Nevertheless we won't follow this way here. We think that these cases are useful for further knowledge acquisition or improvement phases of the AI system.

The second step - Minimizing all the sets P_i

The approach for minimizing each P_i individually (with the exception of P_0) is as follows:

1. Segregate the largest possible subset P_{seg} of m-tuples which differ in only one value (for instance the value of the variable s_j) from the set P_i.
2. With this segregated subset P_{seg}, look through R_i and gather all expressions in which s_j is compared to any value or another variable s_k. The purpose is to find values of s_j that subsume other ones (depending on the used relations). Therefore all potential test cases that are subsumed by other ones will be removed from that subset. The remaining set is P_{good}.
3. Let the new set P_i be the minimized set:
$$P_{i\_new} := (P_{i\_old} \setminus P_{seg}) \cup P_{good}$$
4. Repeat this (i.e., go to the top item) with P_{i_new} until no subset P_{seg} is creatable from P_{i_new}.
5. The minimized subset $P_i^\star$ is the set P_{i_new} computated in step (3).

The above procedure should be carried out for each of the sets P_i. Upon completion of this procedure, there are only 1 or 2 test cases per s_j and per expression being part of a rule's premise, which describes a relation between s_j and a certain value or another variable.

In the end, the minimized set of test cases able to test the knowledge based system exhaustively can be written as an n-tuple $[P_1^\star, \ldots, P_n^\star]$ and the minimized set of all test cases looked for is the union of all sets $P_i^\star$ $(0 < i \leq n)$:

$$FES = \bigcup_{i=1}^{n} P_i^\star.$$

What to do with P_0?

The elements of P_0 describe a non-faulty system's behaviour, faulty states of the system undetected by the knowledge based system or faulty states of the system detected as having another f_j than the considered f_i.

Our tendency is to ignore these negative test cases, as they do not explicitely test the ability of the knowledge based system to identify problems or classsify into classes. However, it could be argued that negative test cases are as important as positive ones, as a misdiagnosis by ommission (not identifying something when it should) can be at least as serious as identifying the wrong problem. We did not deal with

this issue during the course of this investigation, but feel strongly that it should be part of future continuation of this work. In fact, [4] also shows a technique for developing negative test cases. One approach that could be taken is that negative cases for some malfunctions become positive cases for others. The only requirement would be to keep track of which test case is what to which conclusion, something that could be easily done.

By the way, if we imagine all the test cases $t \in FES$ as a map, then $P_0 \not\subseteq FES$ can be considered as an "ocean of ignorance" in the technical system's "map of malfunctions". All the known test cases $t \in P_i$ $(t > 0)$ of one known malfunction f_i would be situated "on land" just like a country with borders to other countries (i.e., known malfunctions) or with a coast to the ocean. All the test cases $t \in P_0$ would be situated in the "ocean" itself. The shores of that "ocean" are the borders to the regions of influence of known conclusions. In that case there is a chance to acquire new knowledge about the technical system by using P_0 for finding "isles of new, unknown or unconsidered conclusions in the ocean of ignorance". But this topic should be discussed in a future article.

4 EVALUTION OF PROCEDURE

In order to evaluate the technique described above, we designed an example problem consisting of 6 sensor inputs, 12 final conclusions, and 19 rules. The relationships between the sensors and the final conclusions ranged from 2 sensors affecting one final conclusion to 5 sensors to one final conclusion. This provided a variety of complexity which was more realistic than it was worst case.

The total set of test cases generated initially by the described procedure numbered 2394 cases. After minimizing the set P, P* was reduced to 103 test cases, a more manageable number. We feel that these test cases are functionally equivalent to the 2394, the only difference being that the former number contains many test cases which subsume one another, and are thus redundant. The negative test cases were also removed.

We furthermore feel that these test cases also represent an exhaustive coverage of the knowledge-based system as each final conclusion was tested with all combinations of sensor values that affect it within the significant ranges.

5 SUMMARY AND CONCLUSION

The procedure described showed the ability to create a set of test cases that is capable of testing a knowledge-based system exhaustively, yet doing so in a minimal fashion to avoid repetitious effort. This sets the gold standard for test case validation of a knowledge-based system, something which has been lacking in the literature. Yet, it does it in a manageable fashion.

A drawback of the approach is its high time complexity, but that price can and should be paid, because

1. the procedure of finding test cases is of a large complexity, but it isn't as time critical as expected, because computers do that task without any human assistance nearly "over night", but
2. humans having the task to judge about the results of test cases are usually not willing to spend plenty of time for doing such a job, and
3. work that can be done by "dumb" computers is mostly cheaper than work that has to be done by "smart" experts.

We are not suggesting that all knowledge-based systems should be tested using this test case set. In many situations, it would be unnecessary and in others, the negative test cases would have to be added to avoid errors of ommission. The appropriate testing intensity for a knowledge-based system is a topic of future research, and it should be based on the customer's criteria as defined in the specifications. The availability of a gold standard, however, permits a set level of testing intensity from which the developer can back off if appropriate.

References

[1] Chandrasekaran, B.: *On Evaluating AI Systems for Medical Diagnosis*; In: *AI Magazine*, 4:2, pp. 34-37.

[2] Gonzalez, A.J.; Dankel, D.D.: *The Engineering of knowledge-based Systems – Theory and Practice*; Englewood Cliffs, N.J.; Prentice Hall, 1993.

[3] O'Keefe, R.M.; O'Leary, D.E.: *Expert System Verification and Validation: A Survey and Tutorial*; In: *Artificial Intelligence Review*, Vol. 7, pp. 3-42.

[4] Zlatareva, N.P.: *A Framework for Knowledge based System Verification, Validation and Refinement*; Proceedings of the fifth Florida Artificial Intelligence Research Symposium, St. Petersburg, FL, 1992, pp.10-14.

Surprises in Probabilistic Reasoning

Ahmed Y. Tawfik and Eric Neufeld
Department of Computer Science
University of Saskatchewan
{tawfik,eric}@cs.usask.ca

Abstract

This paper introduces surprise as a useful tool for probabilistic reasoning. A quantitative measure of surprise 'surprise index' serves as a measure of goodness of probability estimates. As such, estimate revision is initiated when actual outcomes are very surprising. Surprises relate easily to other concepts in artificial intelligence such as entropy and nonmonotonic reasoning.

1 Introduction

In probabilistic reasoning, probabilities are ascribed to each proposition. These probabilities represent the degree of belief in the truth of the proposition. A very low probability represents a proposition believed to be false while probabilities closer to one are ascribed to proposition believed to be true. A surprise occurs when the actual state of the world defies our expectation. Surprises can happen in two situations: first as a result of the random nature of the process even an extremely unlikely outcome is still possible (e.g. a dice thrown ten times may come to rest showing the same number each time with probability 1.653817e-08), and the second situation results from a poor probability estimate (e.g. the actual dice has the same number on all faces; the actual probability is one while the estimate is 1/6). In both cases, the person throwing the dice would be surprised. The surprise prompts a belief revision: the person examines the dice. If it turns out to be a defective dice, the beliefs are updated.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/285 ©1996 FLAIRS

The role of surprise is therefore to motivate belief revision. Gärdenfors [1] suggests that human rational thinking uses belief revision as a response to surprises as illustrated by the following story:

> Oscar believed that his wedding ring was made of gold. However, one day, while repairing his boat, he noticed that the sulfuric acid stained his ring. This surprised him because he knew from his school chemistry that sulfuric acid is not supposed to affect gold. He now thinks that his ring is not made of gold.

Simon [10] describes the importance of surprises in scientific discovery by stating that a considerable number of [Nobel] prizes are given to people who had the good fortune to experience a surprise. That is how X-rays, radium, penicillin and dozens of other things were discovered. To exploit surprise, we have to detect it, identify its range and conditions and contrast it with our assumptions to get a better understanding of the process.

In artificial intelligence, few works have studied surprise. Goldszmidt and Pearl [2] rank beliefs according to their potential surprise (or normality) to perform qualitative probabilistic reasoning. Tawfik and Neufeld [11] use surprise to trigger more elaborate investigation in a diagnostic application. Some earlier works have discussed the applicability of surprises to rational agents. Hsia [5] states that any rational agent can be surprised. His argument relies on a proof that uses Dempster- Shafer theory of evidence and a logical definition of a surprise. Hsia argues that probability theory cannot capture the intuitive notion of surprise. This conclusion results from the logic definition of surprise and a strong rationality assumption.

In this work, we study how can intelligent systems exploit surprises. In Section 2, we consider the question

of measuring and detecting a surprise. Section 3 and 4 relate surprise to two established concepts: entropy and nonmonotonic reasoning. Section 5 shows how to reason about surprises.

2 Measuring Surprises

A surprise is the occurrence of an event that is unlikely to happen. A surprise index is a measure of the degree of surprise associated with the occurrence of an event. A good surprise measure must allow us to detect a surprise when it happens. The magnitude of the measure must reflect that of the surprise and it has to allow us to compare surprises.

Shackle [9] suggests the use of a rank based surprise measure. The probabilities of possible events are sorted in descending order. Each event is assigned a surprise measure corresponding to its rank in the sorted list. This measure is similar to the surprise measure proposed by Goldszmidt and Pearl [2]. The problem with this type of surprise measure is that it provides only an indicator but the numeric value of the surprise index has no significance. This index allows us to compare surprises only within the same distribution.

The observation that less likely outcomes are more surprising than the more probable ones has prompted several researchers to express the degree of surprise as the inverse of the probability [7], or requiring the surprise index to be inversely proportional to the probability [1]. Using this simple surprise measure results in some problems. For example, the occurrence of an event from a set of n events with equal likelihood may be very surprising or not depending on n. In most situations such occurrence should not be surprising. It is also impossible to compare surprises resulting from two different distributions if we use this measure for surprise.

Weaver [13] proposes a mathematical formulation of a surprise index. His surprise index is the ratio of the expected value of the probability to the probability of the event that actually occurs or

$$Surprise = \frac{\sum_i (p_i)^2}{p_r}.$$

The summation is over an exhaustive mutually exclusive set of possible outcomes and p_r is the probability of the event that actually occurs. The surprise index values range from zero to infinity. A value between zero and one corresponds to the case of a likely outcome occurring (i.e. no surprise). Values greater than one indicate a surprise, the larger the index the more astonishing. The Weaver index avoids the problems of using the inverse only by multiplying it by the expected value of the probability.

The Weaver surprise index is a better indicator of the degree of surprise than the probability of the event by itself because it allows us to compare surprises belonging to different distributions. The following example illustrates how this surprise index works. While it is not surprising at all that *someone* wins the lottery it is a big surprise if I win. Let there be n tickets sold to an equal number of persons and let us assume that the draw is among the sold tickets. The probability of winning for each individual is then $1/n$. Let X be a person unknown to us. The surprise index for the proposition X *won the lottery* is then

$$Surprise = \frac{n(1/n)^2}{1/n} = 1.$$

The proposition I *won the lottery* has only two possible outcomes actually winning or not with probabilities $1/n$ and $(1 - 1/n)$. For these two outcomes, the surprise associated with winning is

$$Surprise = \frac{(1/n)^2 + (1 - 1/n)^2}{1/n} = n - 2 + 2/n.$$

For a large n the surprise is very large, while there is no surprise at all if $n = 1$ (why be surprised to win if there is only one ticket sold?). The surprise index measures how unlikely an outcome is compared with other possible outcomes. It is worth noting that the representation dependence of the surprise index is desirable because it captures a common-sense reasoning pattern as illustrated in the previous example.

Weaver's surprise index assumes exhaustive mutually exclusive events. Suggestions to alleviate the constraints implied by these assumptions include the use of a scale factor for compensation in situations where the probabilities of a set of mutually exclusive outcomes do not add up to one [11].

An information theoretic formulation of surprise [12] favors the use of a logarithmic surprise index. The surprise associated with multiple independent outcomes is more easily expressed in terms of such surprise index. Watanabe proposes to measure the surprise as

$$Surprise = -\log(p_r).$$

This measure suffers from the same problems as the inverse of the probability because it does not take into account the distribution.

Good [4] proposes a different form for the logarithmic surprise index. This surprise index takes the form

$$Surprise = \sum_i p_i(\log(p_i) - \log(p_r)).$$

This measure results in 0 or $-ve$ values for expected events and $+ve$ values for surprising ones. Moreover it does not require mutual exclusion. The surprise associated with the occurrence of two independent outcomes $r1$ and $r2$ is given by

$$Surprise_{r_1,r_2} = \sum_i p_i \log(p_i) - \log(p_{r1}) - \log(p_{r2}).$$

This additivity is one of the advantages of this surprise measure. For any number of independent events, the logarithmic surprise of the conjunction can be calculated similarly. In the rest of this paper, the term surprise index is used to denote this index.

3 Surprise and Entropy

Entropy is a basic notion in information theory and physical sciences. It is generally used to measure the amount of information or ignorance in a stochastic system. It is used in artificial intelligence as a relevance criterion [7] and entropy minimization helps in dealing with uncertainty in many systems [3]. The entropy in a system is given by

$$H = -\sum_i p_i \log(p_i).$$

Comparing this expression with that for the surprise index, we realize that

$$Surprise = -\sum_i p_i log(p_r) - H.$$

The entropy is therefore a measure of potential surprises. In fact, the entropy is minimum when all the probabilities are equal. In this case, the surprise is nil. Higher entropy indicates that high surprises are possible.

Surprise index shares some properties with entropy. Both quantities are representation dependent. We have illustrated this earlier using a lottery example for Surprise. The difference in surprise for I won the lottery as opposed to *someone* won is due to a different representation. When considering my chances of winning, I consider the probability that I win or loose which are $1/n$ and $1 - 1/n$ respectively. For unknown persons, they all have equal opportunities of winning $1/n$. The same holds for entropy.

No surprise can occur if all outcomes are equiprobable. Entropy is minimum for equiprobable outcomes, also the amount of information is at its lowest. This means that surprises do not happen to the ignorant. Pasteur said 'Accidents only happen to the prepared mind.' The implication of this observation is that we cannot start without any knowledge and rely on surprise to point us to update our beliefs. We need some knowledge to start with.

By analogy to conditional entropy, we define the conditional surprise index. The conditional surprise when r occurs knowing that y is true is given by

$$Surprise_{r|y} = \sum_i p_{i|y}(\log(p_{i|y}) - \log(p_{r|y}).$$

It measures the surprise associated with the occurrence of i in the presence of y. The conditional surprise can be numerically very different from the unconditional one if there is a strong dependence between r and y.

4 Surprises and Nonmonotonic Reasoning

Nonmonotonic logics generally assume as a default the more likely outcome [8]. For example, birds are assumed to fly. Knowing that Tweety is a bird, it is therefore possible to assume that it flies. Similarly in probabilistic reasoning, the probability $P(flies(Tweety)|Bird(Tweety))$ is high. Consider if we somehow knew that Tweety does not fly. This would result in a conflict forcing the default assumption to be retracted. In probabilistic reasoning, we simply assume that Tweety belongs to a minority of birds that do not fly with probability $1 - P(flies(Tweety)|Bird(Tweety))$. Consider now that on a rainy day, birds do not fly. The minority explanation would not stand as more and more non-flying birds are observed. An automated reasoner that have no way of relating rain and the flight of birds would be surprised to see such a number of non-flying birds. This surprise would prompt a belief revision

which would hopefully allow the reasoner to find the relation between rain and the flight of birds. Such relation can be discovered if the reasoner contrasts his previous information about weather and flying birds with the current observations. Learning techniques able to perform such deductions from statistical information are known.

Nonmonotonic reasoning generally lacks the capability of performing surprise based belief revision. Such feature may be required in a number of situations. Achieving such a capability may require some form of hierarchical reasoning such that surprising events are explained at a lower level of the hierarchy where the details are available. In the absence of such a hierarchical representation, surprises may remain unexplained. Lifschitz 's miracles [6] leave surprises unexplained but prefer an explanation that minimizes the miracles (or surprises).

5 Reasoning with Surprises

Reasoning with surprise involves estimating the surprise associated with each finding (or observation). Non surprising events are ignored. Highly surprising events trigger a more detailed analysis and belief revision. Moderately surprising events are accumulated according to an additive storage model with a release function. This model acts like a leaky bucket; it becomes empty after some time if no further surprises occur. Each belief is assigned its own storage. Distinction between high, moderate and low surprises involves choosing some thresholds. Different thresholds are set for different beliefs. The setting of these thresholds is done according to a decision theoretic criterion aiming at minimizing the following cost function

$$cost = P(Surprise, BadEstimates)cost(Revision) + P(NoSurprise, BadEstimates)cost(BadEstimates)$$

To illustrate how this decision theoretic threshold setting work consider the following example:

A process has six possible outcomes that can occur according to a binomial distribution with probabilities 1/32, 5/32, 10/32, 10/32, 5/32 and 1/32. The cost of belief revision is 100.00 dollars. A system malfunction may occur with probability 0.25 causing all the outcomes to have equal probabilities 1/6. A loss of 1000.00 dollars is incurred when the system is not repaired immediately. The outcome are exhaustive and mutually exclusive, we can then use Weaver's surprise

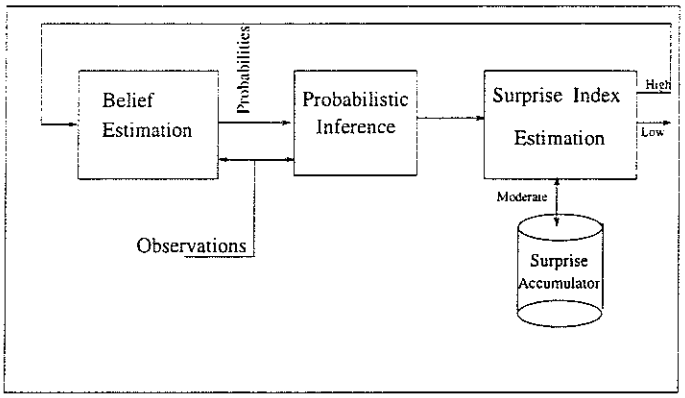


Figure 1: Reasoning System Structure

index for convenience. If the probability estimates are correct, then the surprise index can take the values 0.7, 1.575 and 7.5 with probabilities 10/16, 5/16 and 1/16 corresponding to the occurrence of outcomes with probabilities 10/32, 5/32 and 1/32 respectively. If we set the surprise threshold for revision to 1.0, we will have unnecessary revisions performed with probability 3/8. This contributes a cost of 37.50 dollars + 0.15 * 1000.00 because it is still possible that the system fail and produces the most likely outcome. The total cost is then 187.50 dollars. Comparing this cost with the cost of setting the threshold to say 1.6, we have 1/16 probability of a false surprise resulting in a belief revision cost of 6.25 dollars. The cost of ignoring surprises is now 191.6 , giving a total cost of 197.85. In this case, setting the surprise to 1.6 is more economic. Had the cost of ignoring the surprise been 10,000.00 dollars, setting the threshold to 1.0 would have been more advantageous.

The above example illustrates that the proper setting of thresholds involves a rather deep analysis of the problem. This analysis may be hard in some cases but we think that a reasonable threshold setting can be done subjectively depending on the utility of revision as opposed to its cost. A low threshold results in more revisions while a higher one results in potential ignoring some surprises.

The structure of an inference engine capable of reasoning with surprises is illustrated in figure 1. Such an engine consists of a probability estimation module. This module generates probability estimates using a statistical causal learning technique based on previous observations. The probabilities generated are then fed into a probabilistic reasoning module. This module can be a Bayesian network or any similar engine

that performs probabilistic reasoning based on observations. The surprise evaluation module is given a set of possible outcomes along with their probabilities and a surprise index is calculated for the actual outcome. A low surprise index indicates that this outcome is expected, a moderately surprising outcome is noted for future follow-up and a highly surprising outcome is immediately passed for further investigation. The investigation would start by checking assumptions and their credibility. Utility, credibility and surprise constitute the basic block of the proposed belief revision system. Once surprises, the decision regarding whether to proceed with belief revision or not is utility driven. When beliefs are examined, more credible belief are preferred. To illustrate how the proposal works, let us consider Oscar's story mentioned in the introduction. Oscar was surprised to noticed that his gold ring was stained by the acid. He has perceived some utility in revising his beliefs. He has ascribed more credibility to his school chemistry course than the jeweler. He then concluded that the ring is not made of gold.

6 Conclusions

Surprises are introduced here as a useful tool for belief revision. The type of belief revision resulting from observing a surprising outcome is different from the belief revision based on evidence. In the former case the joint probability distribution is to be examined while the latter involves updating individual belief using the same joint probability distribution. Different surprise indexes give a similar ranking of surprises within the same distribution but the surprise index used here allows us to compare surprises generated from different distributions. Surprise as a concept is closely related to entropy. It can also be used to explain some of the limitations of nonmonotonic reasoning. This work also proposes a general structure for inference engines that can handle surprises.

Acknowledgements

The first author acknowledges the support of the University of Saskatchewan. Research of the second author is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] P. Gardenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States.* MIT Press, Cambridge, MA, 1988.

[2] M. Goldszmidt and J. Pearl. Qualitative probabilities for default reasoning, belief revision, and causal modeling. *Artificial Intelligence*, 1995. to appear.

[3] M. Goldzsmidt, P. Morris, and J.Pearl. A maximum entropy approach to nonmonotonic reasoning. *Ieee transactions on pattern analysis and machine Intelligence*, 15(3):220, Mar 1993.

[4] I.J. Good. *Good Thinking: The Foundations of Probability and Its Applications.* University of Minnesota Press, Minneapolis, 1983.

[5] Y. Hsia. A rational agent can be surprised no matter what. In *Proc. of the ninth Canadian Conference on Artificial Intelligence AI'92, Vancouver, Canada*, pages 106– 112, 1992.

[6] V. Lifschitz and A. Rabinov. Miracles in formal theories of action. *Artificial Intelligence*, 38:225–237, 1989.

[7] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kauffmann, San Mateo, CA, 1988.

[8] D. Poole. What the lottery paradox tells us about default reasoning. In *First International Conference on the Principles of Knwoledge Representation and Reasoning*, 1989.

[9] G. Shackle. *Expectations in Economics.* Cambridge University Press, UK, 1949.

[10] H. Simon. *Econoics, Bounded Rationality and the Cognitive Revolution.* Edward Elgar, Hants, England, 1992.

[11] A. Tawfik and E. Neufeld. Lifetime information for model-based diagnosis. In *Proceedings of Time-95: International Workshop on Temporal Knowledge Representation and Reasoning, Melbourne, Florida*, 1995.

[12] S. Watanabe. *Knowing and Guessing.* John Wiley, New York, NY, 1969.

[13] W. Weaver. Probability, rarity, interest and surprise. *Scientific Monthly*, 67:390–392, 1948.

Implementing Belief-network Inference in Real-World Systems

Adnan Darwiche and Gregory Provan
Rockwell Science Center
1049 Camino Dos Rios
Thousand Oaks, CA 91360
{*darwiche, provan*}*@risc.rockwell.com*

Abstract

We describe a new paradigm for implementing inference in belief networks, which relies on compiling a belief network into an arithmetic expression called a *Query DAG* (Q-DAG). Each non-leaf node of a Q-DAG represents a numeric operation, a number, or a symbol for evidence. Each leaf node of a Q-DAG represents the answer to a network query, that is, the probability of some event of interest. It appears that Q-DAGs can be generated using any of the algorithms for exact inference in belief networks — we show how they can be generated using the clustering algorithm. The time and space complexity of a Q-DAG *generation* algorithm is no worse than the time complexity of the inference algorithm on which it is based; that of a Q-DAG on-line *evaluation* algorithm is linear in the size of the Q-DAG, and such inference amounts to a standard evaluation of the arithmetic expressions it represents. The main value of Q-DAGs is in reducing the software and hardware resources required to utilize belief networks in on-line, real-world applications. The proposed framework also facilitates the development of on-line inference on different software and hardware platforms given the simplicity of the Q-DAG evaluation algorithm.

INTRODUCTION

Consider designing a car to have a self-diagnostic system that can alert the driver to a range of problems. One approach to building such a diagnostic system is to put a diagnostic belief network, along with belief-network inference code, onto the car's computer (Pearl 1988). We have encountered a number of difficulties when using this approach to embody belief network technology in industrial applications, such as having to provide the technology on multiple platforms, some of which had stringent memory limitations. This is problematic because belief network algorithms are not trivial to implement, especially when optimization is crucial, and porting these algorithms to multiple platforms and languages would have been prohibitively expensive, time-consuming and demanding of qualified manpower.

To overcome these difficulties, we have devised a very flexible approach for implementing belief network systems, which splits inference into separate offline and online steps. This split is based on the following observation. Almost all the work performed by standard algorithms for belief networks is independent of the specific evidence gathered about variables. For example, if we run an algorithm with the battery-sensor variable set to *low* and then run it later with the variable set to *dead*, we find almost no algorithmic difference between the two runs. That is, the algorithm will not branch differently on any of the decisions it makes, and the only difference between the two runs is the specific arguments to the invoked numeric operations. Therefore, one can apply a standard inference algorithm on a network with evidence being a *parameter* instead of being a specific value. The result returned by the algorithm will then be an arithmetic expression with some parameters that depend on specific evidence. This parametrized expression is what we call a Query DAG, an example of which is shown in Figure 2.

The approach we are proposing consists of two steps. First, given a belief network, a set of variables about which evidence may be collected (evidence variables), and a set of variables for which we need to compute probability distributions (query variables), a Q-DAG is compiled off-line, as shown in Figure 1. The compilation is typically done on a sophisticated software/hardware platform, using a traditional belief network inference algorithm in conjunction with the Q-DAG compilation method. This part of the process is far and away the most costly computationally. Second, an on-line system composed from the generated Q-DAG and an evaluator specific to the given platform is used to evaluate the Q-DAG. Given evidence, the parameterized arithmetic expression is evaluated in a straightforward manner using simple arithmetic operations rather than complicated belief network infer-

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/290

ence. The computational work needed to perform this on-line evaluation is so straightforward that it lends itself to easy implementations on different software and hardware platforms.[2]

It is important to stress the following properties of the proposed approach. First, declaring an evidence variable in the compilation process does *not* mean that evidence must be collected about that variable on-line (this is important because some evidence values, e.g., from sensors, may be lost in practice). It only means that evidence *may* be collected; therefore, one can declare all variables to be evidence if one wishes. Second, a variable can be declared to be both evidence and query. This allows one to perform value of information computations to decide whether it is worth collecting evidence about a specific variable. Third, the size of a Q-DAG is linear in the number of evidence variables; therefore, this is not a simple enumerate-all-possible-cases approach. Finally, the time and space complexity for generating a Q-DAG is no worse than the time complexity of the standard belief-network algorithm used in its generation. Therefore, if a network can be solved using a standard inference algorithm, we can construct a Q-DAG for that network.

The following section explains the concept of a Q-DAG using a concrete example. Based on this framework, we show how Q-DAGs can be generated using a clustering algorithm. We conclude by discussing some implications of Q-DAGs and by comparing this work to related work on symbolic probabilistic evaluation.

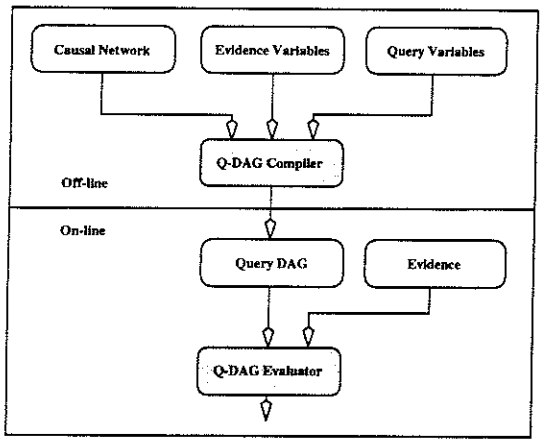


Figure 1: The proposed framework for implementing belief-network inference.

[2]We have implemented Q-DAG evaluators for ADA, C, and LISP platforms, and plan to develop evaluators for Programmable Logic Controllers (PLC), which are used to control manufacturing processes, and are considering dedicated hardware for this purpose.

QUERY DAGs

We now present an example Q-DAG and explain how it can be used to perform probabilistic inference.[3]

Consider the belief network in Figure 2. Suppose that we typically have evidence about variable C and we are interested in the probability of variable B. Figure 3 depicts a Q-DAG for answering such queries.

The Q-DAG has two leaf nodes corresponding to $Pr(B = ON, \mathrm{e})$ and $Pr(B = OFF, \mathrm{e})$, respectively. A root node of the form (V, v) is called an *Evidence Specific Node (ESN)* and its value depends on the evidence collected about variable V on-line.

The value of node (V, v) is 1 if variable V is instantiated to v or is unknown, and 0 otherwise. Once the values of ESN nodes are determined, we evaluate the remaining nodes of a Q-DAG using numeric multiplication and addition. The numbers that get assigned to leaf nodes as a result of this evaluation are the answers to the queries represented by these leaf nodes.

For example, suppose that the evidence we have is $C = ON$. Then ESN node (C, ON) is evaluated to 1 and ESN node (C, OFF) is evaluated to 0. The Q-DAG in Figure 3 is then evaluated as given in Figure 4(a), thus leading to $Pr(B = ON, C = ON) = .3475$, and $Pr(B = OFF, C = ON) = .2725$, from which we conclude that $Pr(C = ON) = .62$. If the evidence we have is $C = OFF$, then we can compute posterior probabilities for B analogously, as given in Figure 4(b).

A Q-DAG evaluator can be implemented using an event-driven, forward propagation scheme. Whenever the value of a Q-DAG node changes, one updates the value of its children, and so on, until no possible update of values is possible. We can also implement an evaluator using a backward propagation scheme where one starts from a query node and updates its value by updating the values of its parent nodes. The forward propagation scheme is more selective in that it will only visit those nodes that need updating as opposed to visiting every node on which the value of a query node depends. The specifics of the application will typically determine the most appropriate method (or combination).

It is important that we stress the level of refinement enjoyed by the Q-DAG propagation scheme and the implications of this on the efficiency of query updates. Propagation in Q-DAGs is done at the arithmetic-operation level, which is contrasted with propagation at the message-operation level (used by many stan-

[3]We will use the following notation for denoting variables and their values. Variables are denoted using uppercase letters, such as A, B, C and variable values are denoted by lowercase letters, such as a, b, c. Sets of variables are denoted by boldface uppercase letters, such as $\mathbf{A}, \mathbf{B}, \mathbf{C}$, and their instantiations are denoted by boldface lowercase letters, such as $\mathbf{a}, \mathbf{b}, \mathbf{c}$. We use $\mathbf{E}$ to denote the set of variables about which we have evidence. Therefore, we use $\mathbf{e}$ to denote an instantiation of these variables that represents evidence.

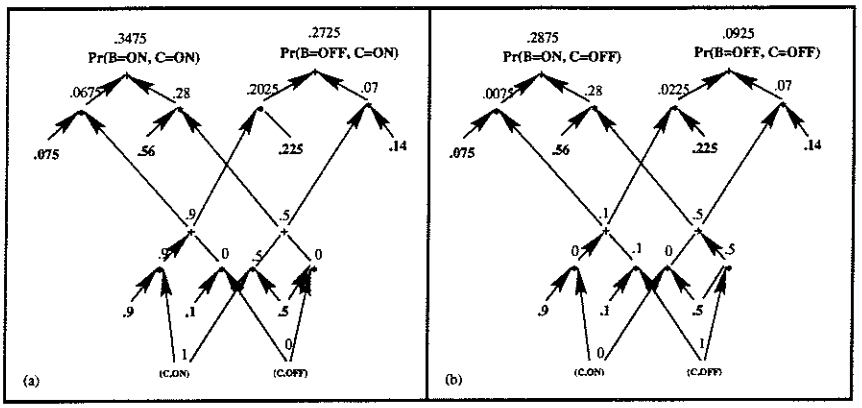


Figure 4: Evaluating the Q-DAG in Figure 3 with respect to two pieces of evidence: (a) $C = ON$ and (b) $C = OFF$.

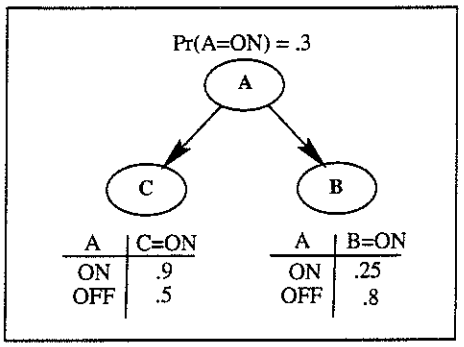


Figure 2: A probabilistic belief network. C is an evidence variable, and we are interested in the probability of Variable B.

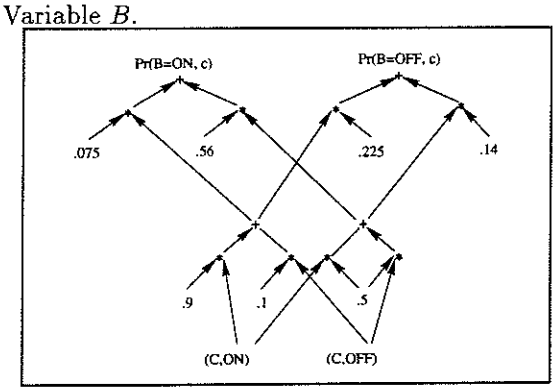


Figure 3: A Query DAG corresponding to the network in Figure 2.

dard algorithms). In (Darwiche & Provan 1995) we argue that Q-DAG propagation is more efficient than message-passing propagation, since it avoids many (unnecessary) operations inherent in message-passing propagation. We also stress that the process of evaluating and updating a Q-DAG is done outside of probability theory and belief network inference.

The construction of a Q-DAG requires the identification of query and evidence variables. This may give an incorrect impression that we must know up front which variables are observed and which are not. This could be problematic in (1) applications where one may lose a sensor reading, thus changing the status of a variable from being observed to being unobserved; and (2) applications where some variable may be expensive to observe, leading to an on-line decision on whether to observe it or not (computing value of information).[4]

Both of these situations can be dealt with in a Q-DAG framework. First, as described earlier, Q-DAGs allow the unknown value $\diamond$ to represent missing evidence which can be used to handle missing sensor readings. Second, a variable can be declared to be both query and evidence. This means that we can incorporate evidence about this variable when it is available, and also compute the probability distribution of the variable in case evidence is not available. This distribution can be used to compute the variable's value of information in order to decide whether to observe the variable (cf. (Darwiche & Provan 1995)).

GENERATING QUERY DAGs

We now discuss how Q-DAGs can be generated using traditional algorithms for exact belief-network inference. In particular, we will show how Q-DAGs can be generated using the clustering algorithm described in (Shachter, Andersen, & Szolovits 1994;

[4]Thanks to Jack Breese and Bruce D'Ambrosio for stressing this point to us.

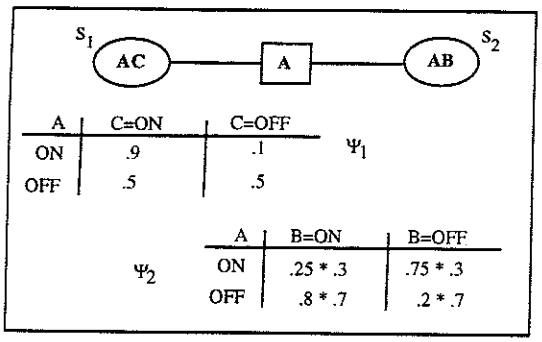


Figure 5: A cluster tree quantified with numbers.

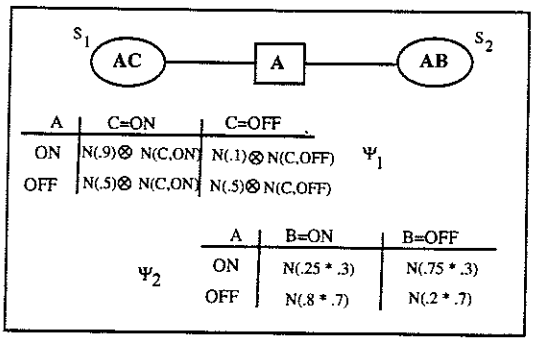


Figure 6: A cluster tree quantified with Q-DAGs. Here $n(.9)$ means a Q-DAG node labeled with .9 (always evaluates to .9) and $n(V, v)$ is an evidence specific node for variable V.

Jensen, Lauritzen, & Olesen 1990). In the full paper (Darwiche & Provan 1995), we show how Q-DAGs can also be generated using the polytree algorithm and cutset conditioning (Pearl 1988; Peot & Shachter 1991).

The clustering method (Shachter, Andersen, & Szolovits 1994; Jensen, Lauritzen, & Olesen 1990) is one of the most important belief network algorithms in current use. We now describe how the algorithm can be modified to generate Q-DAGs and show a concrete example. A more formal treatment, together with a proof of soundness, can be found in the full paper (Darwiche & Provan 1995).

To generate Q-DAGs using the clustering method, we have to peform two steps. First, we have to modify the initialization of potential functions so that the cluster tree is quantified using Q-DAG nodes instead of numeric probabilities (that is, each probability is replaced with a Q-DAG node that will evaluate to that probability). Second, we have to replace numeric addition and multiplication in the algorithm by analogous functions that operate on Q-DAGs. In particular:

1. Numeric multiplication * is replaced by an operation $\otimes$ that takes Q-DAG nodes $n_1, \ldots, n_i$ as arguments, constructs and returns a new node n with label $*$ and parents $n_1, \ldots, n_i$.
2. Numeric addition + is replaced by an operation $\oplus$ that takes Q-DAG nodes $n_1, \ldots, n_i$ as arguments, constructs and returns a new node n with label + and parents $n_1, \ldots, n_i$.

Before we give an example, realize that when the QDAG algorithm is applied, we do not have evidence e but instead we have a set of evidence variables $\mathbf{E}$ about which we will collect evidence. Therefore, the Q-DAG algorithm will not compute an answer to a query $Pr(x, \mathbf{e})$ for some variable X, but instead will compute a Q-DAG node, $Qnode(X = x)$, that will evaluate to $Pr(x, \mathbf{e})$ once we instantiate variables $\mathbf{E}$ to e.

Consider the belief network in Figure 2 for an example. We have only one evidence variable in this example, C. We are interested in generating a Q-DAG for answering queries about variable B, that is, queries of the form $Pr(b, \mathbf{e})$. Figure 5 shows the cluster tree for the belief network in Figure 2, where the tables contain the potential functions needed for the probabilistic clustering algorithm. Figure 6 shows the cluster tree again, but the tables contain the potential functions needed by the Q-DAG clustering algorithm. Note that the tables are filled with Q-DAGs instead of numbers; we use the notation that $n(\cdot)$, means a Q-DAG node labeled with $\cdot$, where $\cdot$ is a number and $n(V, v)$ is an evidence specific node for variable V.

We now apply the Q-DAG version of the clustering algorithm. The message M_{12} is computed by summing the potential function Ψ_1 over all possible values of variable C, $M_{12} = \bigoplus_C \Psi_1$, which leads to

$$M_{12}(A = ON) = [n(.9) \otimes n(C, ON)] \oplus [n(.1) \otimes n(C, OFF)];$$

$$M_{12}(A = OFF) = [n(.5) \otimes n(C, ON)] \oplus [n(.5) \otimes n(C, OFF)].$$

The posterior distribution over cluster S_2, P_2, is computed by factoring in the message M_{12}, $P_2 = \Psi_2 \otimes M_{12}$, which leads to, for $B = ON$,

$$P_2(A = ON, B = ON) = n(.25) \otimes n(.3) \otimes [[n(.9) \otimes n(C, ON)] \oplus [n(.1) \otimes n(C, OFF)]];$$

$$P_2(A = OFF, B = ON) = n(.8) \otimes n(.7) \otimes [[n(.5) \otimes n(C, ON)] \oplus [n(.5) \otimes n(C, OFF)]].$$

The Q-DAG node $Qnode(b)$ for answering queries of the form $Pr(b, \mathbf{e})$ is computed by summing the posterior P_2 over variable A, $Qnode(b) = \bigoplus_{S_2 \setminus \{B\}} P_2$, leading to, for $Qnode(B = ON)$,

$$[n(.25) \otimes n(.3) \otimes [[n(.9) \otimes n(C, ON)] \oplus [n(.1) \otimes n(C, OFF)]]] \oplus [n(.8) \otimes n(.7) \otimes [[n(.5) \otimes n(C, ON)] \oplus [n(.5) \otimes n(C, OFF)]]].$$

The resulting Q-DAG is equivalent to the one depicted in Figure 3.

The formal statement of the Q-DAG generation algorithm and its proof of correctness can be found in (Darwiche & Provan 1995).

Most algorithms for exact inference in belief networks can be adapted to generate Q-DAGs. In general, an algorithm must satisfy a key condition to be adaptable for computing Q-DAGs as we suggested above. The condition is that the computational complexity of the algorithm should never depend on the specific evidence obtained, but should only depend on the variables about which evidence is collected. That is, whether variable E is instantiated to value v_1 or value v_2 should not affect the computational performance of the algorithm. Only whether variable E is instantiated or not should matter.

The computational complexity of the algorithm for generating Q-DAGs is determined by the computational complexity of the clustering algorithm. In the full paper we show that the space complexity of a Q-DAG is no worse than the time complexity of the clustering algorithm. Another complexity result is that size of a Q-DAG is linear in the number of evidence variables. Specifically, each evidence variable E will only add m evidence-specific nodes to the Q-DAG, where m is the number of values that variable E can take. This is very important to stress because without this complexity guarantee it may be hard to distinguish between the proposed approach and a brute-force approach that builds a big table containing all possible instantiations of evidence variables together with their corresponding distributions of query variables. Finally, we note that similar complexity claims can be made with respect to other algorithms for generating Q-DAGs.

DISCUSSION

We have introduced a new paradigm for implementing belief-network inference that is oriented towards real-world, on-line applications. The framework utilizes knowledge of query and evidence variables in an application to compile a belief network into a semi-arithmetic expression called a *Query DAG* (Q-DAG). Each non-leaf node of a Q-DAG represents a numeric operation, a number, or a symbol for a node that depends on available evidence. Each leaf node of a Q-DAG represents the answer to a network query, that is, the probability of some event of interest. Inference on Q-DAGs is linear in their size and amounts to evaluating the arithmetic expressions they represent.

A most important point to stress about the work reported here is that it is *not* proposing a new algorithm for belief-network inference. What we are proposing is a paradigm for implementing belief-network inference that is orthogonal to the standard inference algorithms and is engineered to meet the demands of real-world, on-line applications. This class of applications is typically demanding since it places strong restrictions on processing time, platforms and computational resources. To address these real-world constraints, we are proposing that one compiles a belief network into a Q-DAG as shown in Figure 1 and uses a Q-DAG evaluator for on-line reasoning. This reduces the required memory to that needed for storing a Q-DAG and its evaluator. It also reduces the required software to that needed for implementing a Q-DAG evaluator, which is very simplistic as we have seen earlier.

Our proposed approach still requires a belief-network algorithm to generate a Q-DAG, but it makes the efficiency of such an algorithm less critical. We have shown how clustering and conditioning algorithms can be used for this purpose, but other algorithms such as SPI (D'Ambrosio 1994a; 1994b; Shachter, D'Ambrosio, & del Favero 1990) can also be used.

This approach shares some commonality with other methods that symbolically manipulate probability expressions, like SPI; it differs with SPI on the objective of such manipulations and, hence, on the results obtained. SPI explicates the notion of an arithmetic expression to state that belief-network inference can be viewed as an expression-factoring operation. This allows results from optimization theory to be utilized in belief-network inference. The Q-DAG is a factored expression, but we do not explicate it for algorithmic purposes as does SPI. We explicate an arithmetic expression to explicate and formalize the boundaries between on-line and off-line inference, with the goal of identifying the minimal piece of software that is required on-line.

References

D'Ambrosio, B. 1994a. Algebraic techniques for efficient inference in bayes networks. In *Proc. RSSC*, 21–28.

D'Ambrosio, B. 1994b. Local Expression Languages for Probabilistic Dependence. *International Journal of Approximate Reasoning* 11:1–15.

Darwiche, A., and Provan, G. 1995. Query dags: A practical paradigm for implementing on-line causal-network inference. Technical Report 95-86, Rockwell Science Center, Thousand Oaks, Ca.

Jensen, F. V.; Lauritzen, S.; and Olesen, K. 1990. Bayesian updating in recursive graphical models by local computation. *Comp. Stats. Quarterly* 4:269–282.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, Inc., San Mateo, California.

Peot, M. A., and Shachter, R. D. 1991. Fusion and propagation with multiple observations in belief networks. *Artificial Intelligence* 48(3):299–318.

Shachter, R.; Andersen, S.; and Szolovits, P. 1994. Global Conditioning for Probabilistic Inference in Belief Networks. In *Proc. Tenth Conference on Uncertainty in AI*, 514–522.

Shachter, R.; D'Ambrosio, B.; and del Favero, B. 1990. Symbolic Probabilistic Inference in Belief Networks. In *Proc. Conf. on Uncertainty in AI*, 126–131.

DISTRIBUTED STRUCTURE VERIFICATION IN MULTIPLY SECTIONED BAYESIAN NETWORKS

Y. Xiang
Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada S4S 0A2, yxiang@cs.uregina.ca

Abstract

Multiply sectioned Bayesian networks (MSBNs) provide a framework for probabilistic reasoning in a single user oriented system in a large problem domain or in a cooperative multi-agent distributed interpretation system. During the construction or dynamic formation of a MSBN, an automatic verification of the acyclicity of the overall structure is desired. Although algorithms for testing acyclicity are well known, they assume a centralized storage of the graphical structure to be tested. We discuss why a centralized representation of the overall structure is undesirable. We then propose a set of distributed operations that are performed by individual subnets/agents in the system to cooperatively verify the acyclicity of the overall structure.

1 INTRODUCTION

Multiply sectioned Bayesian networks (MSBNs) is an extension of Bayesian networks (BNs) [4, 3, 1]. A MSBN consists of a set of interrelated Bayesian subnets [13, 12]. Each subnet shares a non-empty set of variables with at least one other subnet. Subnets are organized into a hypertree structure such that probabilistic inference can be performed coherently in a modular and distributed fashion. The modularity improves inference efficiency in a single user oriented system in a large problem domain [11]. It also extends MSBNs into a framework for probabilistic reasoning in cooperative multi-agent distributed interpretation systems [7].

As the structure of a BN is a directed acyclic graph (DAG), the overall structure of a MSBN, the composition of subnet structures, is also a DAG. To ensure the correct composition, an automatic verification of the acyclicity of the composed structure is desired. Although algorithms for testing acyclicity are well known [6], they assume a centralized storage of the graph to be tested. In this paper, we propose a set of distributed operations used by individual subnets/agents to test cooperatively the acyclicity of the composed structure.

Section 2 briefly reviews the theory and applications of MSBNs. The concepts necessary for the rest of the paper are formally defined. Section 3 discusses reasons why a distributed verification of acyclicity is preferred over a centralized test. Section 4 shows that some 'obvious' solutions to the distributed verification do not solve the problem. Section 5 derives the graph-theoretic fundation for the proposed operations and Section 6 presents these operations for cooperative verification.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/295 ©1996 FLAIRS

2 OVERVIEW OF MSBNS

In this section, we briefly overview the theory of MSBNs and their applications. More details on MSBNs can be found in [13, 11, 7, 8].

A BN S is a triplet (N, D, P) where N is a set of variables, D is a directed acyclic graph (DAG) whose nodes are labeled by elements of N, and P is a joint probability distribution (jpd) over N. We shall call N the *domain* of S, D the *structure* of S and P the *distribution* or jpd of S.

Let $G^i = (N^i, E^i)$ $(i = 1, 2)$ be two graphs (directed or undirected). We shall refer to the graph $G = (N^1 \cup N^2, E^1 \cup E^2)$ as the *union* of G^1 and G^2, denoted $G = G^1 \sqcup G^2$.

A MSBN M is a collection of Bayesian subnets that together define a BN. These subnets are required to satisfy certain conditions that permit the construction of distributed inference algorithms. One of these conditions requires that nodes shared by different subnets form a *d-sepset*, as defined below.

Definition 1 (d-sepset) *Let $D^i = (N^i, E^i)$ $(i = 1, 2)$ be two DAGs such that $D = D^1 \sqcup D^2$ is a DAG. The intersection $I = N^1 \cap N^2$ is a* `d-sepset` *between D^1 and D^2 if for every $A_i \in I$ with its parents π_i in D, either $\pi_i \subseteq N^1$ or $\pi_i \subseteq N^2$. Each node in a d-sepset is called a* `d-sepnode`.

It can be shown that when a pair of subnets are isolated from M, their d-sepset renders them conditionally independent. Figure 1 (left) shows the three DAGs D^i $(i = 1, 2, 3)$ of a MSBN for diagnosis of three neuromuscular diseases, Median nerve lesion (Medn), Carpal tunnel syndrome (Cts) and Plexus upper trunk lesion (Plut).[1] Each d-sepnode is highlighted by a dotted circle. The d-sepset between each pair of DAGs is $\{Medn, Cts, Plut\}$. In general, d-sepsets between different pairs of DAGs of M may be different.

Just asthe structure of a BN is a DAG, the structure of a MSBN is a multiply sectioned DAG (MSDAG) of a hypertree structure, or simply a *hypertree MSDAG* defined as follows.

Definition 2 (Hypertree MSDAG) *A hypertree MSDAG $\mathcal{D} = \bigsqcup_i D^i$, where each D^i is a connected DAG, is a DAG that is built by the following procedure:*

Start with an empty graph (no node). Recursively add a DAG D^k, called a `hypernode`*, to the existing MSDAG $\bigsqcup_{i=1}^{k-1} D^i$ subject to the constraints:*

[d-sepset] For each D^j $(j < k)$, the intersection $I^{jk} = N^j \cap N^k$ is a d-sepset.

[Local covering] There exists D^i $(i < k)$ such that, for each D^j $(j < k; j \neq i)$, we have $I^{jk} \subseteq N^i$.

[1] The example is taken from a fraction of PAINULIM [11] with modification.

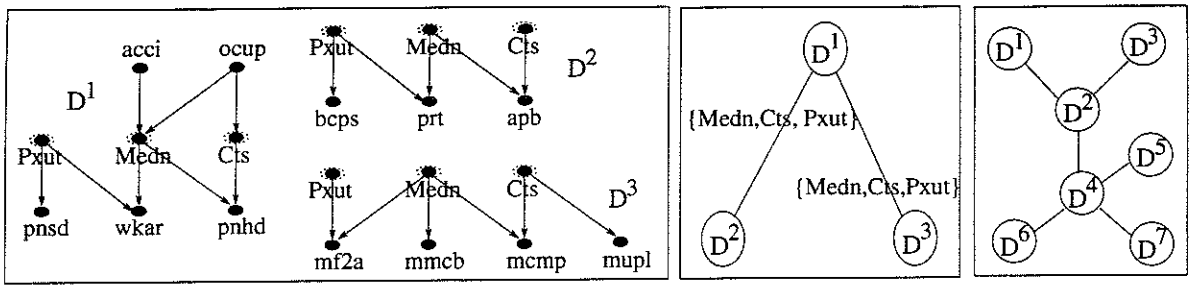


Figure 1: Left: The DAGs of an example MSBN for neural muscular diagnosis. Middle: The hypertree organization of the DAGs in the left. Right: A general hypertree MSDAG. Each d-sepnode is highlighted by a dotted circle.

I^{jk} is called the `hyperlink` *between hypernodes D^j and D^k, and D^i and D^k are said to be* `adjacent`.

The DAGs in Figure 1 (left) can be organized into the hypertree MSDAG in Figure 1 (middle). Figure 1 (right) depicts a general hypertree MSDAG. Although DAGs of subnets of a MSBN M are organized into a tree as defined above, each DAG may be multiply connected (more than one path exist between a pair of nodes), e.g., D^1. Moreover, there can be multiple paths between a pair of nodes in different DAGs in a hypertree MSDAG. For instance, multiple paths are formed between *apb* and *mcmp* after D^2 and D^3 are unioned. A hypertree structured M ensures that each hyperlink render the two parts of M that it connects conditionally independent. An intuitive justification of this structure is given in [10].

A MSBN is defines as follows. Readers are referred to [13] for more details.

Definition 3 *A MSBN M is a triplet $(\mathcal{N}, \mathcal{D}, \mathcal{P})$. $\mathcal{N} = \bigcup_i N^i$ is the* `total universe` *where each N^i is a set of variables. $\mathcal{D} = \bigsqcup_i D^i$ (a hypertree MSDAG) is the* `structure` *where nodes of each DAG D^i are labeled by elements of N^i. $\mathcal{P} = \prod_i P^i(N^i) / \prod_k P^k(I^k)$ is the* `joint probability distribution` *(jpd). Each $P^i(N^i)$ is a probability distribution over N^i such that whenever D^i and D^j are adjacent in $\mathcal{D}$, the marginalizations of $P^i(N^i)$ and $P^j(N^j)$ onto the d-sepset I^{ij} are identical. Each $P^k(I^k)$ is such a marginal distribution over a hyperlink of $\mathcal{D}$. Each triplet $S^i = (N^i, D^i, P^i)$ is called a* `subnet` *of M.*

Without confusion, we shall say that two subnets S^i and S^j are adjacent if D^i and D^j are adjacent. We shall refer to the computational entity that holds the representation of a single subnet as an *agent*.

A MSBN can be used as a framework for probabilistic reasoning in a single user oriented system in a large problem domain. Using a MSBN is most beneficial if subdomains of the problem domain are loosely coupled (the size of each d-sepset is reasonably small relative to the size of the subdomain) and evidence and queries are focused on one subdomain for a period of time before shifting to a different subdomain. For example in Figure 1 (right), the user may focuses attention on the subnet of the structure D^1, denoted by S^1. After several pieces of evidence are entered and queries are issued to this subnet, the user may shift attention to the subnet S^3. The inference operations of MSBNs will then propagate evidence from S^1 to S^2 and then to S^3. The user can then enter evidence on variables contained in S^3. It can be shown that with such a restricted belief propagation during attention shift, the answers to queries obtained in S^3 are always consistent to all evidence accumulated in the *entire* MSBN. Computational complexity, however, is reduced by not having to update any subnets not on the hyperpath from the current subnet to the next target subnet. Application domains of single-user MSBNs include diagnosis of natural systems [11] and model-based diagnosis of artificial systems [5].[2]

MSBNs can be extended into a framework for probabilistic reasoning in cooperative multi-agent distributed interpretation systems. Each agent holds its partial perspective of a large problem domain, accesses a local evidence source, consumes its own computation resource, communicates with other agents *infrequently*, and answers queries. It can be shown [9] that if all agents are cooperative (vs self-interested), and each pair of adjacent agents are conditionally independent given their shared variables and have common initial belief on the shared variables, then a joint system belief is well defined which is consistent with each agent's belief. Even though multiple agents may acquire evidence asynchronously in parallel (compare with the single user case where evidence is always entered into the current subnet), the corresponding communication operations of MSBNs ensure that the answers to queries from each agent are consistent with evidence acquired in the entire system after each communication. Since communication is infrequent, the operations also ensure that between two successive communications, the answers to queries for each agent are consistent with all local evidence gathered so far and are consistent with all evidence gathered in the entire system up to the last communication. Therefore, a MSBN can be characterized as one of functionally accurate, cooperative distributed systems [2]. Potential applications include decision support to cooperative human users in uncertain domains and troubleshooting a complex system by multiple knowledge based subsystems [9].

3 WHY DISTRIBUTED VERIFICATION?

As defined in Section 2, the structure of a MSBN is a MSDAG which is a DAG. Automatic verification of acyclicity of this structure is desirable in the construction of large MSBNs. Algorithms that test whether a directed graph is a DAG based on topological sorting are well known [6]. These algorithms, however, assume a central representation of the graphical structure to be tested.

A central representation of all DAGs in a MSBN is not desirable for at least two reasons. First, the construction of a multi-agent MSBN requires only the knowledge of the

[2] Although MSBNs are not referenced directly, the representation formalism used is a special case of MSBNs. For example, the set of input nodes I, output node O, mode node M, and dummy node D [5], which forms an interface between a higher level and a lower level in the hierarchy, is a d-sepset [13]. The 'composite joint tree' [5] corresponds to the 'hypertree' [13]. The way in which inference is performed in the composite join tree corresponds to the operation *ShiftAttention* [13].

interface (d-sepset) between subnets (BNs) and does not require the knowledge of the internal structure of each subnet. Therefore, each subnet may be developed by an independent vendor who may not be willing to disclose the structural details. The assumption of a central representation of all DAGs will rule out the possibility to cooperate agents built by such vendors.

Secondly, a MSBN can potentially be dynamic. That is, subnets may join or leave the MSBN as the system is functioning. It is desirable to verify the correctness of the structure of the system whenever the member subnets change. It is also desirable that the verification does not require the communication of all DAGs to a central location or does not depend upon a single agent to maintain a repository of all DAGs in the current system.

In this paper, we propose a distributed algorithm for verification of the acyclicity of a MSBN structure. During the verification process, each agent does not need to reveal its internal structure. We shall refer to the structure to be tested as a *DAG union* since it may not qualify to be a MSDAG.

4 ISSUES IN DISTRIBUTED VERIFICATION

Recall that a MSDAG is built subject to the d-sepset and local covering conditions. It should be noted that these two conditions do not rule out the possibility of a directed cycle in the resultant DAG union.

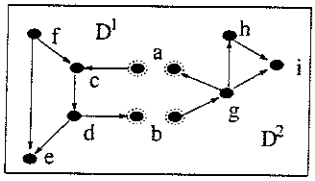


Figure 2: Two DAGs whose union is not a DAG.

Figure 2 shows two DAGs D^1 and D^2 with their d-sepset being $\{a, b\}$. If we union the two DAGs, it clearly satisfies the local covering condition. However, the union contains the directed cycle (a, c, d, b, g, a) and thus is not a DAG.

The above cycle can be detected if we union the pair of DAGs and test the acyclicity. Although the pairwise verification may detect some directed cycles, pairwise acyclicity in a DAG union does not guarantee the global acyclicity.

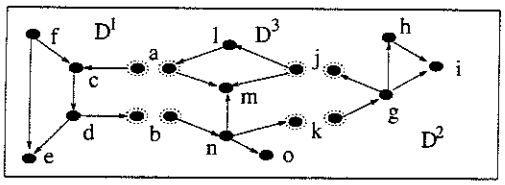


Figure 3: Three DAGs which are pairwise acyclic but whose union is cyclic.

Consider the three DAGs in Figure 3. The union of D^1 and D^3 is acyclic and so is the union of D^3 and D^2. However, when the three DAGs are unioned, a directed cycle $\{a, c, d, b, n, k, g, j, l, a\}$ is formed. Clearly, a distributed verification of acyclicity requires cooperation beyond pairs.

5 VERIFICATION BY MARKING NODES

In this section, we show that acyclicity of a DAG union can be verified by marking non-d-sepnodes and d-sepnodes separately and recursively. Once it is established, we can mark non-d-sepnodes locally and mark d-sepnodes by cooperation as presented in the next section. A node x is *marked* if x and arcs connected to x are ignored from further verification process. The following two propositions show that marking of non-d-sepnodes can be performed separately.

Proposition 4 *Let D be a DAG in a DAG union U. Let x be a root in D and be not in any d-sepset. Then the acyclicity of U remains the same after x is marked.*

Proof:

If U is acyclic, then marking x cannot create a directed cycle in U. Suppose U is cyclic. Then there exists a non-empty set O of cycles in U. Since x is a root in D, it does not have any incoming arc in D. Since x is not a d-sepnode, it cannot have any incoming arc from any other DAGs in U. Therefore, x cannot participate in any cycles in O, which implies that none of the cycles in O will be changed after x is marked. □

For example, the node f in Figure 3 is such a root node. Note that the condition that the node must not be in a d-sepset is necessary. For example, the node j is a root node in D^3, but it should not be marked since it is part of an inter-DAG cycle.

Proposition 5 *Let D be a DAG in a DAG union U. Let x be a leaf in D and be not in any d-sepset. Then the acyclicity of U remains the same after x is marked.*

The proof is similar to that of Proposition 4. For example, the node o in Figure 3 is such a leaf node. Again, the non-d-sepnode condition is necessary. For instance, the leaf node j in D^2 should not be marked.

Once a non-d-sepnode root or leaf is marked, other non-d-sepnodes may become roots or leaves. For example, after the leaf i is marked, the node h becomes a leaf and can also be marked. Hence marking of non-d-sepnode roots and leaves can be performed recursively.

The following two propositions show that marking of d-sepnodes can also be performed separately. Their proof are similar to that of Proposition 4.

Proposition 6 *Let x be a d-sepnode in a DAG union U. If x is a root in every DAG that it participates, Then the acyclicity of U remains the same after x is marked.*

Proposition 7 *Let x be a d-sepnode in a DAG union U. If x is a leaf in every DAG that it participates, Then the acyclicity of U remains the same after x is marked.*

Similar to non-d-sepnodes, d-sepnode roots and leaves can also be marked recursively.

Next, we show that if a DAG union is acyclic, every node in it will be marked by recursive application of Propositions 4, 5, 6 and 7.

Proposition 8 *Let a DAG union U be acyclic. For each node x in U, x can be marked after a finite rounds of recursive marking of non-d-sepnode roots and leaves and d-sepnode roots and leaves.*

Proof sketch:

Without lossing generality, we assume that at each round of marking, all current non-d-sepnode roots and leaves are marked first, followed by the marking of all d-sepnode roots and leaves.

If x is either a root or a leaf, then x can be marked in the first round. Suppose x is neither a root nor a leaf. Since U is acyclic, there exists a longest directed path p started at a root r_1 at one end and a leaf v_1 at the other end such that

x is in p. We denode p by $(r_1, r_2, \ldots, r_m, x, v_n, \ldots, v_2, v_1)$, where $m, n > 0$.

Since r_1 is a root, it can be marked either as a non-d-sepnode or as a d-sepnode in the first round. After the first round, r_2 must become a root. We show the opposite cannot be true. If r_2 does not become a root, it must be the case that r_2 has at least one other parent r not being marked in the first round. But this implies that r has at least one parent s, which in turn implies that there exists a directed path $t = (\ldots, s, r, r_2, \ldots, r_m, x, v_n, \ldots, v_2, v_1)$ that is longer than p. This contradicts the assumption that p is the longest path that contains x.

Using the same argument repeatedly, we conclude that x will be marked no later than the $m+1$ rounds. □

Finally, we show that if a DAG union is cyclic, some nodes will remain unmarked.

Proposition 9 *Let a DAG union U be cyclic. At least three nodes cannot be marked after recursive marking of non-d-sepnode roots and leaves and d-sepnode roots and leaves.*

Proof sketch:

Since U is cyclic, there exists at least one directed cycle o in U. Since U is a DAG union, o consists of at least three nodes.

Let x be a node in o. We denote o by $(x, y_1, \ldots, y_n, x)$ where $n > 1$, y_1 is the parent of x and y_n is the child of x in o. Without lossing generality, we assume that at each round of marking, all current non-d-sepnode roots and leaves are marked first, followed by the marking of all d-sepnode roots and leaves.

We claim that x can never be marked as a root. Suppose x is marked in a particular round m as a root, then y_1 must be marked in round $m-1$ or earlier. But this implies that y_2 must be marked in round $m-2$ or earlier. Repeating this argument, y_n must be marked in round $m-n$ or earlier, which implies that x must be marked in round $m-n-1$ or earlier. Now the marking of x becomes the precondition for x to be marked, which is impossible. Hence we conclude that x can never be marked as a root. Similarly, x can never be marked as a leaf either.

Since the argument is applicable to any node x in o, none of the nodes in o can be marked by recursive marking of of non-d-sepnode roots and leaves and d-sepnode roots and leaves. □

6 COOPERATIVE VERIFICATION

As demonstrated in Section 4, in order to verify the acyclicity of a DAG union, agents must cooperate. Since cooperation requires communication which incurs overhead, it is desirable to simplify the task for cooperation as much as possible. According to Propositions 4 and 5, non-d-sepnode roots and leaves in a DAG union can be marked separately and recursively. We define a preprocessing operation to mark these nodes.

Let a DAG in the DAG union be arbitrarily chosen. If we treat this DAG as the root of the hypertree and direct the hyperlinks of the hypertree away from it, then the hypertree is converted into a *directed* tree. For each given DAG, we can then refer to each adjacent DAG as its *child* or its *parent* in the normal sense.

Operation 10 (PreProcess) *When* `PreProcess` *is called in a DAG D, the following are performed:*

1. *D recursively marks each non-d-sepnode root or leaf.*
2. *D calls* `PreProcess` *in each child DAG.*

After `PreProcess` is completed in a DAG union, nodes left unmarked in each DAG are either isolated d-sepnodes, or nodes that form directed pathes ended with d-sepnodes. Cooperation among DAGs is needed to further the verification process. Figure 4 shows the three DAGs in Figure 3 after `PreProcess` is initiated in any of them. Marked nodes are shown as grey. Only directed pathes are left in this case.

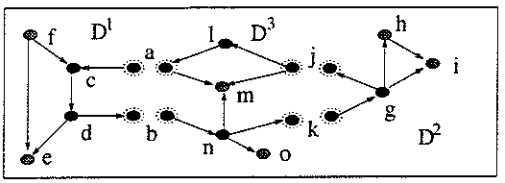


Figure 4: Three DAGs in Figure 3 after local preprocessing. Marked nodes are shown as grey.

Based on Propositions 6 and 7, the operation `CollectFamilyInfo` is used by a DAG to find out if a d-sepnode x can be marked. The operation passes a triple (x, p, c) around all child DAGs which contain x. The purpose is to record the parent/child information for x, where p is a count of the number of DAGs that contain parents of x and c is a count of the number of DAGs that contain children of x. The caller in the following definition refers to either a parent DAG or the next highter level of operation which initiated this operation.

Operation 11 (CollectFamilyInfo) *When* `CollectFamilyInfo(x)` *is called in a DAG D, the following are performed:*

1. *D forms a triple $t_0 = (x, p_0, c_0)$, where $p_0 = 1$ if D contains a parent of x and $p_0 = 0$ otherwise, and $c_0 = 1$ if D contains a child of x and $c_0 = 0$ otherwise.*
2. *If D has no child DAG to which x is a d-sepnode, then D returns t_0 to* `caller`.
3. *Otherwise, D calls* `CollectFamilyInfo(x)` *in each child DAG to which x is a d-sepnode.*
4. *After each child DAG being called has returned their triples (assuming k DAGs are called), $t_1, t_2, \ldots, t_k$, D returns a triple $t = (x, \sum_{i=0}^{k} p_i, \sum_{i=0}^{k} c_i)$ to* `caller`.

Once a d-sepnode is determined to be a root or a leaf, the operation `DistributeMark` is used to mark it in every DAG that it participates.

Operation 12 (DistributeMark) *When* `DistributeMark(x)` *is called in a DAG D, the following are performed:*

1. *D marks the node x.*
2. *D recursively marks any non-d-sepnode root or leaf.*
3. *If D has any adjacent DAG to which x is a d-sepnode except* `caller`, *then D calls* `DistributeMark(x)` *in each of them.*

The operation `MarkNode` combines `CollectFamilyInfo` and `DistributeMark` to recursively check and mark all d-sepnodes that are markable down the hypertree.

Operation 13 (MarkNode) *When* `MarkNode` *is called in a DAG D, the following are performed:*

1. *D returns* `false` *if it has no child DAG, otherwise continues.*

2. *For each unmarked d-sepnode x with a child DAG of D, D calls* `CollectFamilyInfo(x)` *in itself. When the trible (x, p, c) is returned to D, D calls* `DistributeMark(x)` *in itself if $p = 0$ or $c = 0$.*

3. *D calls* `MarkNode` *in each child DAG.*

4. *If any child DAG returns* `true` *or* `DistributeMark(x)` *was called in D, then D returns* `true` *to* `caller`. *Otherwise, D returns* `false` *(no node is marked).*

The operation `MarkedAll` checks if all nodes in a DAG union have been marked after roots and leaves have been recursively marked. The DAG union is acyclic if it returns `true`.

Operation 14 (MarkedAll) *When* `MarkedAll` *is called in a DAG D, the following are performed:*

1. *If there exists a node in D that has not been marked, then D returns* `false`.

2. *Otherwise, if D has no child DAG, it returns* `true`. *If D has children DAGs, D calls* `MarkedAll` *in each child DAG.*

3. *If any child DAG returns* `false` *(with unmarked nodes), then D returns* `false`. *Otherwise, D returns* `true`.

Finally, `TestAcyclicity` combines three previously defined operations to provide the top level operation for the verification of acyclicity of a DAG union.

Operation 15 (TestAcyclicity) *When* `TestAcyclicity` *is initiated in a DAG union, the following are performed:*

1. *A DAG D is arbitrarily chosen as the root of the hypertree.*

2. *D calls* `PreProcess` *in itself.*

3. *D calls* `MarkNode` *in itself repeatedly until* `false` *is returned (no node is marked in the last call).*

4. *D calls* `MarkedAll` *in itself. If* `true` *is returned, then* `TestAcyclicity` *is terminated with* `acyclic` *returned (the union is acyclic). Otherwise, the operation is terminated with* `cyclic` *returned.*

The following theorem establishes the correctness of the above operations.

Theorem 16 *The operation* `TestAcyclicity` *determines correctly if a DAG union U is acyclic or not.*

Proof sketch:

According to Propositions 4 and 5, non-d-sepnode roots and leaves can be marked separately and recursively. `PreProcess` does the first round of such marking and `DistributeMark` (step 2) performs the recursive marking.

According to Propositions 6 and 7, d-sepnode roots and leaves can be marked separately and recursively. `MarkNode` identifies these nodes by `CollectFamilyInfo` (either $p = 0$ or $c = 0$) and then mark them by `DistributeMark` (steps 1 and 3).

According to Proposition 8, if U is acyclic, all nodes can be marked by recursive marking of non-d-sepnode roots and leaves and d-sepnode roots and leaves. Each call of `MarkNode` in `TestAcyclicity` marks at least one root or one leaf, until all nodes are marked at which time `false` is returned. `MarkedAll` in step 4 will identify that all nodes have been marked and return `true` to signify that U is acyclic.

According to Proposition 9, if U is acyclic, at least three nodes are unmarked when `MarkNode` returns `false`. `MarkedAll` will then identify this and return `false` to signify that U is cyclic. □

7 REMARKS

In this paper, we presented a distributed algorithm, in terms of a set of distributed operations, for verification of acyclicity of the overall structure of a MSBN. The algorithm does not require each agent in the system to reveal its internal structure. It only provides information as whether a d-sepnode has a parent or a child in the DAG that the agent is responsible for. Therefore, the algorithm supports the construction of MSBNs constructed from multiple computational agents built by multiple vendors while providing the automatic verification of the correctness of the overall structure.

Acknowledgements

This work is supported by the Research Grant OGP0155425 from NSERC. Helpful comments from anonymous reviewers are acknowledged.

References

[1] E. Charniak. Bayesian networks without tears. *AI Magazine*, 12(4):50–63, 1991.

[2] V.R. Lesser and D.D. Corkill. Functionally accurate, cooperative distributed systems. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-11(1):81–96, 1981.

[3] R.E. Neapolitan. *Probabilistic Reasoning in Expert Systems.* John Wiley and Sons, 1990.

[4] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[5] S. Srinivas. A probabilistic approach to hierarchical model-based diagnosis. In *Proc. Tenth Conf. Uncertainty in Artificial Intelligence*, pages 538–545, Seattle, Washington, 1994.

[6] D.F. Stubbs and N.W. Webre. *Data Structures with Abstract Data Types and Modula-2.* Brooks/Cole, 1987.

[7] Y. Xiang. Distributed multi-agent probabilistic reasoning with Bayesian networks. In Z.W. Ras and M. Zemankova, editors, *Methodologies for Intelligent Systems*, pages 285–294. Springer-Verlag, 1994.

[8] Y. Xiang. Optimization of inter-subnet belief updating in multiply sectioned Bayesian networks. In *Proc. Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 565–573, Montreal, Quebec, 1995.

[9] Y. Xiang. A probabilistic framework for cooperative multi-agent distributed interpretation and optimization of communication. *To appear in Artificial Intelligence*, fall, 1996.

[10] Y. Xiang. Semantics of multiply sectioned Bayesian networks for cooperative multi-agent distributed interpretation. In *To appear in Proc. Canadian Artificial Intelligence Conference*, May, 1996.

[11] Y. Xiang, B. Pant, A. Eisen, M. P. Beddoes, and D. Poole. Multiply sectioned Bayesian networks for neuromuscular diagnosis. *Artificial Intelligence in Medicine*, 5:293–314, 1993.

[12] Y. Xiang, D. Poole, and M. P. Beddoes. Exploring locality in Bayesian networks for large expert systems. In *Proc. Eighth Conference on Uncertainty in Artificial Intelligence*, pages 344–351, Stanford, CA, 1992.

[13] Y. Xiang, D. Poole, and M. P. Beddoes. Multiply sectioned Bayesian networks and junction forests for large knowledge based systems. *Computational Intelligence*, 9(2):171–220, 1993.

INDUCTIVE REASONING WITH CONDITIONAL PROBABILITIES

Paul Snow
P.O. Box 6134
Concord, NH 03303-6134 USA
paulsnow@delphi.com

Abstract

Three formalisms which use the conditional probabilities of evidence are contrasted in their suitability for selected inductive reasoning tasks. A new calculus, called *ignorant induction*, combines some properties of probability and some of possibility to produce coherent partial credal orderings which avoid Kyburg's Lottery Paradox and which resolve Goodman's Grue Puzzle in an intuitively satisfactory way.

1 INTRODUCTION

Throughout this paper, inductive reasoning will consist of drawing sensible inferences about the relative belief-worthiness of sentences, based solely upon the conditional probabilities of evidence. Three formalisms for such reasoning will be contrasted: probability, a modified version of possibility, and a new calculus abstracted from qualitative probability and the properties of conditional probability, which will be called "ignorant induction." Comparisons among the formalisms will be based on intuitive principles developed for commonsense nonmonotonic reasoning, a philosopher's puzzle problem, and coherence in hypothetical gambling situations.

2 IGNORANT INDUCTION

Ignorant induction is "ignorant" because it seeks to apply Bayesian-style inference methods to situations where prior probabilities are unknown, and none are assumed. An intuitive axiomatic derivation of ignorant induction has appeared in Snow (1994). The formalism has been successfully applied to problems in statistics, such as paradox-free Bayesian significance testing of sharp null hypotheses, and the construction of interval parameter estimates in "perfect working" reliability engineering tests (Snow, 1996).

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/300 ©1996 FLAIRS

Ignorant induction blends properties of probability and possibility. When applied to mutually exclusive disjunctions, it decides orderings according to a "maximum rule" like that of possibility: if disjunctions S and T are exclusive, and s is the best-supported disjunct in S, and t is the best-supported disjunct in T, then S is preferred to T if and only if s is preferred to t.

When comparing intersecting sentences, however, ignorant induction ignores their intersection, a property of probability called *quasi-additivity*, and so S is preferred to T if and only if (S **and** ¬T) is preferred to (T **and** ¬S).

3 NOTATION AND CONVENTIONS

3.1 The Domain and Inference Rules

Definition. A *partitioned domain* is a set whose elements are: (i) the always-true sentence, denoted **true**; (ii) the always false sentence, denoted **false**; (iii) two or more mutually exclusive sentences, called *atoms*; (iv) well-formed expressions involving atoms, **or**, and parentheses, called *simple disjunctions*; and (v) well-formed expressions involving simple disjunctions, **true**, **false**, **or**, **not**, and parentheses.

We shall assume that the evidence bears on the atoms in some partitioned domain. Our knowledge base consists of conditional probability values for that evidence given individual atoms. Although partitioned domains are more structured than some other universes of discourse, they are common in statistical work, where much applied induction takes place, and in the theoretical literature of inductive reasoning.

The notation S \e\ T will denote "sentence S is at least as well supported by evidence e as sentence T" in a sense appropriate to each calculus:

Probability: p(S | e) >= p(T | e); or

$$\Sigma_{\text{atom s in S}}\, p(s)*p(\, e \mid s\,) >= \Sigma_{\text{atom t in T}}\, p(t)*p(\, e \mid t\,)$$

In the case of uniform priors, this is equivalent to:

$$\Sigma_{\text{atom s in S}}\, p(\, e \mid s\,) >= \Sigma_{\text{atom t in T}}\, p(\, e \mid t\,)$$

Possibility:

$$\max_{\text{atom s in S}} p(e \mid s) >= \max_{\text{atom t in T}} p(e \mid t)$$

except that if S = ¬T, then the inequality must be strict.

Ignorant induction: S \e\ T if and only if there is no atom t peculiar to T (i.e., t is in T **and** ¬S) such that p(e | t) > 0, or there is some atom s peculiar to S such that p(e | s) > p(e | t) for every atom t peculiar to T.

Examples. Suppose the partitioned domain contains just three atoms, *s*, *t*, and *u*, and there is evidence *e* such that p(e | s) = .5, p(e | t) = .5, p(e | u) = .2.

Probability: Under uniform priors, (t **or** u) \e\ s, since .7 is bigger than .5; if instead of uniform priors, we have p(s) = .8, p(t) = .1, p(u) = .1, then the opposite sense ordering holds, since .4 is bigger than .07.

Possibility: (t **or** u) is *not* ordered ahead of s, since (t **or** u) = ¬s, and the best supported atoms in each sentence are tied at .5; however, t \e\ s, and s \e\ t, despite the tie; the sentences are not complementary.

Ignorant induction: (t **or** u) is *not* ordered ahead of s, since the best supported atoms peculiar to each sentence are tied at .5; nor are s and t ordered with respect to each other, for the same reason; (s **or** t) \e\ (u **or** t), since .5 (for s) is bigger than .2 (for u), and the shared atom t does not affect the result.

3.2 Combinations of Evidence

Sometimes, we shall speak of conjunctive evidence, such as S \e **and** e'\ T. Such evidence is treated as a single piece of evidence, under the usual combination rules for conditional probabilities.

Conjunctive evidence probabilities can take on any value in the range

$$\min(p(e \mid s), p(e' \mid s)) >= p(e \textbf{ and } e' \mid s) >= 0$$

A special case is *conditional independence*, where *e* tells us nothing about *e'* once *s* is assumed, as perhaps the results of two distinct experiments might be related, and then p(e **and** e' | s) = p(e | s) * p(e' | s).

In the case where one of the "evidence" conjuncts is the confirmation that some sentence in the domain is true, as in S \e **and** U\ T, indicating that sentence *U* is confirmed as true, these rules provide that

p(e **and** U | u) = p(e | u) for all atoms u in U;
p(e **and** U | v) = 0 for all atoms v not in U

assuming that the "confirmation of U" means that p(U | u) = 1 and p(U | v) = 0. That the confirmation of a sentence so defined is conditionally independent of all other evidence is an easily proved theorem.

Examples. Suppose again that the partitioned domain contains just three atoms, *s*, *t*, and *u*, with evidence *e* such that p(e | s) = .5, p(e | t) = .5, and p(e | u) = .2, and there is some second piece of evidence *e'* such that p(e' | s) = .3, p(e' | t) = .4, and p(e' | u) = .5. If *e* and *e'* are conditionally independent,

p(e **and** e' | s) = .15, p(e **and** e' | t) = .20, and
p(e **and** e' | u) = .10

If (s **or** u) is confirmed, that is, t is eliminated, then

p(e **and** (s **or** u) | s) = .5, p(e **and** (s **or** u) | t) = 0,
and p(e **and** (s **or** u) | u) = .2

and so s \e **and** (s **or** u)\ u would be asserted by probability under uniform priors, by the possibility rule, and by ignorant induction as well.

3.3 Some Variants of the Possibility Formalism

The present version of the possibility calculus uses possibility values which are proportional to conditional probabilities of evidence. Other approaches exist. Spohn's (1990) kappa-calculus provides a non-additive truth-compositional induction scheme based on evidence conditionals. See Dubois and Prade (1992) for a comparison of kappa-calculus with possibility. It is important to distinguish the possibility calculus discussed here, based on conditional *probabilities*, from a method using conditional *possibilities*, introduced by Dubois and Prade (1991), although they share many properties.

Note also that the possibility calculus usually decides "ties" without treating complements as an exception. The merits of the exception will arise in the next section.

4 INDUCTIVE AND NONMONOTONIC REASONING

Researchers on nonmonotonic reasoning in AI have developed their own principles of intuition. This section concerns an incomplete list of such principles from Kraus, Lehman, and Magidor (1990) and Lehman and Magidor (1992), which apply to the special case where some sentence in a domain might be confirmed as true.

The translations of their principles into the present paper's notation interpret the preferential entailment connective |~ as the ordering of a sentence ahead of its negation. Thus, K |~ S becomes S \e **and** K\ ¬S, where *e* is some evidence bearing on the atoms of the domain, and *K* is the confirmation of some sentence in the domain.

[1] Right Weakening: If S => T and S \e **and** K\ ¬S, then T \e **and** K\ ¬T

[2] Reflexivity: If S is a non-empty sentence, then S \e **and** S\ ¬S

[3] Left Logical Equivalence: If K <=> K', then S \e **and** K\ ¬S implies S \e **and** K'\ ¬S

[4] Cut: If S \e **and** K\ ¬S and T \e **and** K **and** S\ ¬T, then T \e **and** K\ ¬T.

[5] Cautious Monotonicity: If S \e **and** K\ ¬S and T \e **and** K\ ¬T, then T \e **and** K **and** S\ ¬T

[6] Rational Monotonicity (based on a reformulation by Bacchus *et al.*, 1993): If not (¬S \e **and** K\ S) and if T \e **and** K\ ¬T , then T \e **and** K **and** S\ ¬T

The first three principles are all easily confirmed properties of conditional probability and of bounded, transitive orderings, and so probability, possibility, and ignorant induction all share them.

The fourth through sixth properties are shared properties of possibility (the proofs are similar to those of Dubois and Prade, 1991 for their "conditional object") and of ignorant induction (to be shown presently). As discussed by Pearl (1988), Cut and Cautious Monotonicity are *not* properties of probability distributions. Also, since probability is a complete ordering, if ¬S \e **and** K\ S does not hold, then S \e **and** K\ ¬S does hold. So, Rational Monotonicity is identical to Cautious Monotonicity for probability, and hence is not a property of that calculus.

The proof that Cautious Monotonicity holds for ignorant induction is typical of all three properties' proofs. Cautious Monotonicity also provides an opportunity to note some points about the possibility and probability formalisms.

[5] Cautious Monotonicity If S \e **and** K\ ¬S and T \e **and** K\ ¬T, then T \e **and** K **and** S\ ¬T

Proof for ignorant induction: The best supported atoms are all in KS, and they are also all in KT, so if we project onto KS, they are still in KT.

Counterexample for probability: If p(S | Ke) > 1/2, and p(T | Ke) > 1/2, and if Cautious Monotonicity holds, then p(T | SKe) > 1/2. Let p(S **and** ¬T | Ke) = .45, p(S **and** T | Ke) = .15, p(T **and** ¬S | Ke) = .40. So, p(S | Ke) = .6, p(T | Ke) = .55, but p(T | SKe) = p(S **and** T | Ke) / p(S | Ke) = .15/.6 = 1/4 < 1/2.

Note for possibility theory: The modified max rule does not permit S \e **and** K\ ¬S to be asserted in the event of a "tie"(π(s) = π(s'), where s and s' are the best supported atoms in KS and K¬S, respectively), even though ties are ordinarily dispositive in possibility theory. To see that a max rule exception is necessary for Cautious Monotonicity to hold generally, suppose s were in KS and not in KT, and t were in KT and not in KS, π(s) = π(t), and all other atoms had lower possibility values than s or t. The confirmation of S would make p(e **and** K **and** S | t) become zero, and the best supported atom in K **and** T **and** S would have lower possibility than s, which is the best supported atom in K **and** ¬T **and** S. The exception could just as well have been chosen to apply to mutually exclusive sentences generally, and not just complements, but the narrower exception was chosen for this paper.

Some perspective on the failure of the probability calculus to exhibit the Cut and Cautious Monotonicity properties is needed. Kraus, Lehman, and Magidor show that Cut and Cautious Monotonicity together entail another property:

And: If S \e **and** K\ ¬S and T \e **and** K\ ¬T, then (S and T) \e **and** K\ ¬(S and T)

Probability does not possess this property. However, possession of the And property can lead to Kyburg's (1961) notorious Lottery Paradox (Bacchus *et al.*, 1993). Any particular ticket in a lottery where exactly one ticket wins is unlikely to be the winner. The conjunctive conclusion for all tickets would be that *no* ticket is the winner, a conclusion endorsed by And, yet *some* ticket does get drawn.

The lottery is an occasion for reasoning about proportions in a population, a task for which probability is normatively suited. As the Lottery Paradox shows, a sound reasoner about proportions simply cannot display the And property. So at least one of Cut or Cautious Monotonicity cannot hold.

Possibility and ignorant induction avoid the paradox by never ordering each ticket behind all the others in the first place. The evidence conditionals cannot equally disfavor every ticket simultaneously: if the conditionals are all the same, then the evidence is irrelevant. So, the antecedent in the And rule doesn't get satisfied in the Lottery Paradox.

5 THE GRUE PROBLEM

A classic puzzle of inductive reasoning was suggested by the philosopher Nelson Goodman (1955). Suppose we define a silly color called "grue" to mean "currently colored green, but colored blue beginning on February 2, 2222." We then examine several emeralds, and they are all currently green. Our experiment provides obvious support for the proposition: "Emeralds are typically green." Goodman suggests that we ought to avoid a method which would also entertain the silly, but "confirmed," conclusion "Emeralds are typically grue."

To begin the analysis, we create a partitioned domain with atomic elements

green **and** grue, green **and** ¬grue, ¬green

and consider that our search through the emeralds provides evidence *e* such that:

$$p(\, e \mid \text{green } \mathbf{and} \text{ grue}\,) = 1,\ p(\, e \mid \text{green } \mathbf{and} \neg\text{grue}\,) = 1,\ p(\, e \mid \neg\text{green}\,) = a < 1$$

For probability, we assume equal priors over the atoms. The propositions of interest are green, grue, ¬green, and ¬grue.

The calculi are unanimous that each of the propositions of interest is ordered (weakly) ahead of ¬green, and that green comes ahead of any of them. This includes green \e\ grue, which makes sense, since grue implies (currently) green. The calculi are also unanimous that except for ¬green, no proposition of interest comes behind ¬green. No calculus asserts that grue \e\ ¬grue, and so all avoid the silly "confirmed" conclusion.

Probability goes beyond that, asserting ¬grue \e\ grue. That conclusion may be troubling, however, since the evidence tells us *nothing* about the prospects of color change. While the conclusion is fine for the humorous grue, it would be less obviously desirable had grue been defined as "clear and green." We might want our inference scheme to alert us about the irrelevance of the evidence to the clarity question, rather than to decide it for us. Note that *un*equal priors (biased against grue) would not affect this conclusion.

With possibility, no ordering is asserted between grue and ¬grue (since they are complements and they tie). Possibility does assert ¬grue \e\ green, along with grue \e\ green. These inferences, along with the consensus orderings, can be interpreted as "the evidence is equally good for constant and variable color," which is a reasonable response to Goodman's concerns.

Ignorant induction makes no assertions about the ordering of the propositions of interest beyond those on which all three calculi agree. We are thus explicitly told that the evidence speaks about current color, but is silent on color change.

6 COHERENCE IN INDUCTIVE REASONING

William Kingdom Clifford observed in the last century that "scientific thought is the guide of action; that the truth at which it arrives is not that which we can ideally contemplate without error, but that which we may act upon without fear" (cited in Bronowski, 1978). Inference suited for action is the hallmark of the Bayesian paradigm (Cheeseman, 1988), and is the rationale for accounts of induction based on "fair bets" (Kemeny, 1955), even though the bets being modeled are typically unrealizable. How could a "bet" that Einstein's physics describe the universe better than Newton's ever be settled?

Hypothetical betting may be best viewed as a "thought experiment" for checking whether one's beliefs support Clifford's fearless action. If the beliefs would fail as a guide to action in simple clear choices, then maybe they are not reliable in the murky real world.

The basic criterion for fair betting is *coherence*. In its simplest formulation, coherence says that no pattern of orderings will be asserted unless there is some probability distribution which displays the same pattern. If this constraint is violated, then one's preferences typically lead to the "Dutch Book" - a situation where gambling according to one's beliefs would result in a sure loss. Other statements of the constraint exist (e.g., Coletti *et al.*, 1993), but the notion of conforming to some probability distribution is typical.

Example In a partitioned domain with at least four atoms *p*, *q*, *r*, and *s*, none of which is certainly false, if we assert the orderings

$$p \text{ \e\ } (q \textbf{ or } s) \text{ and } (r \textbf{ or } s) \text{ \e\ } (p \textbf{ or } q)$$

then coherence forbids the ordering q \e\ r. It can be verified by simple addition that in any probability distribution, $prob(r)$ is at least twice $prob(q)$.

Objections are raised against Dutch Books and coherence on the grounds of "forced choice." Some versions of the Dutch Book argument require readiness to place bets on *any* sentences in the domain. One is not allowed to abstain from a bet on the grounds of having no opinion about its merits. Clearly, the response to such a demand is of questionable relevance to one's actual beliefs (Zadeh and Shafer comments accompanying Lindley, 1982).

There is no element of forced choice in the present notion of coherence. All that is considered is whether any orderings which are chosen would lead to a loss for sure if relied upon in the thought experiment.

Attempts have also been made to extend Dutch Book arguments to "conditional bets," in order to motivate Bayes' theorem as a paradigm of belief change. Such "dynamic" coherence arguments are controversial even within the probabilist community (Bacchus *et al.*, 1990; Howson, 1993). Our concern is solely with "static" coherence: whether the present beliefs are bet-worthy, regardless of how the beliefs were adopted.

Of course, the probability calculus is the ultimate in coherence: its ordering by belief-worthiness *is* a probability distribution. Probability calculi based on set-valued estimates under a "unanimous agreement" rule (an ordering is asserted if and only if the ordering holds in every probability distribution in the estimate set) are also clearly coherent.

Ignorant induction orderings always conform to some probability distribution. An algorithm for computing probability distributions consistent with any ignorant induction ordering appears in Snow (1994). So the ignorant induction calculus, too, is always coherent.

The possibility calculus, however, is not always coherent. Suppose there are least five not-certainly-false

atoms in the domain, and that four of them, p, q, r, and s are such that p and s are ordered equally with each other and ahead of both q and r, and that q is ordered ahead of r. It is easy to verify that p \e\ (q **or** s) and (r **or** s) \e\ (p **or** q), which along with q \e\ r constitutes the earlier example of an incoherent ordering pattern. This particular problem could be avoided by making the max rule exception introduced for nonmonotonic reasoning apply to all mutually exclusive sentences, not just complements.

Another pattern of possibility incoherence happens, however, with any two not-certainly-false atoms s, t which share equal possibility, in which case (if there are at least three atoms in the domain) intersecting sentences related by implication tie:

s \e\ (s **or** t) and (s **or** t) \e\ s

which can only happen in a probability distribution when t has zero probability.

7 CONCLUSIONS

Probability, possibility, and ignorant induction have been examined from three perspectives relevant to inductive reasoning. The additive probability calculus conforms to fewer intuitive constraints selected from the nonmonotonic reasoning literature than the non-additive possibility or ignorant induction calculi. On the other hand, by doing so, probability provides the right answers in the Lottery Paradox without abstaining. This contrast reflects the difference between a method suited to reasoning about proportions in a population as opposed to methods suited exclusively to interpreting evidence without the benefit of a population.

All three formalisms make some headway in avoiding the conundrum of the Grue Puzzle. Each reaches its own pattern of conclusions. The differences concern how to handle a question about which the evidence is uninformative, and suggest a useful role for partially ordered formalisms in inductive reasoning.

The possibility calculus does not enjoy static coherence, unlike the other two. Given the importance of coherence in the received theory of inductive reasoning, this may impede the application of possibility as an induction tool.

Overall, ignorant induction steers a middle course between possibility and probability, recommending it as an intuitively appealing vehicle for evidentiary inference.

References

Bacchus, F., H.E. Kyburg, Jr., and M. Thalos, Against conditionalization, *Synthese 85*, 475-506, 1990.

Bacchus, F., A.J. Grove, J.Y. Halpern and D. Koller, Generating degrees of belief from statistical information (technical report), preliminary version: Statistical foundations for default reasoning, *Proceedings Thirteenth IJCAI*, 906-911, 1993.

Bronowski, J., *Magic, Science, and Civilization*, New York: Columbia University, 1978.

Cheeseman, P., An inquiry into computer understanding, *Computational Intelligence 4*, 58-66 and 129-142, 1988.

Coletti, G., A Gilio, and R. Scozzafava, Comparative probability for conditional events: a new look through coherence, *Theory and Decision 35*, 237-258, 1993.

Dubois, D. and H. Prade, Possibilistic logic, preferential models, non-monotonicity and related issues, *Proceedings Twelfth IJCAI Conference*, 419-424, 1991.

Dubois, D. and H. Prade, Belief change and possibility theory, in P. Gardenfors (ed.), *Belief Revision*, Cambridge, UK: Cambridge University, 142-182, 1992.

Goodman, N., *Fact, Fiction, and Forecast*, Cambridge MA: Harvard, 1955.

Howson, C. Dutch book arguments and consistency, in D. Hull, M. Forbes, and K. Okruhlik (eds.), *PSA 1992*, East Lansing, MI: Philosophy of Science Association, 161-168, 1993.

Kemeny, J.G., Fair bets and inductive probabilities, *Journal of Symbolic Logic 20*, 263-273, 1955.

Kraus, S., D. Lehman, and M. Magidor, Nonmonotonic reasoning, preferential models and cumulative logics, *Artificial Intelligence 44*, 167-207, 1990.

Kyburg, H. E., Jr., *Probability and the Logic of Rational Belief*, Middletown: Wesleyan University Press, 1961.

Lehman, D. and M. Magidor, What does a conditional knowledge base entail?, *Artificial Intelligence 55*, 1-60, 1992.

Lindley, D.V., Scoring rules and the inevitability of probability, *International Statistical Review 50*, 1-26 (with commentaries), 1982.

Pearl, J., *Probabilistic Reasoning in Intelligent Systems*, San Mateo, CA: Morgan Kaufman, chapter 10, 1988.

Snow, P., The emergence of ordered belief from initial ignorance, *Proceedings of the AAAI Conference*, 281-286, Cambridge, MA: MIT Press, 1994.

Snow, P., Inference using conditional probabilities despite prior ignorance, *IEEE Trans. Systems, Man & Cybernetics*, (forthcoming), 1996.

Spohn, W., A general non-probabilistic theory of inductive reasoning, in R.D. Schacter, T.S. Levitt, L.N. Kanal, and J.F. Lemmer (eds.) *Uncertainty in Artificial Intelligence 4*, Amsterdam: North-Holland, 149-158, 1990.

Qualitative Decision Analysis: Evaluating an Order of Magnitude Calculus (some initial results)*

Paul O'Rorke
(paulo@ai.uwf.edu)
The Cognition Institute
University of West Florida
Pensacola, FL 32514-5750

Max Henrion
(henrion@lumina.com)
The Institute for
Decision Systems Research
Los Altos, CA 94022

Abstract

Decision theory is becoming increasingly significant in AI work on important tasks such as diagnosis and planning and serves as the basis for a new generation of expert systems known as normative expert systems. However, decision analytic approaches to AI problems have some weaknesses. Qualitative decision analysis (QDA) addresses these problems by extending qualitative reasoning methods developed for reasoning about physical systems and for probabilistic reasoning by incorporating information about utilities and values relevant to decisions. Like other qualitative reasoning techniques, QDA is driven by the notion that precise numbers and exact numerical computations are often unnecessary and can be replaced by qualitative approximations. This paper describes methods for evaluating QDA techniques and presents results of initial evaluations of an order of magnitude scheme.

1 Introduction

Decision theory is becoming increasingly significant in Artificial Intelligence. It is being used to address important tasks such as diagnosis (Heckerman, Horvitz, & Nathwani, 1992) and planning (Hanks, 1994), and serves as the basis for a new generation of expert systems known as normative expert systems. In spite of this, decision analytic approaches to AI problems have several major weaknesses. One weakness is that it is often difficult to acquire the numbers (probabilities and utilities) required for traditional decision analyses. Another weakness is that even when these numbers are readily available, the computations needed to assess likelihoods and decide on actions can become infeasible when there are too many numbers that must be taken into account. A final weakness of traditional approaches is that it can be difficult to comprehend the results of quantitative decision-theoretic inferences.

Qualitative decision analysis (QDA) aims to solve these problems. QDA is related to Bayesian decision theory just as qualitative physics is related to traditional physics. QDA is based on decision theory and attempts to be compatible with it to the extent possible. Like other qualitative reasoning techniques, QDA is driven by the notion that precise numbers and exact numerical computations are often unnecessary and can be replaced by symbolic approaches that provide a number of advantages. The main goals of QDA are to support decision making: 1) to support knowledge acquisition, 2) to support efficient inference procedures, 3) to support comprehensible explanation, and 4) to maximize compatibility with quantitative decision theory.

Several approaches that can be considered to be examples of QDA have been proposed, including *linguistic phrases* (Henrion & Druzdzel, 1990), *qualitative probabilistic networks (QPNs)* (Wellman, 1990), *order of magnitude calculi* (Pearl, 1993; Wilson, 1995) and *qualitative logics of decision* (O'Rorke, El Fattah, & Elliott, 1993).

Knowledge acquisition can be facilitated by qualitative approaches by taking advantage of easily acquired qualitative information and by not requiring exact values of numbers such as likelihoods and utilities. For example, linguistic phrases can be used to elicit information from users and to explain conclusions and recommendations without having to resort to numbers (Henrion & Druzdzel, 1990).

Inference algorithms that are more efficient than traditional quantitative approaches to Bayesian inference can result from QDA methods and representations. For example, (Druzdzel & Henrion, 1993) provides an efficient algorithm for performing inference on QPNs.

Explanations that suppress irrelevant details of numerical calculations can be provided by QDA methods. For example, qualitative logics of decision have been used to provide comprehensible explanations of recommended decisions (O'Rorke, et al., 1993). This is important because normative expert systems are unlikely to be accepted and their recommendations are unlikely to be followed unless we can develop automatic expla-

*This work was supported in part by NSF grant IRI-9120330 to the Institute for Decision Systems Research. The authors gratefully acknowledge discussions via electronic mail with Dr. Nic Wilson and his comments and the comments of anonymous reviewers on an earlier draft. The first author appreciates the support of Prof. Ken Ford, Director of the Cognition Institute. *Mathematica* (Wolfram Research, 1993) was used in computational experiments and for visualization.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/305 ©1996 FLAIRS

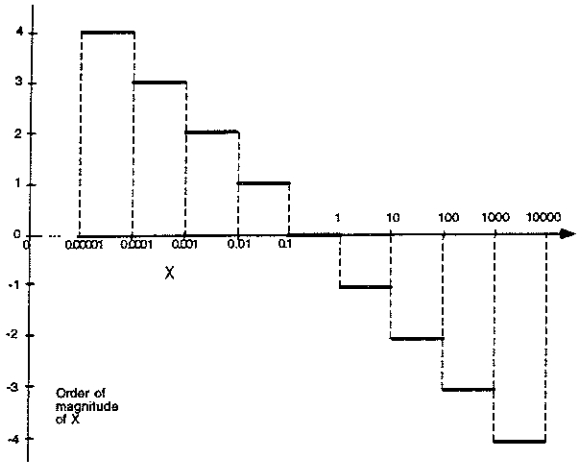


Figure 1: Conversion of real numbers into orders of magnitudes of a real valued ϵ between zero and one (in this case, $\epsilon = .1$). Adapted from (Henrion, Provan, Del Favero, & Sanders, 1994).

nation techniques that can provide comprehensible and convincing explanations for human users.

The common insight underlying all of these approaches is that the numbers required by traditional quantitative decision analysis are often not really needed to solve the decision problem at hand. In such cases, QDA approaches aim to solve the problem using qualitative information that is more readily acquired and more robust.

QDA methods aim to be compatible with decision theory to the extent possible. In particular, it is desirable that QDA methods be as sound and complete as possible: They should render decisions that are consistent with decisions recommended by quantitative decision theory and they should provide recommendations in as many cases as possible.

Sometimes the goals of QDA are incompatible: methods that are helpful with respect to one of the goals above sometimes make other goals more difficult to achieve. In particular, adapting a qualitative approach rather than a fully quantitative approach in order to facilitate knowledge elicitation, inference, or explanation introduces ambiguity and incompleteness. Some decisions that could be made using decision theory cannot be made using qualitative decision analysis. Worse, some qualitative approximations are unsound as well as incomplete: some qualitative methods introduce errors by recommending decisions that contradict the recommendations of decision theory.

This paper focuses on order of magnitude calculi. In particular, we focus on Wilson's order of magnitude decision theory (Wilson, 1995). It is based on a qualitative approach to probabilities called the κ (Kappa) *calculus* (Goldszmidt, 1992; Goldszmidt, 1995). The κ-calculus represents probabilities by ϵ^κ, where κ is an integral power of ϵ. Events are ranked according to κ. Events with larger κ are assumed to be negligible relative to events with smaller κ. Events that have $\kappa = 0$ are regarded as believed or at least likely, $\kappa = 1$ means an event is unlikely, $\kappa = 2$ very unlikely, and so on. The calculus is consistent with the axioms of probability as $\epsilon \to 0$, so ϵ is sometimes viewed as an *infinitesimal* and appeals are made to interpretations related to non-standard analysis. In practice, however, ϵ is usually given some finite fixed value greater than zero (Darwiche & Goldszmidt, 1994). In this paper, the order of magnitude decision theory is implemented by mapping probabilities and utilities to signed order of magnitude values as shown in Figure 1 and then expected utilities are computed using an implementation of Wilson's calculus. We call this Q-DA method "OoM" — short for "Order of Magnitude."

Although we do not have sufficient space for a complete description of Wilson's order of magnitude calculus, the main features of the calculus, as it has been adapted and implemented using fixed real ϵ values here, are as follows. In order to decide which action to take, expected utilities are computed using the following approximate operations. All quantities (e.g., probabilities, and utilities of outcomes) are qualified prior to further computation using the order of magnitude mapping described above. Note that since $\epsilon < 1$, larger powers correspond to smaller numbers. Sums of orders of magnitude are computed by taking their *min*, and products are replaced by sums since we are dealing with exponents. Expected utilities are simply sums of utilities weighted by their probabilities (converted to κ values). See (Wilson, 1995).

Here we focus on deciding whether to do an action or not. Examples of this type of problem include deciding whether to treat a suspected disease, deciding whether to rent or buy a house, and so on. A specific example of this type of decision problem, based on a medical diagnosis exercise in (Charniak & McDermott, 1985), follows. It involves a patient diagnosed to be suffering from an "enlarged thymus." The diagnosis is uncertain. The probability p_d that the thymus is enlarged is given as .75; there is a .25 probability $p_{\bar{d}}$ that the thymus is not enlarged. The goal of the exercise is to decide whether to operate to remove the thymus. Assume that removing an enlarged thymus costs 100 utility points but it results in a good situation with utility 850 and failing to operate on an enlarged thymus results in an extremely bad situation with utility -1000. Assume that operating on an ordinary thymus results in a situation with utility -150 and doing nothing on an ordinary thymus results in a slightly better situation with utility -100. In terms of odds, losses, and gains, $odds = 3$, $\delta_d = 1850$, and $\delta_{\bar{d}} = -50$; where $odds$ is $p_d/p_{\bar{d}}$, δ_d is the difference in utility of operating versus not operating given that the thymus is enlarged and $\delta_{\bar{d}}$ is the difference in utility of operating versus not operating given that the thymus is not enlarged. It is better to operate than to do nothing in this example.

The remainder of the paper focuses on methods for assessing ambiguity and incompleteness of alternative approaches to qualitative decision analysis. It also contains some results of experimental evaluations of a particular order of magnitude scheme (OoM) with different values of a "grain size" parameter ϵ. The significance of the

results is discussed, and implications for the design of new improved QDA methods are drawn.

2 Assessing incorrectness and ambiguity

Qualitative reasoning methods can be incomplete or unsound with respect to decision theory. Unsound methods can support incorrect decisions, disagreeing with the recommendations provided by decision theory. Incomplete methods can fail to make recommendations in some cases that can be decided using decision theory. Another way to look at this sort of incompleteness is to regard it as an example of the ambiguity inherent in qualitative reasoning. Ambiguity and errors are important issues that must be considered in designing qualitative methods for reasoning about tradeoffs.

In the remainder of this section, we discuss some quantitative metrics that can be used to measure the ambiguity and errors of qualitative decision methods either analytically or experimentally. The first metric that comes to mind is to measure the portion of a set of decision problems that are missed by a qualitative decision system. Adopt assumptions about the distribution of decision problems. Then, in experiments, generate test sets of problems under these assumptions and count how many are undecided by the qualitative system. In analysis, use the assumptions about the distribution to compute the probability that a random problem will be missed.

While this metric provides useful information, it fails to take into account the fact that some problems are more important than others; it treats all the problems as if they were equally important. The following metric is designed to weight problems by importance. The idea is to consider how much we might lose (or fail to gain) if we make an incorrect decision on a problem. In other words, each problem is weighted by the "cost" of a mistake: $m(\delta_d, \delta_{\bar{d}}, p_d) = |EU(op) - EU(\overline{op})|$

This measure can be used empirically by generating points, determining whether they are missed, and if so computing the associated cost. For example, we can generate test cases using a given distribution and plot bar charts comparing the initial distribution of problems versus the distribution of the problems that are not decided by QDA methods.

3 Empirical evaluation of O of M

In this section, we present some new empirical results of comparisons of proposed methods for qualitative decision analysis. These results are based on a computional experiment using 1000 tradeoff problems generated randomly using bounded uniform distributions on p, the probability of a predicate being true, and on four utilities of outcomes corresponding to whether or not the predicate is true and whether or not the action is taken. The action should be taken according to quantitative Bayesian decision theory in roughly half of the problems (508) and it should not be taken in the other cases. (None of the problems happen to be on the decision surface.)

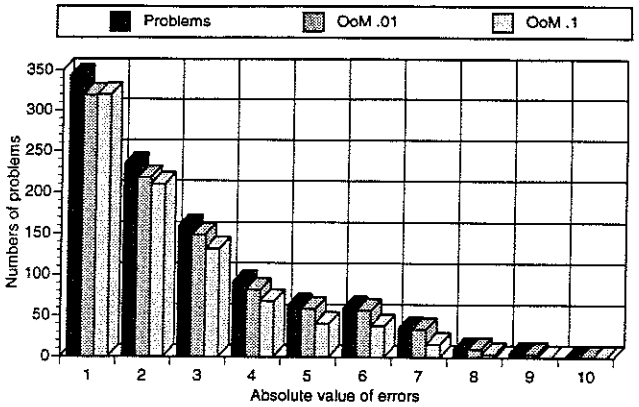


Figure 2: Ambiguity in order of magnitude decision analysis ($\epsilon = .01$ versus .1)

3.1 Order of magnitude calculi

Figure 2 shows results of evaluating OoM for two different values of ϵ, namely $\epsilon = .1$ and $\epsilon = .01$. The figure is a bar chart with three sets of bars. The horizontal axis corresponds to the absolute value of the difference between the expected value of doing an operation and the expected value of not doing it. The bars correspond to sets of problems such that the relative advantage of making the correct decision — or, equivalently, the expected cost of an error — has a value in the corresponding interval. The vertical axis shows the number of problems that fall in the intervals shown on the horizontal axis.

The first set of bars shows the distribution of the test problems. Most problems are unimportant in the sense that little is lost if they are decided incorrectly; few problems are important in the sense that errors lead to large losses. Over a third of the problems are in the lowest value bin. As the cost of errors increases, the number of problems declines. None of the 1000 problems fall in the top value bin. These bars, together with the axis y=0, serve to bound the quality of the decisions. The upper bound corresponds to not covering any of the decision problems. The axis y=0 corresponds to covering all of them.

The second set of bars corresponds to the test problems that are not covered by OoM with $\epsilon = 0.01$. Over 90% of the randomly generated tradeoff problems are not decided by OoM with $\epsilon = .01$. Setting $\epsilon = .1$ offers improved coverage. However, over 80% of the randomly generated tradeoff problems are undecided by OoM with $\epsilon = .1$.

A small number of problems are decided incorrectly by the order of magnitude calculi. When $\epsilon = .1$, 7 problems are decided incorrectly. When $\epsilon = .01$, 11 problems are decided incorrectly. All of these errors are in the lowest value bins.

The sets of ambiguous problems overlap but neither one contains the other. Analysis of the ambiguous problems decided by one order of magnitude calculus but not the other show that OoM with $\epsilon = .1$ solves most of the problems solved by OoM with $\epsilon = 0.01$ but not all of them. Subtracting the 829 problems not covered using

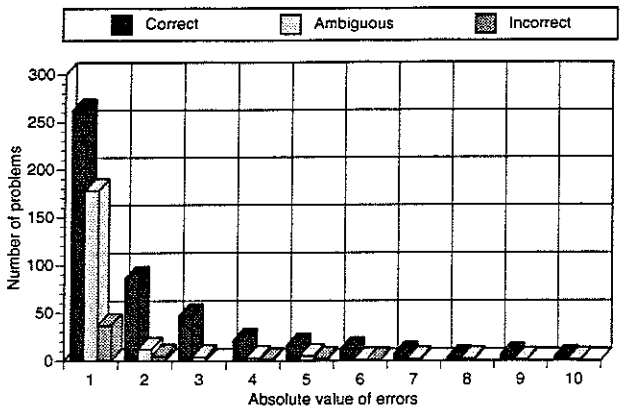


Figure 3: Frequency distribution of values of negative problems from a LogNormal distribution correctly classified, left unclassified, and classified incorrectly by Order of Magnitude Calculus ($\epsilon = .1$).

$\epsilon = .1$ from the 932 problems not covered using $\epsilon = 0.01$ leaves 110 problems. Only 7 problems not covered using $\epsilon = .1$ remain after subtracting the problems not covered using $\epsilon = 0.01$. Most of these problems are decided incorrectly (5 of them are errors). In other words, OoM with $\epsilon = 0.01$ misses a large number of problems that were covered correctly using $\epsilon = .1$ Many of these problems are high value problems. OoM with $\epsilon = 0.01$ covers a few problems that were not covered but provides the wrong decision in most of these newly covered cases.

3.2 Dependence on the problems

Uniform distributions were used in the preceding section in order to get measurements of the general coverage afforded by QDA methods. In addition, in many situations, we might have bounds on utilities but no information about which probabilities and values are more likely than others. In these situations, uniform distributions are recommended to reflect this lack of knowledge (Morgan & Henrion, 1990). If more is known about the problems that are to be expected, this information can be taken into account in the evaluation.

In this section, we consider how different approaches to QDA perform on a nonuniform distribution that reflects situations that often occur in practice. The distribution considered here is a multidimensional Log Normal Distribution centered on small probabilities and on high payoffs for doing the right thing. Log Normal Distributions are often appropriate for highly uncertain, nonnegative quantities and are obtained using Normal Distributions on the exponents (see (Morgan & Henrion, 1990)). Problems like diseases or device failures often have low prior probabilities in the initial stages of diagnosis and the utilities of correctly treating diseases and fixing malfunctioning devices is often high.

The results of an experiment involving 1000 decision problems (5-tuples of real numbers of the form $\langle p, u_{op,p}, -u_{\bar{o}p,p}, -u_{op,\bar{p}}, u_{\bar{o}p,\bar{p}}\rangle$ with the probability and utilities following Log Normal distributions ($\sigma = 1$) around the exponents $-4, 4, 4, 1, 1$, respectively, follow. An example decision problem is $\langle 0.000098727, 4236.82, -43173.9, -0.977757, 0.877305\rangle$. Note that in this example, the probability of the problem is very low, the payoff if we fix the problem is over \$4,000 but if we do not fix the problem we will lose more than \$40,000. The cost of fixing the problem unnecessarily or of not fixing a nonexistent problem are small.

No action should be taken in about 75% (766) of the randomly generated problems according to decision theory. The other 234 problems are cases where the action should be taken; there are no problems on the decision surface defined by decision theory.

With $\epsilon = .1$, about 6% (46 negative problems) are misclassified and roughly 24% of the problems (211 negative and 25 positive decision problems) are not covered by the order of magnitude calculus. However, about 70% the problems are correctly classified (509 of the 766 negative problems and 209 of the 234 positive problems).

4 Discussion

The order of magnitude calculus recommends the wrong decision in some cases. The number of errors made is relatively small, but could be important in some applications.

The order of magnitude calculus exhibits a high degree of ambiguity. This can be visualized graphically. Figure 4 shows the actual decision surface for this problem and approximate decision surfaces defined by cuboids based on orders of magnitude. Note that these cuboids do not correspond exactly to the order of magnitude calculi considered here. For example the cuboids do not misclassify decision problems, and they are defined in terms of the differences rather than the utilities of outcomes. However, they do provide some graphical clues that help explain the performance of the order of magnitude calculi. The figures show that orders of magnitude provide a relatively coarse mesh and thus a relatively crude approximation to the actual decision surface.

The performance of this QDA scheme is highly dependent on the types of problems that one expects to see in practice. The OoM calculus covers a small percentage of the problems generated under a uniform distribution. This is not surprising, since order of magnitude techniques cannot be expected to apply unless the values of probability and utility differ greatly.

When decisions depend on very different values, orders of magnitude can provide useful approximations. The results of the final experiment show that the order of magnitude calculus performs relatively well in such cases, for example on problems taken from a natural distribution intended to reflect the initial stages of diagnosis.

5 Related work

A number of researchers have developed qualitative approaches to probability (Druzdzel & Henrion, 1993; Henrion & Druzdzel, 1990; Wellman, 1988; Wellman, 1990). The *κ-calculus* (Darwiche & Goldszmidt, 1994; Goldszmidt, 1992; Goldszmidt & Pearl, 1991) is one of the most well developed approaches to qualitative probability.

Probabilistic approaches to AI problems are being extended to include utilities and values (Henrion, Breese,

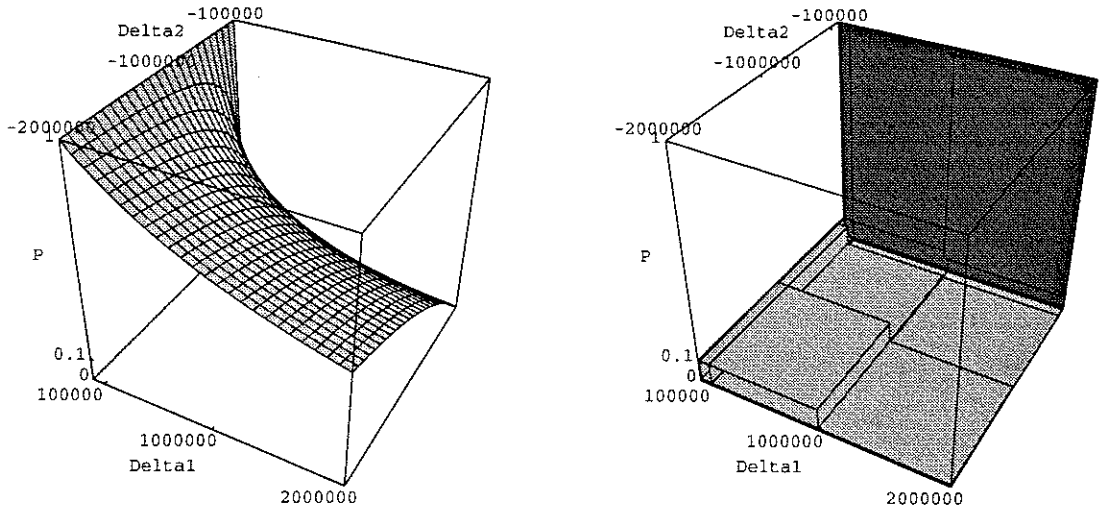


Figure 4: Decision Surface versus Approximation by Order of Magnitude Cuboids (with $\epsilon = .1$)

& Horvitz, 1991). Qualitative approaches to decision-making (Pearl, 1993; Wilson, 1995) and risk analysis (Krause, Fox, & Judson, 1995) are also being developed by a number of researchers.

Some work exists evaluating probabilistic reasoning methods that bears on qualitative approaches to probabilistic reasoning. Differences between fully quantitative probability theory and an order of magnitude version of the *κ-calculus* approach to probabilities on certain types of problems (problems whose Bayesian networks form chains or fuse inputs) have been documented by (Darwiche & Goldszmidt, 1994; Henrion, Darwiche, Goldszmidt, Provan, & Del Favero, 1994). Computational experiments on a Bayesian network for a car diagnosis problem show that this approach performs well for problems with small prior probabilities of faults but it performs poorly when the priors are significant (larger than ϵ) (Henrion, et al., 1994). These experiments also indicated that values of ϵ near .1 are better than smaller values. Very little work exists evaluating qualitative approaches to decision making.

6 Conclusions

In our view, the results indicate that order of magnitude methods that render decisions using powers of a real number ϵ to represent probabilities and utilities can provide solutions to a significant number of important decision problems (as shown in Section 3.2). However, these methods are relatively weak: they leave most problems undecided, even in simple situations. We conclude that order of magnitude methods alone will not provide a sufficient basis for qualitative decision analysis. One way of addressing this problem that we are currently exploring involves combining order of magnitude methods with other qualitative techniques.

The results indicate that qualitative approaches to probabilistic reasoning and decision theory that interpret ϵ as an infinitesimal and that draw analogies to non-standard analysis should be abandoned in favor of order of magnitude approaches that treat ϵ as a real number with a value of, say 1/10. This conclusion is based on the observation that fewer problems are decided by order of magnitude methods with smaller ϵ values (more problems are left undecided). It may seem counterintuitive that OoM works less well with smaller values of epsilon, since one is accustomed to approximations becoming more accurate as ϵ approaches zero. However, this can be explained by the fact that all the landmarks available using $\epsilon = .01$ can be obtained as multiples of $\epsilon = .1$ since $.01 = .1^2$ so a larger ϵ provides finer meshes and thus more accurate approximations.

Finally, the results reported here clearly indicate that it is important to take the types of problems that one expects to encounter into account. The evaluation methods described here can be used to take problem distributions and the importance of problems into account while evaluating and attempting to identify appropriate QDA methods.

References

Charniak, E., & McDermott, D. (1985). *An Introduction to Artificial Intelligence.* Reading, Mass.: Addison-Wesley.

Darwiche, A., & Goldszmidt, M. (1994). On the relation between kappa calculus and probabilistic reasoning. In R. L. de Mantaras, & D. Poole (Ed.), *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence* (pp. 145–153). Seattle, WA: Morgan Kaufmann.

Druzdzel, M., & Henrion, M. (1993). Efficient reasoning in qualitative probabilistic networks. *Proceedings of the National Conference on Artificial Intelligence* (pp. 548–553). Washington, D. C.

Goldszmidt, M. (1992). *Qualitative probabilities: A Normative framework for commonsense reasoning* (Ph.D. dissertation R-190). UCLA, Cognitive Systems Laboratory.

Goldszmidt, M. (1995). Fast belief update using order-of-magnitude probabilities. In P. Besnard, & S. Hanks (Ed.), *Uncertainty in Artificial Intelligence: Proceedings of the Eleventh Conference* (pp. 208–216). Montréal: Morgan Kaufmann.

Goldszmidt, M., & Pearl, J. (1991). System Z^+: A formalism for reasoning with variable strength defaults. *Proceedings of the National Conference on Artificial Intelligence* (pp. 399–404). Anaheim, CA: AAAI Press/The MIT Press.

Hanks, S. (Eds.). (1994). *Working Notes of the AAAI Spring Symposium on Decision-Theoretic Planning.* Palo Alto, CA: AAAI.

Heckerman, D. E., Horvitz, E. J., & Nathwani, B. N. (1992). Towards normative expert systems: Part I The Pathfinder project. *Methods of Information in Medicine, 31,* 90–105.

Henrion, M., Breese, J. S., & Horvitz, E. J. (Winter 1991). Decision analysis and expert systems. *AI Magazine,* pp. 64–91.

Henrion, M., Darwiche, A., Goldszmidt, M., Provan, G., & Del Favero, B. (1994). An Experimental comparison of infinitesimal and numerical probabilities for diagnostic reasoning. *Working Notes of the Fifth Workshop on the Principles of Diagnostic Reasoning.* New Paltz, NY.

Henrion, M., & Druzdzel, M. J. (1990). Qualitative and linguistic explanations of probabilistic reasoning in belief networks. In P. Bonissone (Ed.), *Sixth Conference on Uncertainty in AI* (pp. 10–20). Cambridge, MA: Association for Uncertainty in AI.

Henrion, M., Provan, G., Del Favero, B., & Sanders, G. (1994). An Experimental comparison of numerical and qualitative probabilistic reasoning. In R. L. de Mantaras, & D. Poole (Ed.), *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence* (pp. 319–326). Seattle, WA: Morgan Kaufmann.

Krause, P., Fox, J., & Judson, P. (1995). Is there a role for qualitative risk assessment? In P. Besnard, & S. Hanks (Ed.), *Uncertainty in Artificial Intelligence: Proceedings of the Eleventh Conference* (pp. 386–393). Montréal: Morgan Kaufmann.

Morgan, M. G., & Henrion, M. (1990). *Uncertainty: A Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis.* London: Cambridge University Press.

O'Rorke, P., & El Fattah, Y. (1993). Qualitative and order of magnitude reasoning about diagnostic decisions. In C. Preist (Ed.), *The Fourth International Workshop on Principles of Diagnosis* (pp. 136–148). Aberystwyth, Wales: University College.

O'Rorke, P., El Fattah, Y., & Elliott, M. (1993). Explaining and generalizing diagnostic decisions. In P. E. Utgoff (Ed.), *Machine Learning: Proceedings of the Tenth International Conference* (pp. 228–235). San Mateo, CA: Morgan Kaufmann.

Pearl, J. (1993). From conditional oughts to qualitative decision theory. In D. Heckerman, & A. Mamdani (Ed.), *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence* (pp. 12–20). Washington, D. C.: Morgan Kaufman.

Wellman, M. (1988). Qualitative probabilistic networks for planning under uncertainty. In J. F. Lemmer, & L. N. Kanal (Eds.), *Uncertainty in Artificial Intelligence 2* (pp. 197-209)North Holland.

Wellman, M. P. (1990). Graphical inference in qualitative probabilistic networks. *Networks, 20*(5), 687–701.

Wilson, N. (1995). An order of magnitude calculus. In P. Besnard, & S. Hanks (Ed.), *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* (pp. 548–555). Montréal: Morgan Kaufmann.

Intelligent Assistance for Navigating the Web

Christopher A. Welty
Vassar College
weltyc@cs.vassar.edu

Abstract

The Untangle Project is an attempt to apply KR&R techniques to the problem of finding information on the ever-expanding World Wide Web. There are two key enabling technologies that allow for Untangle to work: a deep ontology of the kind of information that can be found on the web. and an HTML interface that provides on the fly access to the knowledge-base. The interface allows for queries formulated in the underlying representation language (Classic), which provides a far more expressive facility for searching than is currently available in any web navigation tools.

Subject Areas: Ontologies, Electronic and On-Line Information, KR&R User Interfaces, Representing large domains.

1 Introduction

There can be no doubt that the Word Wide Web is growing at an incredible rate. The current default model for Web use is casual "surfing" – many users consider The Web a novelty and browse around without any specific goal. Clearly this model will change, if The Web is to survive, and tools will have to be available that support users who are searching for specific information.

This paper describes aspects of the Untangle Project, an effort to provide intelligent assistance to someone searching The Web for information. This assistance is manifested by several technologies which are briefly described in this paper.

The first technology is an ontology and knowledge-base implemented in the the description logic Classic [Brachman, et al., 1989], a descendant of KL-ONE. The ontology provides a deep representation of the information that is available on The Web.

The second technology is a web interface for Classic, which allows the knowledge-base to be accessed interactively through any web browser (such as netscape or mosaic). This access is not to a static set of HTML files that are generated automatically at periodic intervals, but to the live knowledge-base.

The paper closes with a discussion of how Untangle can be used as a vehicle to demonstrate the power and utility of KR in real applications.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/311 ©1996 FLAIRS

This paper is available in HTML format at http://www.cs.vassar.edu/faculty/welty/papers/untangle/flairs-96_1.html. The HTML version of the paper contains numerous hot links to the references and to live demos of the Untangle system.

2 Ontology for Electronic Information

From a KR perspective, the central issue in the Untangle project is the ontology for representing information that is in electronic form.

2.1 Previous Work

This work began before the World Wide Web became popular, and the initial goal was to support intelligent email distribution. The basic elements of the ontology remain unchanged from the KBEDS system originally described at FLAIRS-94 [Welty, 1994a], however some minor changes have been made to bring it in line with the standard bibliographic ontology that is part of the Ontolingua Ontology Library [Gruber, 1994]. This updated ontology is described in [Welty, 1995], and the reader is referred to either of these previous papers for more in-depth details of the main ontological elements.

2.2 New Goals of the Research

With the explosive growth and popularity of The WWW as a medium for disseminating information, it seemed far more interesting, practical, and trendy to apply the ontology for electronic information to this huge tangled mess. The original goals were therefore subtly altered: to provide intelligent assistance for navigating The Web.

A popular navigational tool available on The Web is Yahoo, which presents a hierarchical view of subject areas that can be browsed top-down, or searched for keywords. The incredible popularity of this service indicates that it is useful and desperately needed, and clearly this form of information organization (a hierarchy of increasingly specific subjects) lends itself quite easily to expression in a KR system.

There are numerous shortcomings to Yahoo that can easily be overcome with KR technology. There is little place in the Yahoo taxonomy for web pages that can't be explicitly classified. A person's home page, for example, will typically not be

be located on Yahoo, since it describes a person, not a specific topic.

The granularity of the Yahoo categories is often too large, and there is no other information about the web pages that might be used to constrain a search, other than keywords appearing in the link name. The Artificial Intelligence category, for example, has over 150 links. If you are interested in only AI conferences, however, you can either browse the 150 links, or try to find everything in the AI category that has the word "Conference" in its name. The latter search would miss such important items as FLAIRS-96 or the AAAI Spring Symposium Series, and clearly it completely lacks the much needed ability to search for "conferences in Florida."

In addition, without something to tie them together, related web pages will simply be listed in alphabetical order, thus separating such things as the web pages that describe the various special tracks of FLAIRS. If there was knowledge that these different pages describe aspects of the same event, it could be helpful to someone searching for this information.

Clearly KR has much to offer as a technology for assisting web navigation, and the Untangle Project began as an effort to apply KR to this domain.

2.3 Obstacles

Criticisms of Yahoo aside, it is an extremely useful service, and the initial plan for the Untangle Project was to at least duplicate the functionality of Yahoo, and build from there. This deceptively simple goal, however, presented some obstacles whose solutions were fairly good lessons for ontological development.

The initial approach used to integrate the Yahoo subject taxonomy into the ontology of electronic information was to create a concept called `web-page`, below which were the concepts, on per subject. This was essentially the Yahoo hierarchy, since Yahoo is only capable of representing Web pages, and not the objects or events behind them.

This approach works for web pages, but when the individuals in the `event-or-object` category were considered, it quickly became clear that they, too, needed to be grouped by subject. Suddenly the ontology grew to the point where there was a subject taxonomy below the concept `person` (i.e. `science-person`, `art-person`, etc...), below `conference` (i.e. `science-conference`, `art-conference`, etc...), below `publication`, `organization`, and so on.

Aside from the obvious problem of too many redundant concepts, there was nothing to indicate that sets of concepts such as `ai-person`, `ai-page`, `ai-conference`, etc., were somehow related. It would seem like an obvious and desirable inference to be able to make that if a web page is an information item of a conference, and the web page is considered an ai-page, then the conference must be an ai-conference. Figure 1 shows a situation with the simple inference missing.

Clearly this would be fairly easy to remedy with a rule that

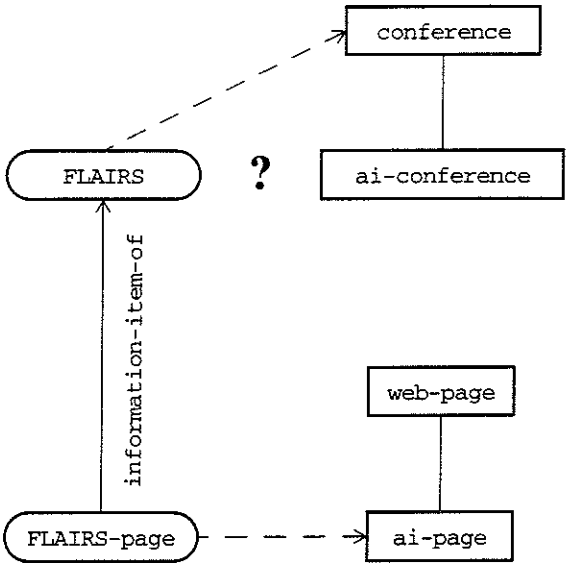


FIGURE 1. Missing inference.

says any individual of an `ai-page` is the `information-item-of` an individual of `ai-conference`. The problem is that a separate rule is needed to infer this between `database-conference` and `database-page`, `art-conference` and `art-page`, and between every pair of related subjects.

The next approach was to take out the subject sub-taxonomies, and create a concept called `subject`, whose individuals were all the different subjects, and these individuals could fill the role `has-subject` in all events, objects, web pages, etc. This worked a little better, since one rule would now cover all the cases mentioned above, but the hierarchical ordering of subjects was lost. It was possible to create a role called `sub-subject` between individuals of `subject`, but this was undesirable because Classic uses subsumption as its central form of inference [Brachman, et al., 1991], and the built-in subsumption facilities do not recognize any named roles as a specialization link.

2.4 Solution

The obvious solution may seem fairly clear at this point, but it was stubbornly avoided for semantic reasons. The solution was to create a fourth kind of concept, call it `subject` for the moment, that was the parent concept of a taxonomy of subjects, such as `computer-science`, `artificial-intelligence`, `art`, `drama`, etc. Figure 2 shows an example of this fourth taxonomy merged with the previous example.

With this approach the taxonomy of subjects is preserved, the natural subsumption relationship between Classic concepts is used, there is no duplication in the concept hierarchies, and a single rule can be used to infer the subject of an event or object if it is known of an information item. Every individual is an individual of one of `event-or-object`, `information-item`, or `information-view`, and in addition is an

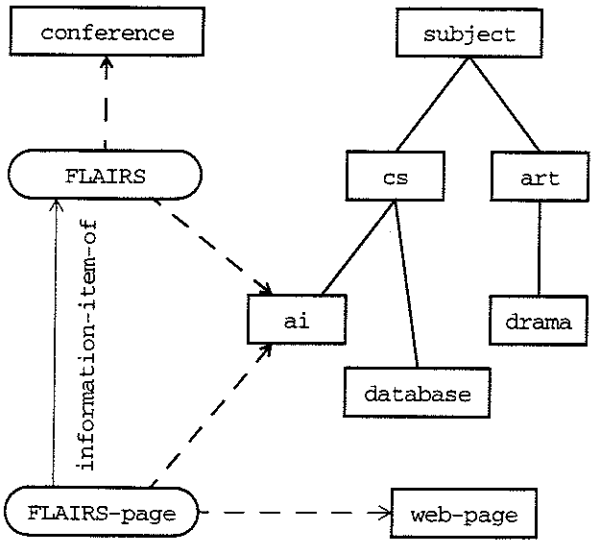


FIGURE 2. First step towards a solution.

individual of any number of concepts in the `subject` taxonomy.

Clearly this is an elegant solution to the problem in an operative sense, but semantically there is a problem with it. Every individual of a concept is also an individual of all the concept's parents, thus in Figure 2 the individual `FLAIRS` is an individual of `cs` and `subject`. Consider the normal meaning of being an individual: the individual *is a* whatever the concept is. For FLAIRS, it makes sense to say that it is a conference, but is it a subject, or even an AI or CS?

This semantic difficulty may seem trivial, but the Untangle Project was founded on the very notion that a deep and more accurate representation of information on the web would facilitate access to that information. Compromising on the semantics, on the very way the information is interpreted, seemed to violate the principles of the project and to create the possibility for confusion.

Religion aside, the final (perhaps *current* is more appropriate) solution was actually not far off. Simply by adding `thing` to the names of the concepts in the subject taxonomy, and changing the name of the top concept to `represented-thing`, the semantics are no longer confusing. FLAIRS is a `conference`, an `ai-thing`, `cs-thing`, and `represented-thing`.

3 Web Interface

Probably the most significant enabling technology that makes the Untangle Project able to actually demonstrate how KR can offer improvements over existing web navigation tools is the Common LISP Hypermedia Server (CL-HTTP) [Mallery, 1994]. This is an HTTP server implemented in Common LISP, which enables HTML forms, queries, and searches to invoke LISP functions. These LISP functions output HTML, which is interpreted by the client browser and displayed.

The significance of this development is profound in two ways:

1. It immensely simplifies the generation of a user interface, since the web browser handles all formatting, graphics, and viewing. The size of the Untangle Project team would have precluded a usable interface otherwise.
2. The actual data in the knowledge-base can be *shared*. Any number of people, anywhere on the net, can access the data and see live results in a presentation format (HTML) they are familiar with. Most knowledge bases can only be used by a single person at a time.

The project focused its early stages on developing a web interface for Classic. This interface displays concept and individual descriptions as dynamic web pages – the HTML is generated on the fly to reflect the state of the Classic object *at the time of the query*. References to other concepts or individuals in these descriptions are hot links to the functions that will display them. Concept taxonomies can also be displayed as nested HTML lists. Within concept or individual descriptions, role taxonomies are displayed as nested HTML lists as well.

KR systems are well known for producing lots of information, much of which is not useful to a person. The interface suppresses a lot of this using per-concept meta-information to control which roles are displayed. It also by default suppresses the display of information inherited along the role hierarchies if the original information is also displayed. All this default behavior can be overridden by the user using forms accessible through every description page.

The interface also overrides the default display of an individual if any of that individual's parents have a *display-function*. The purpose of this is particularly for information-items. The roles of an information item typically link it to the object or event the information is about, and the views of that piece of information. For example, the individuals `Chris-resume` and `Chris-resume.html` are shown below:

```
Chris-resume::
(and information-item description
  (fills information-item-of Chris)
  (fills has-information-view
         Chris-resume.html
         Chris-resume.ps))

Chris-resume.html::
(and information-view html-view
  (fills information-view-of Chris-resume)
  (fills URL
         "http://www.cs.vassar.edu/..."))
```

A user will rarely want to see these actual descriptions. When interested in a particular information item, such as `Chris-resume`, a user will want to see the information itself. Every information view has a URL role which is filled with a string that identifies the location of the information. When a user (through various mechanisms such as the result of a query or clicking on a role in another individual) selects an information item to be viewed, the display function for `information-`

`items` bypasses the actual individual description, retrieves the string from the URL slot of the item's view, and passes that back to the web browser (which should know what to do with a URL). If the item has more than one view, as is the case in the example above, the display function generates an HTML list in which each item is a link to the URL of one of the views. In other words, it lets the user choose the view.

4 Showing off KR

The Untangle Project has the potential of being a highly visible testimony to the power of knowledge representation techniques in making information more accessible. The advantages are more than simply adding the events and objects underlying web pages to Yahoo. Adding this depth to the representation then makes it possible to do inference, and it is inference that sets KR techniques apart from others [Welty, 1994b]. This section provides some examples of useful inference that the ontology currently supports.

4.1 Classifying Events and Objects

It is useful, as discussed in previous sections, to be able to infer where events and objects fit in the `represented-thing` taxonomy based on how their information items are classified. This is accomplished with a description rule in Classic.

Classic description rules are attached to concepts and fire on any individuals of that concept that pass a filter. This rule is attached to `event-or-object`, and the filter insures that the individual has an `information-item`. When the rule fires, it returns the parents below `represented-thing` of the information item, and adds these to the parents of the `event-or-object`.

Note that the converse of this rule does not work, an information item can not be assumed to be of the same type as the event or object it describes. A person, for example, may have many interests and thus be classified under several unrelated subjects, yet not all the web pages associated with that person will describe all those subjects.

The utility of this rule can probably only be realized in practise. The work of populating the knowledge-base with individuals is triggered by discovering interesting web pages and other on-line information. When discovered, the appropriate information items and views are created, and placed in the represented-thing taxonomy. Especially in the case where a new information item describes an object or event that already exists in the knowledge-base, new subject information is typically not added to the object or event (for no other reason than that it is usually overlooked by the person doing it). This automated assistance then helps keep the knowledge-base more accurate.

4.2 Common Navigational Rules

In addition to providing a deep representation of the knowledge that is behind web pages, the ontology also supports rules that express common navigational rules employed by "experts" at navigating the web. These rules can be used to find information that Untangle does not have.

There are basically two kinds of these rules:

- *Good places to start a search.* The ontology supports an understanding that certain web sites, such as Yahoo in general, or others that are more domain-specific, are good places to start a search.
- *How to construct an address.* Most expert web navigators fall back on several simple rules about where web servers might be. For example, if you are looking for information about a college, try www.college.edu. This may seem inherently obvious to some, but based on experience at Vassar this useful piece of knowledge is not very widespread.

4.3 Automatic Classification

This section describes ideas that have not been implemented or explored deeply yet, and are the subjects of the next phase of research in the Untangle Project.

Classic provides a powerful and flexible language for describing the sufficient conditions for subsumption. In other words, part of a concept description can be the sufficient conditions for classifying an individual under it. We intend to study how this can be used in conjunction with a web crawler (a program that follows links around the web) to automatically populate the knowledge-base.

This effort is not as far-fetched as it may seem. The development of HTML is moving towards providing more meta-information in the language, and encouraging HTML developers to make use of it. This meta-information can be of tremendous use in a system capable of doing inference.

4.4 Subsumption as Search

The subsumption language of Classic also provides an excellent query language. Without needing to know before-hand what kinds of queries will be done (and thus forming a hash table), subsumption-based searches can still be performed in a reasonable amount of time.

The more you know about what you are looking for, the more information that is provided in the query, the faster it goes. The ontology was specifically designed to help a user find previously seen web pages or papers with only a vague recollection of what they were.

Here are some example queries supported by the ontology, and their accompanying translations into Classic:

- "Find all the AI Conference pages"

```
(and information-item
     (all information-item-of
          (and conference ai-thing)))
```

- "Find all the home pages of people interested in AI."

```
(and home-page
```

```
(all information-item-of
     (and person ai-thing)))
```

- "Find all the home pages of people interested in AI and who are in industry."

```
(and home-page
     (all information-item-of
          (and person ai-thing
               (all employee-of company))))
```

- "Find all AI papers available on-line"

```
(and paper ai-thing
     (at-least 1 has-document-text))
```

- "Find all papers published in the FLAIRS-96 conference proceedings that are available on-line."

```
(and paper
  (at-least 1 has-document-text)
  (fills year-published 1996)
  (all published-in
       (all proceedings-of
            (fills name "FLAIRS-96"))))
```

These queries can by entered through the interface, and their results viewed as HTML. A primer on writing Classic queries will be available through the interface as well.

5 Conclusion

This paper does not present any startling new discoveries. It is the application of of tried and true Knowledge Representation techniques to a highly visible domain: navigating the web.

The contributions of this work are threefold:

- Making KR technology available on the WWW can potentially demonstrate to a much larger community the practical benefits of using KR&R. This paper has described a *user* interface that is completely generic to Classic, and has been used for other Classic knowledge-bases, yet is flexible enough to support the display of objects outside the representation (actual web pages, in this case). It provides a "look and feel" that is familiar to a very large and still growing community. Web interfaces to KR systems will be a prerequisite for success of any knowledge sharing system in the future.
- The ontology for electronic information merged with a subject taxonomy carries the standard bibliographic ontology [Gruber, 1994] one step further. After further practical testing and use, the new ontology will undergo more formal and rigorous analysis for submission to the Ontolingua Ontology Library [Gruber, 1993].
- The application domain for this project is quite real, and by no means of a toy scale. While Untangle is not near the size of, e.g. Yahoo, this is simply a matter of manpower. The size of the knowledge-base grows daily, and new uses and benefits of the deep ontology and inference capabilities are constantly being found.

The original motivation for moving this research from Email distribution [Welty, 1994a] to the WWW was to demonstrate to the Digital Library community that KR techniques can improve on what is being done with Information Retrieval (IR) [Welty, 1994b]. The Untangle Project therefore hopes to be a showcase for KR&R. This will be tough going, Digital Libraries and the WWW in general are strongly tied to IR, but the deeper knowledge and inference capabilities that characterize KR do have quite a bit more to offer.

Acknowledgments

Anthony Schorr and Derek Gaasch have worked on various aspects of this project.

REFERENCES

[Brachman, et al., 1989] Brachman, R., Resnick, L., Borgida, A. and McGuinness, D. *CLASSIC/DB: A Structural Data Model for Objects.* ATT Bell Labs Technical Report. 1989.

[Brachman, et al., 1991] Brachman, R., McGuinness, D., Patel-Schneider, P., Borgida, A. and Resnick, L. Living with CLASSIC: When and How to Use a KL-ONE-Like Language. *Principles of Semantic Networks.* Morgan Kaufman. Pp. 401-456. May, 1991.

[Gruber, 1993] Gruber, T. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition,* 5(2):199-220, 1993.

[Gruber, 1994] Gruber, T. Introduction to the Bibliographic Data Ontology. Available as http://www-ksl.stanford.edu/knowledge-sharing/ontologies/html/bibliographic-data.text.html.

[Lenat, et al., 1990] Lenat, D., Shepherd, M., Pratt, D., Pittman, K. and Guha, R. Cyc: Towards Programs with Common Sense. *Communications of the ACM.* **33**(8). Pp. 30-49. Aug, 1990.

[Mallery, 1994] Mallery, J. A Common LISP Hypermedia Server. *Proceedings of The First International Conference on The World-Wide Web.* Geneva: CERN, May 25, 1994.

[Welty, 1994a] Welty, C. A Knowledge-Based Email Distribution System. *Proceedings of the 1994 Florida AI Research Symposium (FLAIRS-94).*

[Welty, 1994b] Welty, Chris. Knowledge Representation for Intelligent Information Retrieval. *Proceedings of the CAIA-94 Workshop on Intelligent Access to Digital Libraries.* March, 1994.

[Welty, 1995] Welty, Chris. An Ontology for Electronic Information. Available as http://www.cs.vassar.edu/faculty/welty/papers/on-line-rep/dl-96_1.html.

BELIEF AND PROBABILITY BASED DATABASE MINING

Pawan Lingras

Department of Computer Science, Algoma University College
Sault Ste. Marie, Ontario, Canada, P6A 2G4.
e-mail: lingras@tbird.auc.on.ca

Abstract

This paper describes a unified approach for generating rules from relational databases. The relational databases have been successfully used in the past for generating probabilistic rules. The databases used for probabilistic rule extraction consist of tuples for which all the values of the attributes are precisely known. The proposed approach uses a belief function based generalization of the existing approach by allowing for a set of possible attribute values. Such an approach can be useful in situations where the value of an attribute may be unknown or may be one of a few possible choices. Whenever the available information is sufficient, the proposed method automatically generates probabilistic rules.

Keywords: Belief functions, expert systems, probability functions, rule generation, database mining.

1 INTRODUCTION

Database mining can be defined as the process of mining for implicit, previously unknown, and potentially useful information from very large databases by efficient knowledge discovery techniques. It is one of the most promising research topics in the fields of database systems and machine learning. Researchers have demonstrated efficient generation of probabilistic rules from databases (Wong and Ziarko, 1987). The conventional rule generation procedures use databases for which the precise values of all the attributes for each tuple are known. This paper argues that such a restriction in the rule extraction process is unnecessary. If the restriction to specify precise attribute values is removed, it will make the data collection process more flexible. If a data collector or an expert cannot precisely answer a question, she will be able to specify a set of possible answers. The proposed approach also provides an ability to specify the missing attribute values by using the entire domain as a set of possible values.

The rules generated using complete databases use probability functions for specifying the degree of uncertainty. A probability function is one of the most useful and meaningful measures of uncertainty. The theory of probability has played an important role in decision making in a wide variety of applications. However, in many situations in information science, the available information may not be sufficient to employ the theory of probability. The incomplete database used in this study provides an example of such a situation. If it is not possible to obtain precise values of attributes in a database, it will not be possible to generate probabilistic rules. Belief functions (Shafer, 1976), a generalization of probability functions, are useful in some of the situations where the available information is insufficient to employ probability functions. The rule generation procedure from incomplete databases, discussed in this paper, uses belief functions as the measurement of uncertainty.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/316 ©1996 FLAIRS

2. BRIEF OVERVIEW

For completeness, this section provides a brief overview of the belief functions, the probability functions and the conventional rule extraction process.

2.1 Belief and Probability Functions

The discussion in this paper uses the notion of *frame of discernment* (or simply a *frame*) which is defined as a set of all possible answers to a given question (Shafer, 1976). It is assumed that the answers in the frame are mutually exclusive and exhaustive.

Let T be a frame of discernment. A probability function $P:2^T \rightarrow [0,1]$ satisfies the following three axioms:

P1. $P(\varnothing) = 0$,

P2. $P(T) = 1$, and

P3.

$$P\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1} P(A_i) - \sum_{i<j} P(A_i \cap A_j) + \ldots + (-1)^{n+1} P\left(\bigcap_{i=1}^{n} A_i\right),$$

where $A_i, A_j \subseteq T$. A belief function $Bel:2^T \rightarrow [0,1]$ satisfies the following three axioms:

B1. $Bel(\varnothing) = 0$,

B2. $Bel(T) = 1$, and

B3.

$$Bel\left(\bigcup_{i=1}^{n} A_i\right) \geq \sum_{i} Bel(A_i) - \sum_{i<j} Bel(A_i \cap A_j) + \ldots + (-1)^{n+1} Bel\left(\bigcap_{i=1}^{n} A_i\right).$$

It can be easily seen that the condition B3 for belief functions is less restrictive than the condition P3 for probability functions. Hence, a belief function is a generalization of the probability function.

The theory of belief functions uses a few other functions such as the *plausibility functions* and the *basic probability assignments* (bpa) which are relevant to the discussion in this paper. A plausibility function $Pl{:}2^T \rightarrow [0,1]$ is given by:

$$Pl(A) = 1 - Bel(\overline{A}),$$

where $\overline{A}$ is the set theoretic complement of *A*. A basic probability assignment $m{:}2^T \rightarrow [0,1]$ obeys the following axioms:

M1. $m(\varnothing) = 0$, and

M2. $\sum_{A \in 2^T} m(A) = 1$.

Belief functions and plausibility functions can be derived from basic probability assignments as:

$$Bel(A) = \sum_{B \subseteq A} m(B) \quad (1)$$

$$Pl(A) = \sum_{B \cap A \neq \varnothing} m(B) \quad (2)$$

2.2 Probabilistic Rule Generation from Complete Databases

The database used for the extraction of rules consists of one or more *student* attributes, and an *expert* attribute. Let us assume that an expert can classify an object into three categories $x_1, x_2,$ and x_3. Let the objects be represented as tuples in a database. Let *Expert* be the attribute which describes whether a given object is $x_1, x_2,$ or x_3, i.e. the domain of *Expert* is $D_{Expert} = \{x_1, x_2, x_3\}$. The objective is to use some other evidence to emulate the expert classification. Let *J* be the *student* attribute (Wong and Ziarko, 1986) which describes the evidence, and let the domain for *J* be a set $D_j = \{j_1, j_2, j_3\}$. Let $U = \{e_1, e_2, \ldots, e_m\}$ be the set of all the objects represented by a relation *r* over the relation scheme $\{ID, J, Expert\}$, where *ID* is an attribute which is used to identify each object, such as e_1, e_2. Let $X_{\{x_k\}}$ be the set of all the objects in *U* which are classified as x_k, and let $E_{\{j_i\}}$ be the set of all the objects in *U* whose value for attribute *J* is j_i, for i and $k = 1,2,3$. This information can be used to generate the following probabilistic rules:

If $J = j_i$, then $Expert = x_k$ with a probability of

$$P(X_{\{x_k\}} | E_{\{j_i\}}) = \frac{|E_{\{j_i\}} \cap X_{\{x_k\}}|}{|E_{\{j_i\}}|}.$$

Pawlak, *et al.* (1985) have shown that the function $P(X_{\{x_k\}} | E_{\{j_i\}})$ is a probability distribution. The above probability distribution can be extended using P3 to get the values of probability for non-singleton sets.

3. INCOMPLETE DATABASES

The previous section discussed the conventional methodology for extraction of probabilistic rules from complete databases. This section proposes a similar rule extraction process from incomplete databases. The proposed rule extraction process uses belief and plausibility functions as measures of uncertainty.

3.1 Belief and Plausibility Based Rule Generation

The extraction of rules from a database discussed in section 2.2 is based on the assumption that values of attributes for each tuple in the database are precisely known. Such an assumption may not always be valid in practice. In the section 3.2, we will consider the accumulation of information from different sources where such an assumption is not valid. In general, providing an ability to specify a set of possible values for a given attribute can improve the usefulness of the rule extraction procedure. In the proposed approach, the values of attributes are subsets of their domains. The *student* and the *expert* attributes in the conventional rule generation systems described in section 2.2 can be used to illustrate the proposed approach. The information in the proposed approach will be stored in a database relation r' with the relation scheme $\{ID, J', Expert'\}$, where the domain of J' is $2^{D_j} - \{\varnothing\}$ and the domain of $Expert'$ is $2^{D_{Expert}} - \{\varnothing\}$. Table 3.1 shows an example of such a database.

For each value $q \in 2^{D_j} - \{\varnothing\}$, it is possible to define a set E_q such that

$$E_q = \{e_h | q = value_{J'}(e_h)\}. \quad (3)$$

Similarly, a set X_p for each value $p \in 2^{D_{Expert}} - \{\varnothing\}$ is defined as:

$$X_p = \{e_h | p = value_{Expert'}(e_h)\} \quad (4)$$

One can construct two basic probability functions (*bpa*) $m_q{:}2^{D_{Expert}} \rightarrow [0,1]$ and $m^q{:}2^{D_{Expert}} \rightarrow [0,1]$ as follows:

$$m_q(\varnothing) = 0, \qquad m_q(p) = \frac{\left| X_p \cap \left(\bigcup_{q \subseteq s} E_s \right) \right|}{\left| \left(\bigcup_{q \subseteq s} E_s \right) \right|} \tag{5}$$

$$m^q(\varnothing) = 0, \qquad m^q(p) = \frac{\left| X_p \cap \left(\bigcup_{q \cap s \neq \varnothing} E_s \right) \right|}{\left| \left(\bigcup_{q \cap s \neq \varnothing} E_s \right) \right|} \tag{6}$$

Theorem 3.1:

The functions $m_q : 2^{D_{Expert}} \rightarrow [0,1]$ and $m^q : 2^{D_{Expert}} \rightarrow [0,1]$ defined by eq. (5) and eq. (6) are basic probability assignments.

Proof:
The condition M1 for basic probability assignment is satisfied by the definitions in eq. (5) and eq. (6).

Since every object e_h has precisely one value from the domain $2^{D_{Expert}} - \{\varnothing\}$, condition (ii) for *bpa* can also be easily proved for both the basic probability assignments. □

The basic probability assignments m_q and m^q can be used to define a belief function $Bel_q : 2^{D_{Expert}} \rightarrow [0,1]$ and a plausibility function $Pl^q : 2^{D_{Expert}} \rightarrow [0,1]$ as follows:

$$Bel_q(p) = \sum_{s \subseteq p} m_q(s) \tag{7}$$

$$Pl^q(p) = \sum_{s \cap p \neq \varnothing} m^q(s) \tag{8}$$

The belief and plausibilistic rules can then be defined as follows:
If $J' = q$, then $Expert' = p$ with $Bel_q(p)$ and $Pl^q(p)$ given by eq. (7) and eq. (8).

ID	*J'*	*Expert'*
e_1	$\{j_1\}$	$\{x_1\}$
e_2	$\{j_2\}$	$\{x_2\}$
e_3	$\{j_3\}$	$\{x_1\}$
e_4	$\{j_1, j_2\}$	$\{x_2\}$
e_5	$\{j_1, j_2\}$	$\{x_1\}$
e_6	$\{j_2\}$	$\{x_2\}$
e_7	$\{j_1\}$	$\{x_1, x_2\}$
e_8	$\{j_1, j_2\}$	$\{x_3\}$
e_9	$\{j_2\}$	$\{x_3\}$
e_{10}	$\{j_3\}$	$\{x_3\}$
e_{11}	$\{j_3\}$	$\{x_1\}$
e_{12}	$\{j_1\}$	$\{x_1\}$
e_{13}	$\{j_1\}$	$\{x_2\}$
e_{14}	$\{j_1, j_2\}$	$\{x_2\}$
e_{15}	$\{j_1\}$	$\{x_3\}$
e_{16}	$\{j_1\}$	$\{x_1, x_2\}$
e_{17}	$\{j_1, j_2\}$	$\{x_1\}$
e_{18}	$\{j_1\}$	$\{x_2, x_3\}$
e_{19}	$\{j_2\}$	$\{x_3\}$
e_{20}	$\{j_2\}$	$\{x_3\}$
e_{21}	$\{j_3\}$	$\{x_3\}$
e_{22}	$\{j_1, j_2, j_3\}$	$\{x_3\}$
e_{23}	$\{j_1, j_2, j_3\}$	$\{x_3\}$
e_{24}	$\{j_2\}$	$\{x_1\}$
e_{25}	$\{j_1\}$	$\{x_2\}$
e_{26}	$\{j_1\}$	$\{x_1\}$
e_{27}	$\{j_3\}$	$\{x_2\}$
e_{28}	$\{j_2\}$	$\{x_1\}$
e_{29}	$\{j_1, j_2, j_3\}$	$\{x_3\}$
e_{30}	$\{j_3\}$	$\{x_1\}$

Table 3.1 Database with Values of Attributes Specified as Subsets of the Domain

p	X_p
$\{x_1\}$	$\{e_1, e_5, e_{11}, e_{12}, e_{17}, e_{24}, e_{26}, e_{28}, e_{30}\}$
$\{x_2\}$	$\{e_2, e_4, e_6, e_{13}, e_{14}, e_{25}, e_{27}\}$
$\{x_3\}$	$\{e_8, e_9, e_{10}, e_{15}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{29}\}$
$\{x_1, x_2\}$	$\{e_7, e_{16}\}$
$\{x_1, x_3\}$	$\varnothing$
$\{x_2, x_3\}$	$\{e_{18}\}$
$\{x_1, x_2, x_3\}$	$\varnothing$

Table 3.2 Values of X_p for the Example Database

q	E_q
$\{j_1\}$	$\{e_1, e_7, e_{13}, e_{15}, e_{16}, e_{18}, e_{25}, e_{26}\}$
$\{j_2\}$	$\{e_2, e_6, e_9, e_{19}, e_{20}, e_{24}, e_{28}\}$
$\{j_3\}$	$\{e_3, e_{10}, e_{11}, e_{21}, e_{27}, e_{30}\}$
$\{j_1, j_2\}$	$\{e_4, e_5, e_8, e_{14}, e_{17}\}$
$\{j_1, j_3\}$	$\varnothing$
$\{j_2, j_3\}$	$\varnothing$
$\{j_1, j_2, j_3\}$	$\{e_{22}, e_{23}, e_{29}\}$

Table 3.3 Values of E_q for the Example Database

It should be noted here that belief and plausibility functions are obtained from two different basic probability assignments. Hence, they do not follow the relationships specified in the theory of belief functions. Table 3.2 shows the values of E_q and X_p for the proposed rule extraction process as applied to the database shown in table 3.1. The following are some of the rules generated using tables 3.2 and 3.3:

If $J = j_1$ then $Expert = x_1$
with a belief of $\frac{3}{9}$ and a plausibility of $\frac{7}{17}$.

If $J = j_1$ then $Expert = x_2$
with a belief of $\frac{2}{9}$ and a plausibility of $\frac{7}{17}$.

If $J = j_1$ then $Expert = x_3$
with a belief of $\frac{1}{9}$ and a plausibility of $\frac{6}{17}$.

If $J = j_1$ then $Expert = x_1$ or x_2
with a belief of $\frac{7}{9}$ and a plausibility of $\frac{11}{17}$.

If $J = j_1$ or j_2 then $Expert = x_1$
with a belief of $\frac{5}{21}$ and a plausibility of $\frac{8}{24}$.

If $J = j_1$ or j_2 then $Expert = x_1$ or x_2
with a belief of $\frac{11}{21}$ and a plausibility of $\frac{12}{24}$.

If $J = j_2$ then $Expert = x_1$
with a belief of $\frac{1}{7}$ and a plausibility of $\frac{3}{15}$.

Note that totally 49 such rules can be generated using the proposed approach. However, for brevity all possible rules are not included in this paper.

The approach proposed in this section is a generalization of the conventional probabilistic rule generation. If the database used for the rule generation were complete, the belief and plausibility based rules described above will reduce to the probabilistic rules as indicated by the following theorem.

Theorem 3.2.
The belief and plausibility functions generated by eq. (7) and eq. (8) will reduce to the probability function $P\left(X_{\{x_k\}} | E_{\{j_i\}}\right)$ given in section 2.2.

Proof: It is sufficient to prove that the basic probability assignments given by eq. (5) and eq. (6) will reduce to the probability distribution

$$P(X_{\{x_k\}} | E_{\{j_i\}}) = \frac{\left|E_{\{j_i\}} \cap X_{\{x_k\}}\right|}{\left|E_{\{j_i\}}\right|}$$ (Shafer, 1976).

Since all the values of the attributes are singleton sets, all the E_q and X_p for non-singleton p and q are empty sets. Elimination of empty-sets from the eq. (5) and eq. (6) results in the following:

$$m_q(\{x_k\}) = m^q(\{x_k\}) = \frac{\left|E_{\{j_i\}} \cap X_{\{x_k\}}\right|}{\left|E_{\{j_i\}}\right|}.\square$$

3.3. Don't Know Values and Accumulation of data

In some of the applications of the intelligent systems, the available information may come from different sources and/or may be based on different sample spaces. In such applications, it will be necessary to make decisions based on all the available evidence. Since different pieces of information may focus on different sets of attributes, it will be difficult to obtain information about all the objects for all the attributes in the combined approximation space.

For example, assume that one data collection uses a student attribute *J* (as described in section 2.2 and 3.2) to obtain rules to predict the values of the *Expert* attribute. Let another data collection use another student attribute called *M* to predict the values of the *Expert* attribute. Let D_M be the domain for attribute *M*. If the sample spaces from these two data collections are combined, for the objects from the first data collection, there may not be any information about the attribute *M*. Similarly, for the objects from the second data collection, there may not be any information about the attribute *J*. That means the combined sample space will consist of several objects with no information for either the attribute *J* or the attribute *M*. That is, the combination of evidence may result in databases with missing values for certain attributes. More generally, an ability to specify *don't know* or missing values will provide a greater flexibility in the data collection and the rule extraction process.

The approach proposed in this study can easily accommodate databases with missing values. The missing values of certain attributes will be specified as the entire domain of the attribute. In the example discussed in the previous paragraph, the combined sample space will be stored in a database with a relation scheme $\{ID, J', M', Expert'\}$, where the domain of J' is $2^{D_J} - \{\varnothing\}$, the domain of M' is $2^{D_{M'}} - \{\varnothing\}$, and the domain of $Expert'$ is $2^{D_{Expert}} - \{\varnothing\}$. The tuples for which the value of *J* is missing will use the D_J as the value for J'. Similarly, the tuples for which the value of *M* is missing will use the D_M as the value for M'.

4. CONCLUSION

The conventional rule extraction procedures use complete databases, where all the values of the attributes are precisely known. The resulting rules use probability functions as the measure of uncertainty. The approach presented in this study is a generalization of the conventional approach by allowing for a set of possible attribute values. The available information in the proposed approach may not be sufficient to generate probabilistic rules. When the values of all the attributes are precisely known, the resulting rules are shown to be the same as the probabilistic rules generated by the conventional approach. Such an approach can be useful in situations where the value of an attribute may be unknown or may be one of a few possible choices.

References

Chokr, B. A. and Kreinovich, How Far are We From the Complete Knowledge? Complexity of Knowledge Acquisition in the Dempster-Shafer Approach. in *Yager, R. Fedrizzi, M., and Kospryk, J. (Editors) Advances in the Dempster-Shafer Theory of Evidence,* John Wiley and Sons, New York, pp. 555-576.

De Finetti, B. (1974). *Theory of Probability, Vol. I,* Wiley, New York.

Fine, T. L. (1973). *Theories of Probability*, Academic Press, New York.

P.J. Lingras and S.K.M. Wong, Two Perspectives of the Dempster-Shafer Theory of Belief Functions, *International Journal of Man-Machine Studies*, Vol. 33, pp. 467-487.

Pawlak, Z., Wong, S.K.M, and Ziarko, W. (1988). Rough sets: probabilistic versus deterministic approach, *International Journal of Man-Machine Studies*, 29, pp. 81-95.

Shafer, G. (1976). *A Mathematical Theory of Evidence*, Princeton University Press, Princeton. N.J.

Shafer, G. (1987). Probability Judgment in Artificial Intelligence and Expert Systems, *Statistical Science*, 2, pp. 3-16.

Skowron, A. and Grzymala-Busse, J. (1991). From the Rough Set Theory to the Evidence Theory, in *Yager, R. Fedrizzi, M., and Kospryk, J. (Editors) Advances in the Dempster-Shafer Theory of Evidence,* John Wiley and Sons, New York, pp. 193-226.

Smets, P. What is Dempster-Shafer's Model, in *Yager, R. Fedrizzi, M., and Kospryk, J. (Editors) Advances in the Dempster-Shafer Theory of Evidence,* John Wiley and Sons, New York, pp. 5-34.

Wong, S.K.M, and Ziarko, W. (1986). INFER -- an adaptive decision support system, *Proceedings of the Sixth International Workshop on Expert Systems and Their Applications*, Vol. 1, pp. 713-726, Avignon, France.

Yao, Y.Y., Li X., Lin, T.Y. and Liu, Q. (1994). Representation and Classification of Rough Set Models, *Conference Proceeding of Third International Workshop on Rough Sets and Soft Computing*, November 10-12, San Jose, California, pp. 630-637.

KNOWLEDGE INTEGRATION OF MEDICAL TERMINOLOGICAL SOURCES: AN ONTOLOGICAL MEDIATION

Geri Steve, Aldo Gangemi, Angelo Rossi Mori

Reparto Informatica Medica, Istituto Tecnologie Biomediche, CNR, Roma, Italy
email: steve@relay.itbm.rm.cnr.it; {aldo, angelo}@color.irmkant.rm.cnr.it

Abstract

In order to start up a comprehensive process of medical terminological knowledge integration, we analyzed — from an ontological viewpoint— about 2700 high-level medical textual items (objects and relations) from 5 terminological systems. A methodology (ONIONS) has been defined to compare and integrate conceptual paradigms (ontologies) inherent in independent sources, and we applied it to build an ontological framework (ON7), able to explain the ontological correspondences underlying terminological correspondences. The ontological differences in the systems are exemplified by mapping those terms which refer to Body Part. ON7 has been partially implemented in the GRAIL formalism for the GALEN European project.

1. INTRODUCTION

Is my *body part* the same as your *body part* and his/her *body part*? Analysing terminological systems widely used in health care, the answer cannot be *-yes-*.

For example, in the UMLS semantic network[13] a *"body part, organ or organ component"* is a kind of *"fully formed anatomical structure"*; in SNOMED[3], *body part* is implicitly referred to by several instances within the *Topography* module; in the current GALEN model[8,19] (see below), *body part* is a *normal part* of a *body*. The reason for dissimilarity is that ideal knowledge (either background knowledge or operational knowledge of a health care operator) of *body part* is not in any non-trivial sense correspondent to the phrase 'body part' nor to a body part 'out there'. Phrases in terminological systems are only links to that ideal knowledge; systems organization, depending on the particular task they are made for, selects some relevant issues in the context of the ideal knowledge.

All those *body parts* are equally acceptable. Only, one cannot guess, from the organization of a single system, all the valid usages of that phrase in the world out there.

Terminological systems select some issues of the ideal knowledge, like pointing the finger to a site in the mind of a health care operator.

Today, integration of large knowledge bases is a main issue[5,11,12,15,17], which embraces terminological knowledge integration as well.

But having my, your, and his/her *body part* integrated requires a model which says at least that *body part* is a *completely developed proper part* of an *organism* (*organism* should be an *intentional biologic structure*, and a *biologic structure* should be something as an *atemporal object* pertaining to the *biological world*); moreover, a *body part* could be described as a *body system* when it is conceived according to a *function* which is *embodied* in it. The construction of a comprehensive formal model needs an integration which is *ontological*, not simply terminological.

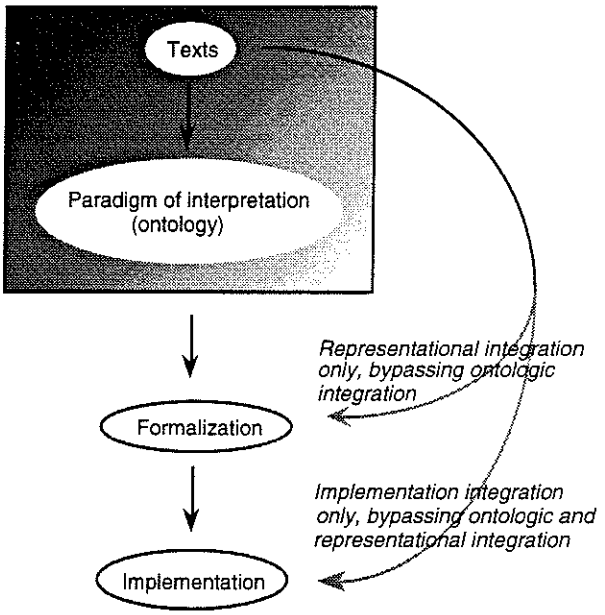


Figure 1: Ontological integration is prior to representational integration, which concerns formal languages, database schemata, etc.

The first aim of our ontological integration has been to develop a CORE model of medical concepts which — inside GALEN project[8] — supports the conceptual link among different terminological systems or repositories.

Terminological systems developed in medicine —such as classifications, nomenclatures, semantic networks— are dependent on ontologies, usually implicit, but coherent with specific tasks (epidemiology, indexing, retrieval, acquisition, expert systems)[20,4].

Notice (*fig.1*) that the ontological integration envisaged here is prior to or at least independent from representational integration. In fact, representational integration addresses to heterogeneity of *formal languages* (e.g.[17]), or to heterogeneity of *data base schemata* (e.g.[25,27]). Ontological integration addresses the heterogeneity among *paradigms of knowledge organization*.

In most knowledge representation, ontological integration is bypassed, while task-oriented systems usually bypass representational integration as well.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/321 ©1996 FLAIRS

1.1 General and domain ontologies

Here we need a distinction between *general* and *domain ontologies*[10]. General ontology deals with the most general structures of reality, independently on the knowledge used in a given domain. Domain ontology describes a specific field of activities, knowledge, etc. It has to take into account the usage of knowledge, a lot of different cognitive arrangements of reality, and some specific features of the language exploited in the domain. Medical ontology is a very complex domain ontology involving many sub-domains, such as the ones of medical specialties.

Our research (besides the GALEN project) is related to other efforts —CEN/TC251/PT003[2], CANON[5], and Ontolingua[6,11,15]. In fact, knowledge integration in medicine is a good experiment of integration of domain specific knowledge within the wider general medicine knowledge; however it is not so wide and complex as general knowledge is.

Following such a strategy, our work has these two main goals: a) generalizing a framework to integrate terminological knowledge from various medical sources, by analysing their domain ontologies, and b) defining a new and open domain ontology which merges a general ontology with various domain ones.

2. MATERIALS AND METHODS

Up to now, our research has taken into account 5 sources (terminological repositories): the UMLS semantic network[13] (all ~170 semantic types and relations, and their defined combinations), SNOMED-III[3] (~600 most general concepts) and GMN[7] (~700 most general concepts) nomenclatures, ICD10[24] classification (~250 most general concepts), and the CORE model previously developed by the GALEN project[8] (version 5g[8,20], non-ontologically oriented, all ~2000 items).

UMLS has a hierarchical structure, includes relations, provides free-text definitions and combinations of types and relations. It has a browser but does not allow to create new assertions. It uses the MeSH thesaurus and other nomenclatures as its 'bottom level'.

SNOMED and GMN have some general axes (partially hierarchical), do not apply relations, are homogeneous between top and bottom parts. ICD is hierarchical, has no relations, is homogeneous between top and bottom parts. CORE model v.5g is hierarchical, applies relations, is homogeneous between top and bottom parts, and has a terminological engine which allows to compose concepts and relations —with some degrees of validity— into canonical forms, and provides tools to debug and browse large models.

2.1 The ONIONS methodology

In order to analyse domain ontologies, we developed the ONIONS (ontologic integration on naïve sources) methodology (*fig. 2*)[9,22].

It creates a common framework to interpret the criteria that organize a set of terminological systems. In other words, it allows to work out coherently a domain ontology for each system, which can be then compared and mapped to an integrated system.

We examined some general ontologies[17,21,23] as heuristical sources. Note that we consider concepts as objects or relations in a homogeneous way, independently on how they have been implemented in a given system: this is a major strength of ontological analysis.

Step I): *Extracting Terms*

Selecting relevant sets of terms from terminological sources (***source terms***): code definitions or key-words from classifications, nomenclatures, coding systems, thesauri. This step has hooks to corpora formation techniques and textual types definition and acquisition (not investigated here).

Step II): *Setting (Surface) Criteria*

Focusing and defining criteria of classification (***surface ontology criteria***): the question to answer is: which is the differential criterion within a group of homogeneous concepts from the same source, typically between two children-concepts of the same parent concept? Surface criteria could be maximally to minimally explicit in the source:

—*formally explicit*, as in GALEN 5g, which exploits the GRAIL formal subsumption[18]:

> (BodyProcess which hasOutcome AreaOfPolyposis) name PolyposisProcess
>
> which states that a *polyposis* is a process of the body and is denoted by its usual outcome: an *area of polyposis.*

—*dictionary-like explicit*, as in textual definitions of UMLS:

> Activity: An operation or series of operations that an organism or machine carries out or participates in

—*implicit, but inferable from extension*, as in SNOMED. These criteria are extensionally inferable because they can be suggested by their children and their position in the hierarchy:

> Topography
> Cardiovascular System
> Heart and Pericardium
> ...
> Blood Vessels
> ...
> ...
> [SNOMED-III]

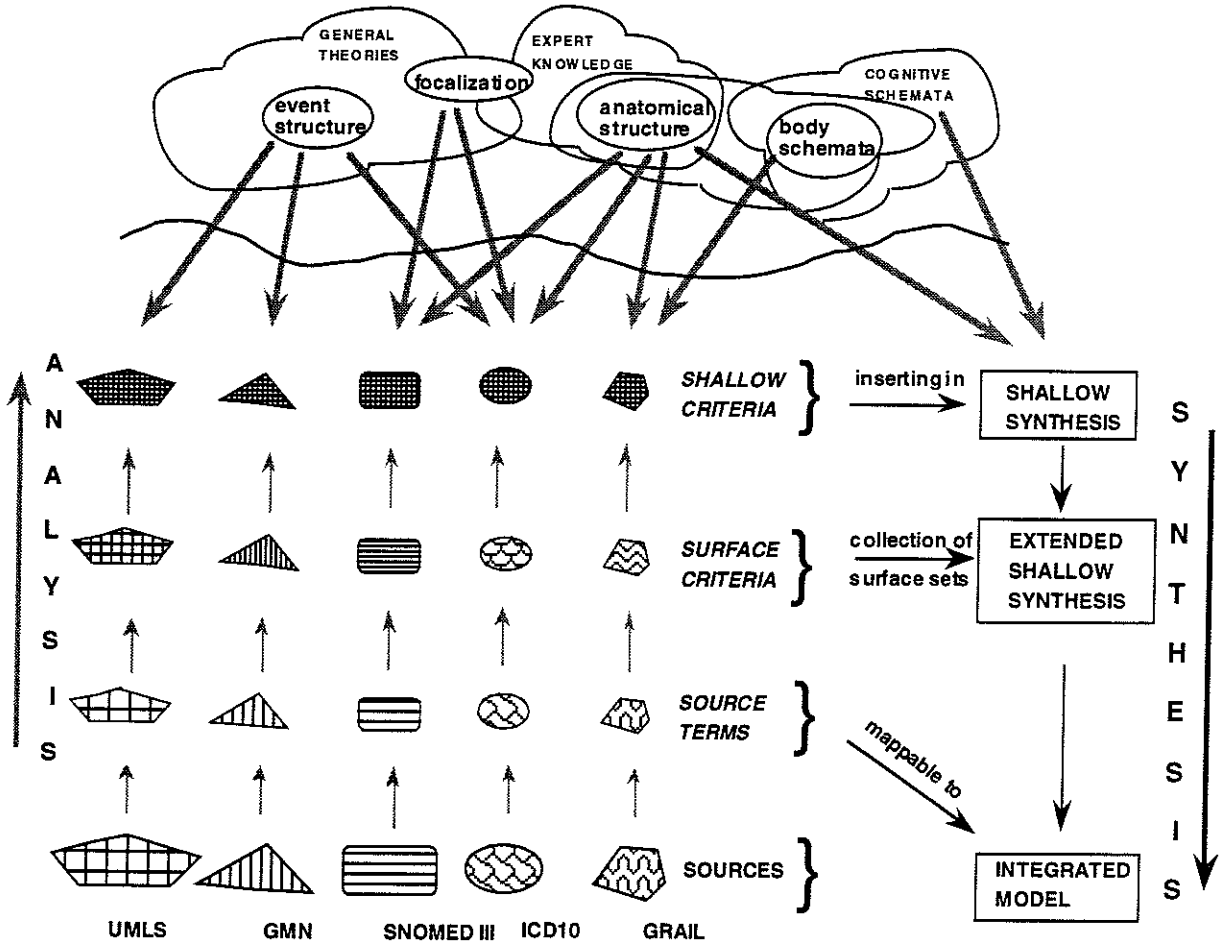


Figure 2: the procedure of *ontological integration*: from the heterogeneous terminological repositories to an integrated model. Different patterns inside figures symbolize different levels of ontological explicitness, the maximal being the shallow criteria application. Bold arrows from upper clouds show some links among shallow theories inside the context of general knowledge, and shallow criteria.

Obviously, formal and dictionary-like explicit criteria are extensionally explicit as well (i.e. their position in a hierarchy is also checkable).

—*thoroughly implicit*, as in natural language texts: these can be only analysed through modeller's knowledge and experts' help:

> ... respiratory failure is defined as respiratory dysfunction resulting in abnormalities of oxygenation or CO_2 elimination severe enough to impair or threaten function of vital organs...[26]

Step III): *Setting Shallow (meta)Criteria according to Theories*

Understanding and defining theories which underlie criteria of classification (***shallow ontology criteria***). This is much an analysis of the contextual framework of differential criteria. This setting implies investigating a large heuristics (with several degrees of formality): experts' knowledge, general theories (either formal or informal ontological statements), cognitive constraints which structure our knowledge: *spatial orientations*, *kinaesthetic schemata*, naïve *force dynamics*, etc. This heuristics can help us defining (*hypothesizing*) a set of shallow theories (see e.g. Table 1), that have to be refined (*checking*) when subsuming the surface criteria from Step II.

Step IV): *Mapping Criteria to a Shallow Synthesis and Collecting a Surface Synthesis*

Since shallow criteria are assigned according to a set of defined shallow theories, and shallow theories are developed by comparing the single surface criteria sets to hypothetical general theories, it is straightforward to compare shallow criteria sets and mapping them on a unique framework (***shallow synthesis***). Some examples of how heterogeneity is referred to a homogeneous framework are given in *Tab. 3.* A mapping of the apparently similar term *body part* (see Introduction) from ontologically heterogenous sources is given in *tab. 4.*

On the other hand, shallow criteria are not extensively detailed, thus usually we have to recollect a plenty of surface distinctions to populate the integrated model (***surface synthesis***).

TOPOLOGICAL STRUCTURES, including MEREOLOGICAL STRUCTURE	what is a *structure*, a *part*, a *whole*, a *group*, etc.?; *discrete-continuous, neighbour, adjacent, distant, connected-separate*
BODY SCHEMA	*verticality, laterality, front-back*, other *direction schemata*; possibly *kinesthetic schemata*
PERCEPTION	*interoception, exteroception*
SIMILARITY	what *generalization* means?
INTENTION	what is an *internal goal*?
GRANULARITY, including FOCUS	which *layers-strata-plans* are taken into account in a domain?
EVENT STRUCTURE, including: NATURAL/ARTIFICIAL, MORPHOGENESIS STATIC/DYNAMIC,BALANCE	what is a *process*, a *situation*, a *function*, a *role*, a *cause*, an *effect*, a *plan*, a *product*, an *outcome*, a *temporal line* or *cycle*, etc.?
BIOLOGICAL	what is *living-animate-biological* (lab) vs. not-lab?
MORPHOLOGY	general patterns; what is a *form* for?
SOCIAL ACTION	how a *causal event* can be situated *socially*?
INFERENCE STRUCTURE	how *event structure* and *topological structure* are projected (simulated) in reasoning?
NORMALITY	what is a *rule*? how a rule interacts with events and *diagnosis*? what's a *disease*?
SCALE AND QUANTITY	what is *more or less*?; *measurement* and *unit* as a framework of reference, statistical rules, etc.; *duality, plural objects*

Table 1: shallow (meta)criteria and hints to theories. A provisional set of (medical) shallow theories has been selected to now, according to surface criteria extracted (bottom-up) and current literature on formal ontology and cognitive semantics (top-down). It should be stressed that shallow criteria proposed are not modules: some are more fundamental than others (possibly in the order above), but they are deeply interconnected. For instance, FOCUS is connected to intentional perception; cognitive TOPOLOGICAL STRUCTURES are in a chicken-and-egg relation with BODY SCHEMATA; INFERENCE STRUCTURE largely depends on SCHEMATA, PERCEPTION, SIMILARITY, EVENT and TOPOLOGICAL STRUCTURES, etc.

StepV) *Formalizing an Integrated Model*

The model is formalized —and eventually implemented— according to the *syntactic* and *pragmatic constraints* of the logic formalism and/or of the computational tool employed.

In the following section we introduce some outcomes of the application of ONIONS methodology to the abovementioned five medical terminological repositories, and —in the Discussion— we will come back to the *body part* example: its integration and formalization.

3. RESULTS

The results of our research currently are:
i) a mapping amomg ~4000 heterogeneous terms from the 5 terminological sources;
ii) ON7: a formal domain ontology including the definition of ~2800 high-level items. ON7 has been partially modelled in two different formalisms: GRAIL terminological language[8] (as part of the ontological foundation of the CORE model within the GALEN project, cf.[19]), and first-order predicate logic;
iii) the computer-based model of ON7, implemented as a GRAIL application[6] (CORE model v.6f, running on Smalltalk).

It includes ~800 computational definitions and allows the computational mapping of ~1000 heterogeneous items from UMLS, SNOMED, and GRAIL5g, to ON7[9].
Since our guiding example concerns the partonomy of organisms, *fig. 1* shows a fragment of the top-level of ON7, with relevant items related to body structures.All items in ON7 are defined by two prime unary predicates: *Object* and *Viewpoint*. The kinds of Viewpoint act as different viewpoints (or contexts, perspectives, strata)[a] on the same object, through the prime binary predicate:

[a] The terminological (and modelistic) choices should depend on the assumptions of the ontological engineer: a 'realist' looks for Objects within a given 'stratum' of reality; a 'nominalist' simply constructs different 'contexts' of predication; a 'cognitivist' figures out different 'perspectives' from which (s)he observes an Object. To a certain extent, choice is a matter of implied metaphors (real strata versus discourse contexts versus perceptual orientation), but the modelistic outcome will not be equivalent, in fact, the item *BiologicStructure* would be defined in three slightly different ways:

realistic: $\forall x$: BiologicStructure. x: Structure $\wedge$ T(x, BiologicStratum)
where T means: *belongs to*, or *depends on*

nominalistic: $\forall x$: BiologicStructure. x: Structure $\wedge$ T(x, BiologicContext)
where T means: *is interpreted within*

cognitive: $\forall x$: BiologicStructure. x: Structure $\wedge$ T(x, BiologicLayer)
where T means: *is interpreted from the viewpoint of*

Here the third choice has been adopted, considering the extreme rigidity of pure 'strata of reality', as well as the too generic 'context'.

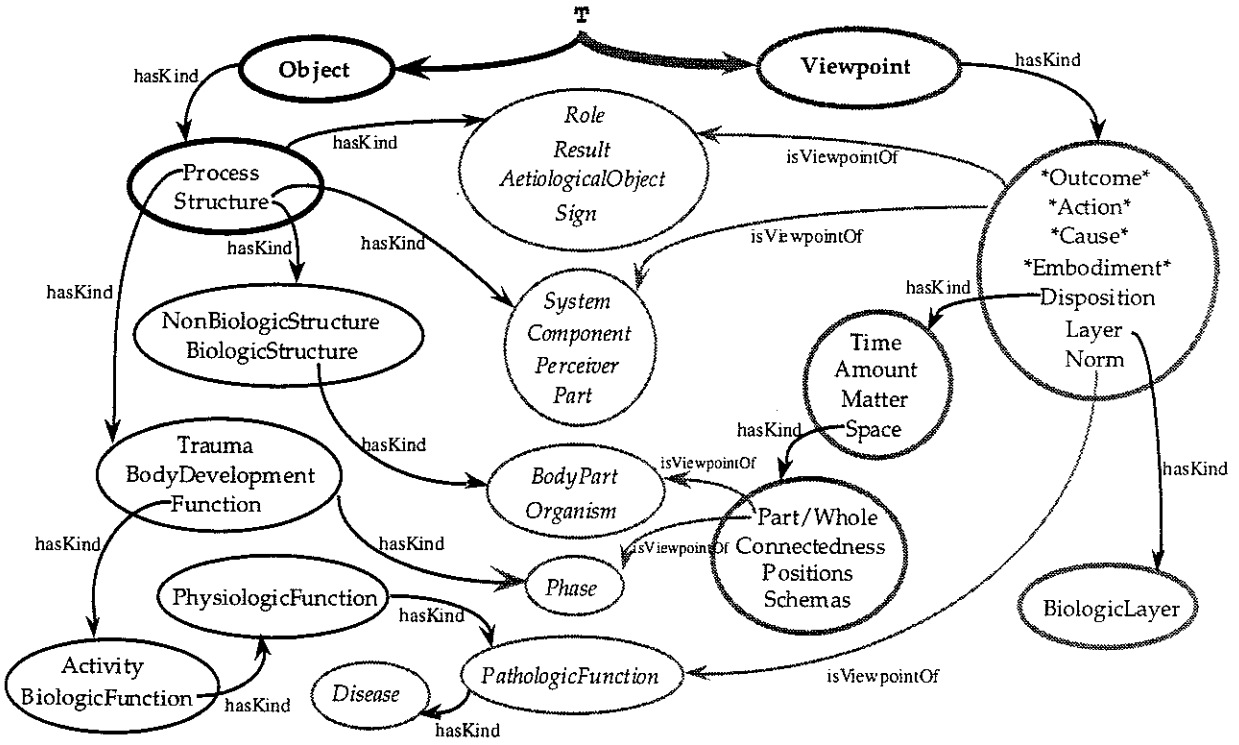


Figure 3: Simplified version of top-level connections in the model ON7. Concepts are inside ellipses. Concepts within stars are n-ary predicates (i.e. implemented as relationships, n>1). Black arcs are inverse-ISA (haKind) links. Grey arcs (isViewpointOf, say, the predicate T) and gray circles are for Object/Viewpoint combinations.

For Objects:
$\forall$x: Structure. x: Object $\wedge$ T(x, Atemporality)
$\forall$x: Process. x: Object $\wedge$ T(x, Interval)

For Structures:
$\forall$x: BiologicStructure. x: Structure $\wedge$ T(x, BiologicLayer)
$\forall$x: Organism. x: BiologicStructure $\wedge$ T(x, Intention)
$\forall$x: BodyPart $\exists$y: Organism. x: BiologicStructure $\wedge$ $<_p$(x, y)

For Functions and Systems:
$\forall$x: Function $\exists$y: Object. x: Process $\wedge$ hasOutcome(x, y)
$\forall$x: System $\exists$y: Function. x: Structure $\wedge$ embodies(x, y)
$\forall$x: BiologicFunction $\exists$y: Organism $\exists$z: System. x: Function $\wedge$ $<_c$(z, y) $\wedge$ (embodies(y, x) $\vee$ embodies(z, x))
$\forall$x: BodySystem $\exists$y: BiologicFunction. x: BiologicStructure $\wedge$ embodies(x, y)

For the notion of Development:
$\forall$x: BodyDevelopment $\exists$y: Organism. x: Process $\wedge$ hasOutcome(x,y) $\wedge$ T(y, GeneticLayer) $\wedge$ embodies(y, x)

Table 2: Sample formulas for the definition of main concepts concerning structures, functions, systems. The predicate *embodies* is used for conveying the notion of any Process featured by any Structure.

T, which intuitively means that an Object has a given interpretation[b]. For example, the concept *biologic structure* can be "seen" from (or interpreted within) "spatial viewpoints" such as WHOLE or PART (which are, or can be, implemented as attributes) generating concepts such as *organism* or *body part*.
Instances of Viewpoint include (*fig.3*):

i) unary predicates for viewpoints of space, time, granularity, normality;
ii) binary predicates for viewpoints of causality, embodiment, outcome, intentional activity.

[b] T has at least these logical properties:

$\neg$(T(xy) $\rightarrow$ T(yx)) T is non-simmetric
(T(xy) $\wedge$ T(yz)) $\rightarrow$ T(xz) T is transitive

The predicate T is actually a focalizer: it stresses the interpretation in a context or the viewpoint for an Object. On the other hand, kinds of Viewpoint might be formalized directly as predicates on Objects, but with such a notation their role as viewpoints could be understated.
Notice that items in ON7 are natural language terms, but their interpretation is constrained by model definitions. In other words, the explicit treatment of dependences among terms provides a *context* for interpreting the terms; this context will be a *multi-local* context which accounts for the local ontologies implied in the knowledge (terminological) sources which are being integrated. *Tab.2* shows some partial definitions of concepts in an order-sorted logic with the mereological operators: '$<_p$' (transitive, structural part) and '$<_c$' (non-transitive, functional part) (cf.[1] for some useful formal treatment of medical partonomy), and some constants filling the second argument of the predicate T for unary viewpoints or theories.
Tab. 3 shows some sample mappings (mostly for the topic *body part* and its cognates) from terminological systems to ON7.

UMLS items	ON7 definitions of consistent mappings
PhysicalObject	$\forall$x: Structure. x: Object $\wedge$ T(x, Atemporality)
AnatomicalStructure	$\forall$x: BodyPart $\exists$y: Organism. x: BiologicStructure $\wedge <_p(x, y)$
FullyFormed AnatomicalStructure	$\forall$x: CompletelyDevelopedBodyPart $\exists$y: Organism $\exists$z: Development. x: BiologicStructure $\wedge <_p(x, y) \wedge$ T(z, Complete) $\wedge$ embodies(x, z)
BodyPartOrOrgan	$\forall$x: MacroStructure. x: CompletelyDevelopedBodyPart $\wedge$ T(x, OrganismLayer)
Tissue	$\forall$x: Tissue. x: BodyPart $\wedge$ T(x, Continuous)
BodySystem	$\forall$x: BodySystem $\exists$y: BiologicFunction. x: BiologicStructure $\wedge$ embodies(x, y)
BodyLocationOrRegion	$\forall$x: BodyRegion. x: BodyPart $\wedge$ T(x, Surface)
BodySpace	$\forall$x: BodySpace $\exists$y: BodySubstance. x: BodyPart $\wedge$ isFillableWith(x, y)
isConceptualPartOf	$\forall$x, y, z: System. $<_c(x,y) \leftrightarrow \neg(((x < y) \wedge (y < z)) \rightarrow (x < z))$
isPartOf	$\forall$x, y, z: Part. $<_p(x,y) \leftrightarrow (((x < y) \wedge (y < z)) \rightarrow (x < z))$

CORE model v.5g items	ON7 definitions of consistent mappings
PhysicalStructure	$\forall$x: Structure. x: Object $\wedge$ T(x, Atemporality)
DiscreteStructure	$\forall$x: DiscreteBiologicStructure. x: BiologicStructure $\wedge$ T(x, Discrete)
BodyStructure	$\forall$x: BiologicStructure. x: Structure $\wedge$ T(x, BiologicLayer)
BodyPart	$\forall$x: NormalBodyPart. x: BodyPart $\wedge$ T(x, Normal)
Tissue	$\forall$x: Tissue. x: BodyPart $\wedge$ T(x, Continuous)
BodyRegion	$\forall$x: BodyRegion. x: BodyPart $\wedge$ T(x, Surface)
SurfaceBodyPart	$\forall$x: DiscreteBodyRegion. x: BodyRegion $\wedge$ T(x, Discrete)
SurfaceOpening	$\forall$x: BodyOpening $\exists$y: (BodyPart $\vee$ Substance). x: BodyRegion $\wedge$ isFillableWith(x, y) $\wedge$ T(x, Surface)

SNOMED items	ON7 definitions of consistent mappings
Topography	$\forall$x: TopographicObject. x: BiologicStructure
System	$\forall$x: AnatomicalSystem $\exists$y: BiologicFunction $\exists$z: Organism. x: BodySystem $\wedge$ embodies(x, y) $\wedge$ embodies(z,y) $\wedge <_c(x, z)$
TopographicRegions	$\forall$x: BodyRegion. x: BodyPart $\wedge$ T(x, Surface)
CellularStructures	$\forall$x: CellularStructure $\exists$y: Organism. x: BiologicStructure(x) $\wedge$ Organism(y) $\wedge$ (x $<_p$ y) $\wedge$ T(x, CellularLayer)
IntegumentarySystem	$\forall$x: IntegumentarySystem $\exists$y: IntegumentaryFunction. x: BodySystem $\wedge$ T(x, Continuous) $\wedge$ embodies(x, z)

Table 3: Sample mappings of items from UMLS semantic network, CORE model 5g, and SNOMED III.

4. DISCUSSION

We have presented some topics concerning the difficulties of conceptual convergence among different medical terminological systems: health care operators accomplish convergence of different intended meanings through context and their operational knowledge. Though, computational integration needs a *formal descriptions* of *explicit intended meanings*.
The integration task has been accomplished by defining the specific ontologies of the 5 source systems (through

the ONIONS methodology) within the framework of ON7, which holds them together by referring to general ontology notions[18,21,23].

The comprehensive medical domain ontology of ON7 acts as a library (cf.[12]) of the more specific domain ontologies which form the base for each system.

The library paradigm is often connected to a claim of *modularity*. ON7 is potentially open to further modules — even if partially not compatible— provided that they can be analysed within a more general ontology framework (e.g., a module on "traditional Chinese" medical record cannot be integrated, because its underlying ontology is not compatible with our "western" general ontology). The modularity hypothesis is being tested through an ongoing experiment which analyzes the categorial structures[2] of concept systems produced by the European Standardization Body (CEN) on *drugs*, *surgical procedures*, *laboratory quantities*, and *medical devices* (in cooperation with CEN/TC251/PT015).

The analysis allowed us to explicitly formalize different viewpoints on important issues:

• diagnosis as *process* or *outcome of a process*. For example, SNOMED or ICD identify disease and diagnosis because they are oriented to statistics, records and reports, and a diagnosis is considered as a *diagnostic outcome*. UMLS is oriented towards scientific knowledge, and diagnosis is considered as an inferential process;

• morphology as *form* vs. *outcome of a function* vs. *structure in a given condition*, in particular when a structure is the outcome of a surgical procedure or a pathological process;

• regions, spaces, holes[23] as *conceptual arrangements of structures* (e.g. in UMLS) vs. *structures* themselves (*immaterial objects*).

4.1 The *body part* example

The idea of *body part* is different in the three terminological systems (*tab. 4*): *Body part* is not isolated in UMLS, but is together with *organs*, and implies full formation; instances are listed in MESH[16] (the bottom level). SNOMED *Topography* puts apart *body regions* and organizes the rest by *body system*; within each body system, it puts apart *body fluids* and the rest deals mainly with *body parts* (implicitly). CORE model 5g (having a terminological language with an engine) presents various hierarchical structures in which *body part* can be browsed. Nevertheless, model 5g had not yet an ontological foundation: it was powerful but ad hoc. Ad-hoc-ness hampers easy merging of different sources (and the production of new coherent extensions) and creates fuzziness when internal homogeneity of the model is the issue and when a general ontological comparison is made among different domain ontologies.

Notice that such an integrational work can be seen as the categorial framework for context translation as well. I could talk of a body part in the context of UMLS, or in the context of SNOMED, and then rearrange everything with a formalism such as *context logic* (see [14], which provides an example of database assumptions integration in the domain of engine parts warehouses).

But this brief outline should have shown that formal issues come *after* (or at least *accompanying*) the special ontological paradigms (domain ontologies) that have been extracted and generalized, and this can be done only through a comprehensive methodology of source analysis and hypothetico-deductive application of generalized paradigms (general ontologies).

The formal definition of *body part* in ON7 is shown in table 1.

body part of the UMLS semantic network corresponds to *completely developed organic body part* in ON7:

$\forall x$: CompletelyDevelopedBodyPart $\exists y$: Organism $\exists z$: Development.
x: BiologicStructure $\wedge <_p(x, y) \wedge T(z, \text{Complete}) \wedge$ embodies(x, z)

body part of the CORE model v.5g in GALEN corresponds to *normal body part* in ON7:

$\forall x$: NormalBodyPart. x: BodyPart $\wedge$ T(x, Normal)

moreover, we have to add a *systemic* interpretation to deal with the SNOMED organization; given the definitions of BodyPart and BodySystem in ON7 (table 1), it will hold that:

$\forall x$: AnatomicalSystem $\exists y$: BiologicFunction $\exists z$: Organism. x: BodySystem $\wedge$ embodies(x, y) $\wedge$
$\wedge$ embodies(z,y) $\wedge <_c(x, z)) \wedge \neg$(x: BodyRegion) $\wedge \neg$(x: BodyFluid)

Table 4: Different formal ontological definitions of the topic *body part* within a unique framework.

5. CONCLUSION

The issue of knowledge integration in medicine has been addressed by focusing on terminological knowledge implied in different sources; an ontological treatment of the non-convergence among different systems has been proposed. Our ontological treatment uses a methodology for the analysis of source conceptual paradigms (ONIONS) and the distinction between general and domain ontology. An outcome, the model ON7, is currently implemented in the GRAIL language for the GALEN project. The model is motivated by (and partially depends on) some influential general ontologies and by the relevant ontological choices defined from a set of influential terminological systems. Our procedures of

convergence treat —as far as this research is concerned— only the necessary and sufficient general and domain ontological criteria to integrate the specific ontologies analysed. We do not claim that it can treat the entire ideal knowledge. On the other hand, one could envisage an extensive use of the procedure to gradually produce a robust and comprehensive domain ontology of medical knowledge.
The next challenge could be to apply the ONIONS methodology to other, non-medical, domains.

Acknowledgements

The work has been partially funded by the EC project GALEN (AIM-2012) and by the Italian CNR special project SOLMC: 'Strumenti ontologici e linguistici per la modellazione concettuale' (Ontological and Linguistic Tools for Conceptual Modelling).

References

[1] Bernauer J. Modelling Formal Subsumption and Part-Whole Relation for Medical Concept Descriptions. In Workshop on Parts and Wholes: Conceptual Part-Whole Relations and Formal Mereology, ECAI '94, 69-79
[2] CEN/TC251/PT003. Model for representation of terminologies and coding systems in medicine. in: Proceedings of the Seminar "Opportunities for European and U.S. Cooperation in Standardization in Health Care Informatics", 1992
[3] Coté RA, Rothwell DJ, Brochu L (eds). SNOMED International, 3rd ed., 4 vols. Northfield, Ill: College of American Pathologists, 1994
[4] EPISTOL Core Group. Knowledge Processing for Decision Support in the Health Sector. in Barahona P & Christensen JP (eds.): Knowledge and Decisions in Health Telematics, IOS Press, 1994
[5] Evans DA, Cimino JJ, Huff SM, Bell DS for the CANON Group. Toward a Medical-Concept Representation Language. Journal of the American Medical Informatics Association 1994; 1:207-17
[6] Falasconi S Stefanelli M. A Library of Medical Ontologies. in Workshop on Comparison of Implemented Ontologies, ECAI 94
[7] Gabrieli E. A New Electronic Medical Nomenclature. Journal of Medical Systems 1989; 3:6
[8] GALEN Project. Documentation available from the main contractor Rector AL, Medical Informatics Group, Dept. Computer Science, Univ. Manchester, Manchester M13 9 PL, UK, 1992-1994
[9] Gangemi A, Steve G, Rossi Mori A. Cognitive Design for Sharing Medical Knowledge Models. in MEDINFO-95
[10] Gangemi A, Poli R, Steve G. General, Regional, and Domain Ontologies: An Outline. Roma, CNR-ITBM: Technical Report 95-05-01 (1995).
[11] Gruber T. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 1993; 5:188-220
[12] Guarino N. Formal Ontology, Conceptual Analysis and Knowledge Representation. In N Guarino & R Poli (eds.) Formal Ontology in Conceptual Analysis and Knowledge Representation. Dordrecht: Kluwer 1995
[13] Humphreys BL, Lindberg DA.The Unified Medical Language System Project. in Lun KC et al. (eds.) MEDINFO 92. Amsterdam: Elsevier Science Publishers, 1992
[14] McCarthy J, Buvac S. Formalizing Context. Stanford Un. Tech. Note STAN-CS-TN-94-13, 1994
[15] Musen M. Dimensions of Knowledge Sharing and Reuse. Computers and Biomedical Research 1992; 25:435-67
[16 National Library of Medicine. MeSH Medical Subject Headings. Bethesda Maryland: NLM (yearly)
[17] Neches R et al. Enabling Technology for Knowledge Sharing. AI Magazine 1991; fall:35-56
[18] Rector A. Compositional Models of Medical Concepts: Towards Re-usable Application-Independent Medical Terminologies. in Barahona P & Christensen JP (eds.). Knowledge and Decisions in Health Telematics, IOS Press 1994
[19] Rector A, Gangemi A, Galeazzi E, Glowinski A, Rossi Mori A. The GALEN CORE Model Schemata for Anatomy: Towards a Re-Usable Application-Independent Model of Medical Concepts. in Proceedings of 12th International Congress of the European Federation for Medical Informatics (MIE94), 1994
[20] Rossi Mori A, Gangemi A, Galanti M. The Coding Cage. in Proceedings of 11th International Congress of the European Federation for Medical Informatics (MIE93), Freund Publishing House 1993 466-71
[21] Sowa JF. Top-Level Ontological Categories. In N Guarino & R Poli (eds.) Formal Ontology in Conceptual Analysis and Knowledge Representation, Dordrecht: Kluwer 1995
[22] Steve G, Gangemi A. Modelling a Sharable Medical Concept System: Ontological Foundation in GALEN. in Artificial Intelligence in Medicine Europe, AIME95
[23] Varzi A, Casati R. Holes and Other Superficialities. Boston: MIT Press 1994
[24] WHO. International Classification of Diseases 10th revision. Geneva: WHO 1994
[25] Fankhauser P, Kracker M, Neuhold E. Semantic vs. Structural Resemblance of Classes. Special issue: Semantic Issues in Multidatabase Systems, SIGMOD RECORD, Vol. 20, No. 4, December 1991, pp. 59-63
[26] Schroeder SA, Krupp MA & Tierny LM. Current Medical Diagnosis and Treatment. Prentice Hall, 1988
[27] Sujansky W, Altman R. Bridging the Representational Heterogeneity of Clinical Databases. Stanford Un. Knowledge Systems Laboratory Report KSL-94-07.

INTERPRETING SPREAD SHEET DATA FOR HUMAN-AGENT INTERACTIONS

Syed S. Ali
Computer Science Dept.
Southwest Missouri State University
Springfield, MO 65804

Susan Haller
Computer Science and Engineering Dept.
University of Wisconsin – Parkside
Kenosha, WI 53141

1 INTRODUCTION

We are concerned with knowledge interchange and mediation for the human-agent interface. Our problem involves interpreting and possibly embellishing information that is extracted from an impoverished representation to produce natural language explanations for humans. Our eventual goal is to interact with human users using natural language to update the initial representation. The nature of our task has several features in common with the knowledge sharing initiative. Our first problem is to develop a suitable interlingua that can represent information from an impoverished representation and enrich it with the types of information that humans possess and express with natural language. We also need to define a protocol that determines the content of sentences and conversational turns. As a testbed, our impoverished information source is SC (a standard Unix-based spreadsheet program) spreadsheets. We are attempting to process and explain the complex content of SC spreadsheets in natural language using a knowledge representation and reasoning system, ANALOG, as an interlingua.

2 METHODS AND TOOLS

Following Campbell, [7], our approach to knowledge interchange between the spreadsheet and the user relies on a mediating agent. The agent will interpret SC text files into an interlingua that will support further domain-specific inferencing, and that will be rich enough to support natural language processing. For an interlingua, we are using ANALOG [4, 5, 2, 1, 3, 6]. ANALOG is a knowledge representation system that is derived from the SNePS/Cassie project, a project that had as a goal to build a rational cognitive agent [15].

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/329

ANALOG is a knowledge representation and reasoning system that is based on a semantic network paradigm. It is *semantic* in that it represents concepts in the mind of a cognitive agent who is capable of using language. It is a *network* because the representation is a graph composed of nodes and labeled directed arcs. These graphs are constrained in the form they may take in the following ways: 1) each node represents a unique concept; 2) each concept represented in the network is represented by a node; 3) each concept represented in the network is represented by a unique node; 4) arcs represent non-conceptual binary relations between nodes; 5) the knowledge represented about each concept is represented by the structure of the entire network connected to the node representing the concept [11]. Propositional information is represented by nodes and not arcs, so ANALOG is a propositional semantic network.

3 OBJECTIVES

For the testbed task, the subtasks that we are undertaking include

1. Collecting student queries (in English) about a spreadsheet that contains grade information and analyzing the queries for their syntactic and semantic structure
2. Specifying appropriate knowledge representation structures for spreadsheet domains
3. Implementing a system for automatically converting SC spreadsheets into these structures
4. Implementing a system for automatically embellishing spreadsheet information with domain-specific information

5. Extending a parsing and generation grammar to process student queries and realize information from the knowledge representation

6. Providing an easy-to-use Web-based interface to the student grade spreadsheets

7. Deploy the resulting system for evaluation

We will describe some of the implemented, or partially implemented components.

4 KNOWLEDGE TRANSFER

At present, we have implemented a system to convert SC spreadsheets into ANALOG representations automatically. In theory, this is an intractable problem since spreadsheet developers have the option of organizing their spreadsheets in any manner that they choose. Research on spreadsheets is limited, but it suggests that in practice, designers build their spreadsheets using predictable organizations. If the spreadsheet organization deviates from an identified standard one, the resulting spreadsheet is hard to understand and use [13, 10, 12, 8]. Limiting ourselves to standard organizations, we feel that the prospects for converting spreadsheets to an interlingua are good.

The internal representation of an SC spreadsheet is easy to process, and spreadsheets that follow one of the conventional organizations can be converted to knowledge representation structures such as in Figure 1. The network shown was built from a spreadsheet with a *horizontal organizing axis*. A horizontal organizing axis means that field labels are in one or more rows across the spreadsheet, and field values are in columns above or below the field label. Figure 2 is a segment taken from a spreadsheet with a horizontal organizing axis. Figure 1 is a representation in ANALOG of some of the spreadsheet information in Figure 2. We note that the horizontal organization of the spreadsheet is lost in the knowledge transfer. Deployment and testing will indicate whether or not this information loss is significant.

The knowledge representation must capture, in propositional form, what is implicit in the tabular format of the spreadsheet. Figure 1 shows two propositions about the cell that holds the value `30`. The node labeled `M21` represents the proposition that the cell has a property called `HW2` (Homework 2), and the value of that property is `30`. The node labeled `M5` represents the proposition that this cell is a `Max Points`.

The representation raises questions about how to augment the sparse information from the spreadsheet with

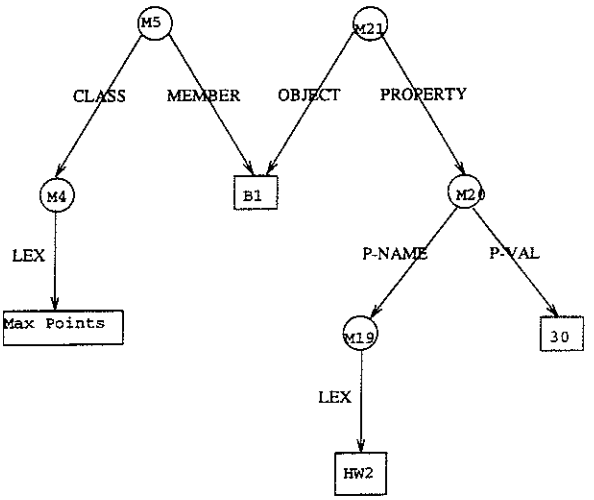


Figure 1: `B1` is a Cell Representing Maximum Points on `HW2` is `30`

CSCI 145		
Max Points	HW1	HW2
	15	30
==========	===	===
Name		

Allis, R.		

Figure 2: A Spreadsheet Representation for the Maximum Points on `HW2`

the kinds of domain-specific, static knowledge that humans use when they interpret spreadsheet information. The information that we must embellish our representation with is implicit in the labels used, the combination of labels used, and their juxtaposition. For example, in Figure 2, humans can infer from labels like `Max Points`, `HW1`, and `CSCI 145` that `15` is the maximum points achievable on the first homework assignment. While our current translation technique takes into account the relative position of labels and values, we have yet to consider how the actual strings that make up labels (like "`HW1`") can be used to relate the spreadsheet information to underlying domain-specific information that humans bring to the process of interpreting spreadsheets.

4.1 Conversion

As mentioned, the translation process from the internal SC spreadsheet file into ANALOG knowledge representation structures is automatic. There are two phases of the conversion. First, the knowledge representation

structures that represent positional (row and column) and type (data or label) information are built. Figure 3 shows the positional and type information for the cell that the value 30 occupies. Second, we use the positional and type information from the first phase to build new representations for cell dependencies like those shown in Figure 1. The final result is a collection of nodes in the network that represent all of the above kinds of information for each cell in the spreadsheet.

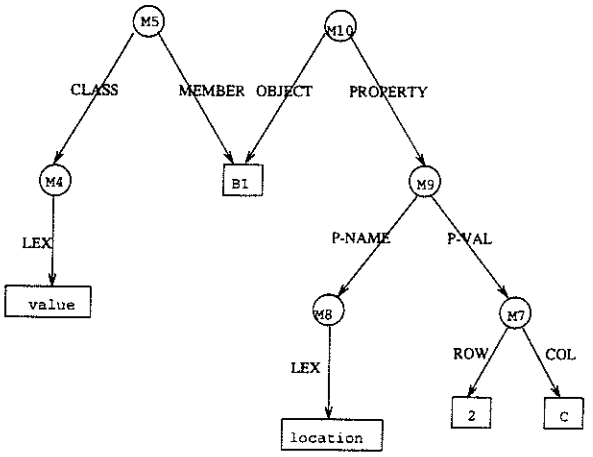


Figure 3: B1 is a Cell Value that is Located at Row 2 Column C

The conversion task is undertaken with the following simplifying assumptions:

- The spreadsheet is horizontally (or vertically) organized.
- Labels are created (in the spreadsheet) with the SC command that creates a label rather using a data value (such as a string).

The former assumption is necessary to do any processing, the latter to reasonably determine whether a cell is a data value or label. These assumptions are consistent with good spreadsheet style [9], and are not overly restrictive for the user. We are using our course grade spreadsheets for test data for the conversion process.

5 QUERIES

Based on the converted representation, we can make the following types of queries (and more) about the spreadsheet.

1. What is the location of a cell?
2. What are the properties of a cell (location, value)?
3. On what labels does a data cell depend?
4. What cells depend on another cell?
5. What values are in a row or column?
6. How is a cell value calculated?
7. In what ranges does a cell participate?

We feel that the ability to process such questions indicates that we have a good representation language for mediating between a spreadsheet agent and a user.

Currently, the questions must be asked in the ANALOG user language. Eventually, the above queries and their answers will be processed in English using an ATN grammar based on [14].

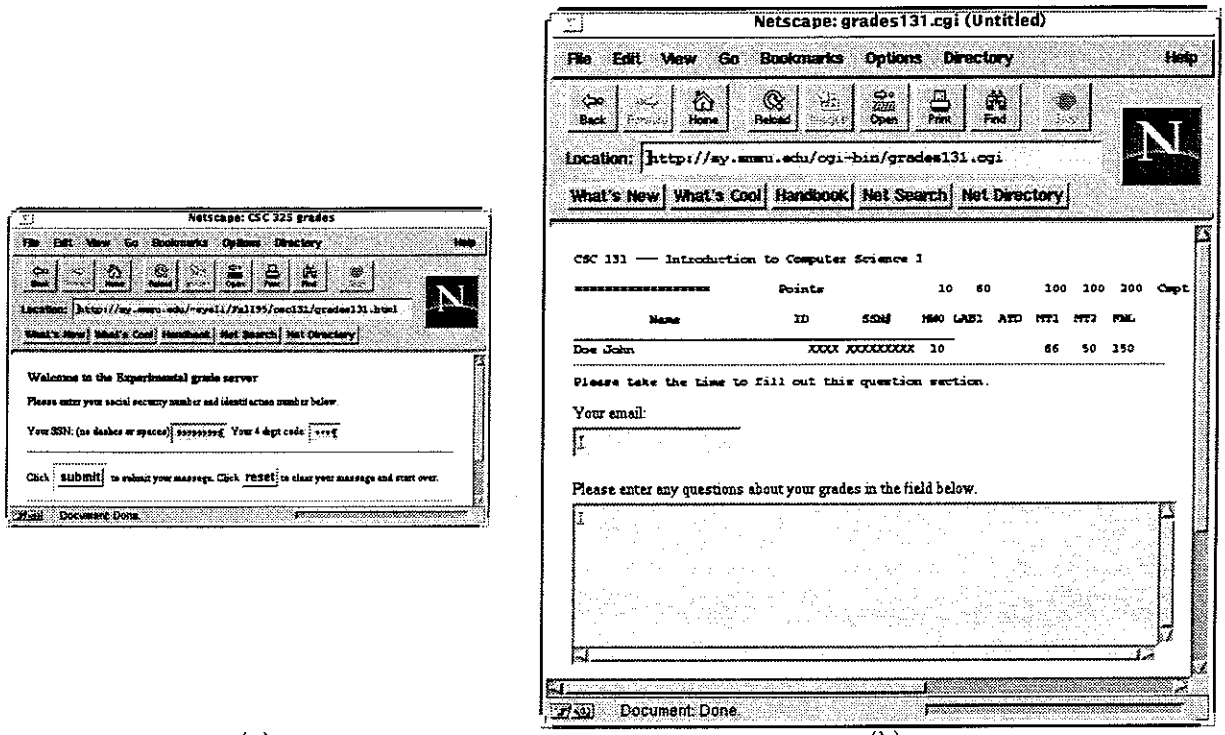
6 DEPLOYMENT

A set of world-wide-web tools (scripts, HTML forms, etc) has been developed to allow students easy viewing access to their grade spreadsheet. The Netscape interface is shown in Figure 4(a). While this initial version works, it is not very robust (it depends greatly on the correct user inputs, and rigid structure of the spreadsheet), and would need significant work. It does, however, represent a simple proof of concept for this part of the task. Students fill in the form of Figure 4(a) and get the form of Figure 4(b), where they can ask any questions in the space provided. Currently, their questions are answered by the authors and collected for later analysis. Students already use this HTML form-based interface and perceive it of value.

7 SUMMARY AND CURRENT STATUS

The collection of student queries is and will be an ongoing part of this task. However, it will require time to collect enough queries for the initial analysis. On the initial deployment of this interface fewer queries were made using this interface than the investigator had hoped. Students were unfamiliar with the interface, the web, etc, and preferred to stop by the office to ask questions. As students become more familiar with the interface and as it improves in usability, it is anticipated that the volume of queries will increase.

Analysis of their queries, their syntactic and semantic structure, will be ongoing in this subtask. We propose



(a) (b)

Figure 4: (a) Initial Netscape Query Form and (b) Grade Query Form

to try and classify these queries into categories (e. g., *how was this computation done*, *what is this component worth*, etc), and determine any regularities. Once this is done, natural language queries can be converted into queries in the ANALOG User Language (AUL) for answering.

Our initial work (ongoing since the Summer of 1995) suggests that the subtasks of designing a knowledge representation for spreadsheets, mapping SC spreadsheets into these representations, and processing queries about them, is tractable. The advantage of such a detailed knowledge representation is that more types of questions can be answered. Additionally, the resulting representations would capture what [12] have called the *deep structure* of the spreadsheet. This corresponds to the dependencies of cells in the spreadsheet, and is explicitly represented in the knowledge base.

References

[1] Syed S. Ali. A Propositional Semantic Network with Structured Variables for Natural Language Processing. In *Proceedings of the Sixth Australian Joint Conference on Artificial Intelligence*. World Scientific, NJ, November 17-19 1993.

[2] Syed S. Ali. A Structured Representation for Noun Phrases and Anaphora. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 197–202, Hillsdale, NJ, June 18-21 1993. Lawrence Erlbaum.

[3] Syed S. Ali. Node Subsumption in a Propositional Semantic Network with Structured Variables. In *Proceedings of the Sixth Australian Joint Conference on Artificial Intelligence*. World Scientific, NJ, November 17-19 1993.

[4] Syed S. Ali. A Logical Language for Natural Language Processing. In *Proceedings of the 10th Biennial Canadian Artificial Intelligence Conference*, pages 187–196, 1994.

[5] Syed S. Ali. *A "Natural Logic" for Natural Language Processing and Knowledge Representation*. PhD thesis, State University of New York at Buffalo, Computer Science, January 1994.

[6] Syed S. Ali and Stuart C. Shapiro. Natural Language Processing Using a Propositional Semantic Network with Structured Variables. *Minds and Machines*, 3(4), November 1993.

[7] A. E. Campbell, H. Chalupsky, and S. C. Shapiro. Ontological mediation: An analysis. manuscript, February 1995.

[8] Charles E. Collyer. Spreadsheet Modelling for Research and Teaching: Programming without Programming. *Behavior Research Methods, Instruments, & Computers*, 24(3):467–474, 1992.

[9] David Harrison and John W. Yu. *The Spreadsheet Style Manual.* Dow Jones-Irwin, Homewood, IL, 1990.

[10] D. G. Hendry and T. R. G. Green. Creating, Comprehending and Explaining Spreadsheets: A Cognitive Interpretation of What Discretionary Users Think of the Spreadsheet Model. *International Journal of Human-Computer Studies*, 40:1033–1065, 1994.

[11] A. S. Maida and S. C. Shapiro. Intensional concepts in propositional semantic networks. *Cognitive Science*, 6(4):291–330, 1982. Reprinted in R. J. Brachman and H. J. Levesque, eds. *Readings in Knowledge Representation*, Morgan Kaufmann, Los Altos, CA, 1985, 170–189.

[12] Pertti Sarriluoma and Jorma Sajaniemi. Extracting Implicit Tree Structures in Spreadsheet Calculation. *Ergonomics*, 34(8):1027–1046, 1991.

[13] Pertti Sarriluoma and Jorma Sajaniemi. Transforming Verbal Descriptions into Mathematical Formulas in Spreadsheet Calculation. *International Journal of Human-Computer Studies*, 41:915–948, 1994.

[14] S. C. Shapiro. Generalized augmented transition network grammars for generation from semantic networks. *American Association of Computational Linguistics*, 8, 1982.

[15] S. C. Shapiro. The CASSIE projects: An approach to natural language competence. In *Proceedings of the 4th Portugese Conference on Artificial Intelligence*, pages 362–380, Lisbon, Portugal, 1989. Springer-Verlag.

HEART — A MODEL OF EMOTION FOR NATURAL LANGUAGE INTERPRETATION. 3. MECHANISM FOR EMOTIONAL REVERBERATION, DECAY, AND PRIMING.

Mark S. Schmalz
mssz@cis.ufl.edu

Douglas D. Dankel II
ddd@cis.ufl.edu

Department of Computer and Information Science and Engineering
University of Florida, Gainesville, FL 32611

ABSTRACT

The modelling of affect (emotional state) has long been of interest in psychological research (e.g., behavior prediction in psychotherapy) as well as advertising and political science (e.g., induction of consumption and voting behaviors). Additionally, emotional models would be useful for law enforcement research, in analysis of kidnapping and hostage negotiation accounts [1,2]. Accordingly, this paper describes realistic extensions to our existing emotional model [3,4], such as mechanisms for emotional reverberation within and between agent and recipient or agent/recipient perception of an observer's emotions. The companion paper [5], overviews emotional modelling that supports the interpretation of natural language (NL) accounts of human abduction (kidnapping) experiences.

1. INTRODUCTION

We previously noted [3,4] that research in mechanistic models of human emotion is progressing slowly due to lack of tools and methods for investigating underlying physiological processes. We classified previous emotional modelling efforts as micro-, meso-, and macro- structural. Microstructural (neurophysiological and neuropharmacological) models are not considered here due to a lack of comprehensive, validated research data. Mesostructural models (e.g., stimulus-response modelling in primates) include nonverbal communication such as emotional responses to perceived facial expressions [6,7], and are attractive implementationally. Macrostructural models based on cognitive psychology (e.g., James' and Ortony's introspective research [8,20]) supported our efforts in emotional categorization, and highlighted applications in the induction of affect by mass communication [9] and law enforcement [21].

Continuity with our previous work in NL interpretation (NLI) [10,11] was achieved by focusing on the deduction of emotional content from NL texts. Dyer's development [5] of software to extract emotional topics from NL texts (e.g., newspaper articles) motivated our development of procedures for translating NL text into (a) representations and (b) perturbations of emotional state. We further developed an emotional model, called HEART (Hearing Emotion, Agent, and Recipient in Text), which describes an emotional state using a vector of emotional identities and degrees. Using HEART, we deduce emotional perturbation vectors from NL text that were combined mathematically with a given emotional state vector to yield a modified emotional state. HEART-II [4] elaborated this concept by incorporating agent's and recipient's assessments of each other's emotional state, to facilitate resolution of ambiguous NL.

Unfortunately, the prototype HEART-II model did not depict temporal effects such as emotional decay to an equilibrium state or reverberation and internal reinforcement of emotions [12]). Additionally, practical applications (e.g., analysis of hostage-negotiator situations and interpretation of abduction accounts) were limited by the absence of a sub-model for interactive emotional perception and transmission on the part of one or more observers. This is important in psychotherapy or law enforcement applications, where abduction accounts are frequently interpreted by a therapist or investigative analyst who may interact with one or more agents or recipients.

In realistic situations, emotional models alone are insufficient for determining emotional intent from NL text and must be augmented with detailed, accurate models of perception, cognition, and behavior. Since such processes generally remain obscure, we currently focus development on models that support emotion-based NL interpretation. Due to the lack of supporting research in belief systems (e.g., ontologies, paradigms, and perspectives that motivate human behavior) and their effects on the perception of emotion, we view emotional modelling research as a long-term effort that may not yield immediately useful results. However, emotional modelling research can be useful in supporting interpretation of psychotherapy session transcripts, ethnological accounts, and historical narrative.

This paper expands previous theory (Section 2.1) to include effects of emotional reverberation and decay (Section 2.2). The effect of observer-agent-recipient interaction is analyzed in Section 3, with comments on the role of belief systems in NLI of ambiguous text. Section 4 discusses implementational issues, with conclusions and future work summarized in Section 5. Salient applications (e.g., interpretation of abduction scenarios) are presented in Reference 5.

2. THEORY OF EMOTIONAL MODELLING

We begin Section 2.1 by recapitulating basic emotional modelling theory, then progress to enhancements incorporated in the HEART-III model (Section 2.2).

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/334 ©1996 FLAIRS

2.1. Basic Concepts.

2.1.1. Definition. An *emotion* is an atomic entity that influences behavior.

2.1.2. Definition. Per [4] and Definition 2.1.1, basic emotions derived from well-established philosophical, psychological, and theological models [13,14] are:

- **Ambition** — motivates or inhibits achievement;
- **Faith** — directs action based on abstractions;
- **Generosity** — motivates resource sharing;
- **Humility** — modulates self-awareness, contemplation, direction, and expression.
- **Joy** — fosters expression of happiness, sadness, etc.;
- **Love** — motivates actions of affection, kindness, etc.;
- **Peace** — modulates emotional variance in time and space; and
- **Tolerance** — predisposes one to serenity in the presence of new or unexpected stimuli.

2.1.3. Definition. Emotional *degree* (i.e., strength or intensity) ranges from -3 (extremely antagonistic) through zero (neutral) to 3 (extremely agonistic).

2.1.4. Example. From [3], *generosity* ranges from *avaricious* (-3) through *greedy* (-2), *covetous* (-1), *content* (0), *giving* (1), and *generous* (2), to *philanthropic* (3).

2.1.5. Definition. Let the eight emotional measures given in Definition 2.1.2 be grouped in set E, which we index by $\mathbb{Z}_8$. Let emotional degrees be grouped in $D = \mathbb{Z}_8$, which supports implementation of Definition 2.1.3 and a *don't care* value. The latter will be used in cases where the interpreter does not interact with various entities. We observe that D can be a continuous interval such as [0.8,1.5], which supports stochastic emotional modelling.

2.1.6. Definition. An entity's *emotional state* is denoted by $\mathbf{e} \in D^E \equiv E \rightarrow D$. In this paper, we use $\mathbf{e} \in D^8$ for brevity, since $|E| = 8$.

2.1.7. Definition. Denote an element of an indexed set of *perturbations* P by $p_i \in P$, where $i \in \mathbb{Z}_8$. (Similar to the definition of E, P may be a discrete or continuous interval.) If p_i corresponds to emotional degree $e_i \in E$, then emotional state can be modified by applying a *perturbation vector* $\mathbf{p} \in P^8$ to e as follows.

2.1.8. Definition. The *additive perturbation* of an emotional state $\mathbf{e} \in D^8$ by $\mathbf{p} \in P^8$ is given by:

$$\mathbf{e}(t+1)(i) = \begin{cases} \vee(\mathbf{e}(t)(i) + \mathbf{p}(i), -3) & \text{if } \mathbf{p}(i) \leq 0 \\ \wedge(\mathbf{e}(t)(i) + \mathbf{p}(i), +3) & \text{if } \mathbf{p}(i) > 0 \end{cases}, i \in \mathbb{Z}_8, \quad (1)$$

where $\vee$ and $\wedge$ denote maximum and minimum operations that limit the output of Equation 1 to the interval [-3,+3], per Definition 2.1.3.

2.2. HEART-III Emotional Model.

We expand our HEART-II model [4], which considered the emotional states of agent and recipient, as well as their assessment of each other's emotional states, by including in HEART–III the following entities:

1. $N_a \geq 1$ *agents*, which perturb the emotional state of a fellow agent, recipient, or interpreter (defined below);
2. $N_r \geq 1$ *recipients*, whose emotional state is perturbed by an agent, fellow recipient, or interpreter;
3. $N_x \geq 0$ *interpreters* (processes or persons), which infer agent, recipient, environment, emotional states, and perturbations from NL that describes agent-recipient interactions or dialogues;
4. For each agent, recipient, and interpreter, there exists a *reverberation vector* that contains the *decay rate*, *oscillation frequency*, and *limiting amplitude* of emotional states;
5. $M^2 - M$ *assessments of emotional state*, where $M = (N_a + N_r + N_x)$. An assessment is made by an agent, recipient, or interpreter concerning a fellow agent, recipient, or interpreter. For simplicity, we assume that self-assessment is reflected in one's emotional state;
6. $M^2 - M$ *assessments of decay and reverberation parameters*, similar to assessments of emotional state;
7. *Perturbation environment*: Context in which the agent-recipient interactions occur; and
8. *Interpretation environment*: Context of one or more interpretive processes.
9. A *transition function* that maps items 1–8 to a set of resultant emotional states.

2.2.1. Assumption. For simplicity, let one interpreter analyze dialogue between an agent and recipient, i.e., $N_a = N_r = N_x = 1$.

2.2.2. Definition. Let the j-th agent having state $\mathbf{e}_j \in E^8$, where the index $j \in \Lambda_a$, cause a perturbation $\mathbf{p}_{j,q}$, where q indexes a fellow-agent, recipient, or interpreter whose emotional states are denoted by $\mathbf{e}_k$, $k \in \Lambda_r$, and $\mathbf{e}_l$, $l \in \Lambda_x$. For convenience, let $\Lambda_a, \Lambda_r, \Lambda_x \subset \mathbb{N}$ be disjoint, where $\Lambda = \Lambda_a \cup \Lambda_r \cup \Lambda_x$. Thus, HEART's emotional state map is denoted by $\mathbf{e} \in (E^8)^{\Lambda}$.

2.2.3. Assumption. Let each entity's emotional state converge to an *equilibrium state* $\mathbf{q}_\lambda$, $\lambda \in \Lambda$. For brevity, let $\mathbf{q}$ be a prespecified initial state.

2.2.3. Definition. An *emotional decay constant* $\tau \in (\mathbb{R}^8)^{\Lambda}$ is specific to the i-th emotional state of the λ-th agent, recipient, or interpreter, where $\lambda \in \Lambda$ and $i \in \mathbb{Z}_8$. Implementationally, τ is expressed in inverse seconds and represents a decay of $1/e$ of the difference between the emotional level $\mathbf{e}_\lambda(i)$, and the equilibrium level $\mathbf{q}_\lambda(i)$, Convergence of $\mathbf{e}$ to its equilibrium state is expressed as:

$$\mathbf{e}(t + \Delta t) = \mathbf{q}(t) - \left[(\mathbf{e}(t) - \mathbf{q}(t)) \cdot e^{\tau \cdot \Delta t}\right], \quad (2)$$

where Δt denotes decay time elapsed since time t.

2.2.4. Remark. Since the form of τ has not been validated for human subjects under various experimental conditions, it is reasonable to assume exponential decay because: (1) numerous physical and physiological processes exhibit exponential decay [12,15] and emotions are primarily physiological processes; (2) we intuit that the convergence of emotional state toward an equilibrium level occurs at a faster rate when a given emotion is near an extremum of its degree range, versus slower convergence near an equilibrium level. The latter effect follows from general observations of homeostatic system convergence behavior [16].

2.2.5. Observation. Emotions tend to *reverberate* (i.e., oscillate about a given degree) in response to various stimuli [12]. For example, absence of information about an antagonist in a fear-laden situation can lead a recipient to confabulate details that help his emotional perception conform to self-perceived normality, which would be determined from world knowledge. Such details, especially if abstracted from a previous fearful situation, can be applied periodically to excite or inhibit existing levels of fear, thus yielding an oscillatory emotional degree. We conjecture that such emotional reverberation can be modelled as a sinusoidal oscillation.

2.2.6. Definition. Let $\nu \in (\mathbf{R}^8)^\Lambda$ denote the *emotional reverberation frequency*, expressed in inverse seconds, which represents the rate of oscillation between extrema of emotional degree defined by the corresponding *limiting amplitude* $\xi \in (\mathbf{R}^8)^\Lambda$. Given the emotional state $\mathbf{e}_\lambda$ of the λ-th entity, where $\lambda \in \Lambda$, let oscillations commence at time t and last for time Δt. The resultant emotional level $e(t + \Delta t)$ is given by:

$$e(t + \Delta t) = f\,[e(t) \cdot (1 + \xi \cdot \sin(\nu \cdot \Delta t))], \qquad (3)$$

where f limits oscillations of an emotional value to extrema of D (e.g., the interval [-3,3], per Equation 1).

Combining decay and oscillation effects yields a model of emotional transition:

$$\begin{aligned} \mathbf{e}(t + \Delta t) = f(\mathbf{q}(t) \\ -\left[(\mathbf{e}(t) \cdot (1 + \xi \sin(\nu \cdot \Delta t)) - \mathbf{q}(t)) \cdot e^{\tau \cdot \Delta t}\right] \end{aligned} \qquad (4)$$

that more realistically depicts reverberative decay.

2.2.7. Definition. There exist $M^2 - M$ *assessments* $\mathbf{a} \in \left(E^8 \times (\mathbf{R}^3)^8\right)^{\Lambda \times \Lambda}$ by entity α of the λ-th entity's emotional state $\mathbf{e}_\lambda$, decay constant τ, and reverberation frequency ν or amplitude ξ, such that:

$$\mathbf{a}_{\alpha,\lambda}(i) \approx (\mathbf{e}_\lambda(i), [\tau_\lambda(i), \nu_\lambda(i), \xi_\lambda(i)]), \ \alpha, \lambda \in \Lambda, \qquad (5)$$

where $i \in \mathbf{Z}_8$ denotes the index of an emotion in E.

2.2.8. Remark. The assessment of reverberation parameters may initially appear fanciful, but is required when modelling interactive interpretation (e.g., therapy sessions). For example, a competent therapist (interpreter) can anticipate an agent's or recipient's transient emotions and thus provide compensatory stimuli. In dialogue, this occurs by couching questions appropriately [5].

2.2.9. Definition. Let S_p and S_x denote a *perturbation environment* and *interpretation environment* that are combined synchronously over time $\Delta t \in \mathbf{R}^+$, together with emotional state $\mathbf{e}$, perturbation $\mathbf{p}$, and assessment $\mathbf{a}$ via a *transition function*:

$$\begin{aligned} h\colon \left((E \times \mathbf{R}^3)^8\right)^\Lambda \times \left((E \times \mathbf{R}^3)^8\right)^{\Lambda \times \Lambda} \times (P^8)^\Lambda \\ \times S_p \times S_x \times \mathbf{R}^+ \to (E^8)^M \end{aligned} \qquad (6)$$

to produce a new emotional state, where emotional states that have the value *don't care* are not affected by the transition function. If the states are adjusted asynchronously over time intervals Δt_λ, $\lambda \in \Lambda$, then the last member of *domain*(h) has the form $(\mathbf{R}^+)^\Lambda$.

2.2.10. Remark. We have shown [4] that a transition function can be programmed using functional predicates. For example, given the text *if a police officer arrests a citizen in public, then the citizen will be irritated*, from an emotional KB one can recall a *perturbation construct* such as:

```
(emot-perturb :current irritated
              :perturb (joy -1)
              :predict angry .
```

This would be linked with a *scene frame* similar to:

```
(scene-frame :agent police-officer
             :recipient citizen
             :environment public-place
             :action arrest)
```

as well as with reverberation parameters and state/parameter assessments to form an *emotion frame* that can be processed by the transition function (Equation 6).

3. ENTITY-INTERPRETER INTERACTION

The inclusion of assessments in the transition function domain allows us to model the following types of observer (interpreter) modulation of agent-recipient interaction.

3.1. Types of Interactions.

Beginning with the case of an unemotional automatic interpreter, we progress to the multiple human interpreter scenario.

3.1.1. Case 1: Unemotional *a posteriori* analysis. If an account of agent-recipient dialogue is analyzed a posteriori by a non-emotional automaton, then the interpreter's emotional state values would be set to the trivial value *don't care* (per Definition 2.1.5). The interpreter's assessment of others' emotional states would be nontrivial, since such assessments comprise an interpretation of agent-recipient dialogue.

3.1.2. Case 2: Emotional *a posteriori* Analysis. When an interpreter of index $\lambda \in \Lambda$ that is capable of emotion (i.e., has emotional states, receives a perturbation vector, and has a working transition function) analyzes a transcript of dialogue between agent $\alpha \in \Lambda$ and recipient $\rho \in \Lambda$, the assessments $\mathbf{a}_{\alpha,\lambda}$ and $\mathbf{a}_{\rho,\lambda}$ remain trivial (i.e., have a *don't care* value). This is easily verified by observing that the interpreter's emotions can affect his own understanding of the interpretation process, but cannot modify *a posteriori* the recorded agent-recipient dialogue.

3.1.3. Case 3: Emotional, Interactive Analysis. If an emotional interpreter analyzes dialogue elicited by the interpreter's stimulation of agent(s) or recipient(s), then $\mathbf{a}$ has nontrivial values. As a result of τ and ν being estimated in $\mathbf{a}$, emotional states of entities indexed in Λ may fluctuate. This case pertains especially to psychotherapy, where emotional interactions occur between the interpreter (therapist) and agent or recipient.

For example, let the α-th interpreter assess τ and ν of the λ-th recipient. Let an emotion e(i), $i \in D$, oscillate

per Equation 4, as shown in the dotted curve of Figure 1. Observation of such behavior by α furnishes information about ξ_λ. As shown in the solid curve of Figure 1, the interpreter can prime λ by applying a small negative perturbation $p_\lambda(i)$ to e at time t_B, where a large depression of emotional state is facilitated by the negative trend in $e_\lambda(i)(t)$. In contrast, if p was applied at t_A, e would tend toward the equilibrium level E, since p could cancel the momentum in e.

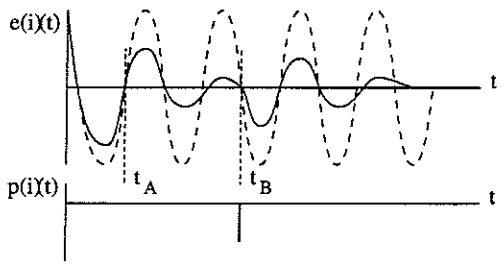


Figure 1. Example of priming α-th recipient by λ-th interpreter.

Such interactions can be strongly influenced by personal belief systems, as follows.

3.2. Role of Belief Systems.

Entities' beliefs, which give rise to expectations, can modulate their perception, modification, or actualization of emotional state [17]. In this study, we assume that modelled entities have similar belief systems that motivate accurate, unambiguous emotional assessments within a given situation. This restricted assumption reflects current lack of objective knowledge about the systematics and pragmatics of belief systems and their effect on emotion and cognition.

In response to this situation, we are developing requirements for a model of belief system structure and interaction that is designed to be compatible with HEART's structure, to further extend HEART's scope and flexibility. We initially model a belief system as an *ontology* (modelling concepts such as existence, causality, and relatedness). Upon the ontology are constructed *paradigms* (models of system structure and function). By observing events in the context of given paradigms, one obtains *perspectives* (individual views or projections) of reality as it is defined within the ontology. Unfortunately, little rigorous mathematical support exists for comparing qualitative measures such as the extent of a given relation between two entities. Although interval logics may be useful for such applications, we have the additional problem of expressing causality between subsets of entities in terms of a specific model of existence. Our research in belief system theory [18] is preliminary, and will thus be overviewed in future publications.

4. IMPLEMENTATIONAL ISSUES

We have elsewhere [3,4] discussed model requirements as well as structure and integration of HEART with a requirements gathering system called GATOR. We next discuss issues of transition function specification (Section 4.1) and asynchronous modelling (Section 4.2).

4.1. Transition Function Specification/Complexity.

HEART's transition function can be programmed using NL expressions that are translated to emotion frames [4]. We plan to enhance HEART-III's KB to account for multiple assessments of emotional state depicted in Equation 5, by extending HEART-II's KB structure to include multiple entities and assessments as well as an interpretation environment specification.

Given the quadratic complexity of the assessment matrix a, HEART-III's KB grows nonlinearly with the number of entities. In practice, a typical abduction scenario may have fewer than six agents, no more than two or three abductees (recipients), and a similar number of interpreters. This implies $M \leq 12$ entities, which yields $|domain(a)| = 4 \times 8 \times (M^2 - M) = 4{,}224$ values. This estimate accounts for each element of the assessment matrix being an eight-tuple of four values per tuple element (emotion, decay constant, reverberation frequency, and limiting amplitude). Using the eight-element value set discussed in Definition 2.1.2, we obtain 4,224 values $\times$ 3 bits/value = 12,672 bits or 1,584 bytes required for the assessment matrix a. Given the requirement of 40 values (states, parameters, and perturbation values) per each of the $M \leq 12$ entities, we have approximately 4,704 values (1,764 bits). When the environment specifiers are accounted for (approximately 128 bytes each), we have a transition function domain size of less than 2K bytes, which fully describes the model's instantaneous state under the aforementioned constraints. This small state description facilitates parallel implementation on processors with small local memories.

4.2. Asynchronous Implementation.

HEART's transition function (Equation 6) is amenable to asynchronous implementation, e.g., a MIMD processor network or neural net (NN). Asynchrony facilitates increased realism in simulating emotional interactions whose fluctuations may be difficult to predict accurately or may be chaotic. It is conceivable that HEART-III with 12 entities (reference Section 4.1) could be encoded in an NN of less than 5K inputs and a hidden layer of approximately 10K nodes to produce an output having $8M \leq 96$ output nodes. The output would be fed back to the input emotional states to obtain a closed-loop condition. Implementation of such a model in digital NN simulators or NN processing hardware is feasible.

5. PRELIMINARY CONCLUSIONS

We presented a mathematical model, called HEART (Hearing Emotion, Agent, and Recipient in Text), that is designed to support interpretation of emotion-laden NL. HEART accepts emotional state descriptions of agent(s), recipient(s), and dialogue interpreter(s) together with the agent's and recipient's perceptions of each other's emotional state, as well as a description of the dialogue or interpretation environments. Via a transition function that can be rule-based or implemented in a connectionist paradigm, HEART predicts entities' emotional states. Via assigned probabilities or values taken from a discrete interval, emotional degree is described in a manner that facilitates emotion prediction based upon continuous mathematics or interval logics [19]. In this paper, we present a graded, transitional model of emotion that is organized in terms of eight measures whose degrees range from extremely

antagonistic (-3) through neutral (0) to extremely agonistic (+3). A perturbation vector is applied to change the degree of a given emotional state.

HEART's transition function is being programmed using information derived from: a) the literature of emotional modelling, b) common-sense knowledge (including introspection), and c) domain-specific literature. HEART is designed to be used in conjunction with an NL-based requirements gathering system called GATOR [10,11], which has the capability of understanding a subset of technical NL. By considering emotional modelling as a separate applications domain, we are able to apply the requirements gathering technology developed for the GATOR system to the task of gathering information about emotions. From the internal model produced by our system, an emotional KB can be derived using the existing prototype, albeit with considerable effort due to the wide variety of environments and possible emotional behaviors that are typically encountered in NL.

6. REFERENCES

[1] Hacker, F.J. *Crusaders, Criminals, Crazies*, New York:W.W. Norton (1976).

[2] Simon, J.D. *The Terrorist Trap: America's Experience with Terrorism*, Bloomington, IN:Indiana University Press (1994).

[3] Schmalz, M.S. and D.D. Dankel. "HEART — A model of emotion for natural language interpretation. 1. Background and model requirements.", in *Proceedings FLAIRS '95 Conference on Knowledge-Based Systems*, (1995).

[4] Schmalz, M.S. and D.D. Dankel. "HEART — A model of emotion for natural language interpretation. 2. Model specification and integration.", in *Proceedings FLAIRS '95 Conference on Knowledge-Based Systems*, (1995).

[5] Schmalz, M.S. and D.D. Dankel. "HEART — A model of emotion for natural language interpretation. 4. Application to the analysis of textual accounts of human abductions.", in *Proceedings FLAIRS '96 Conference on Knowledge-Based Systems* (1996).

[6] Feldman, R.S. and B. Rime, Eds. *Fundamentals of Nonverbal Behavior*, New York: Cambridge University Press (1991).

[7] Pilowsky, I. and M. Katsikitis. "The classification of facial emotions: A computer-based taxonomic approach" *Journal of Affective Disorders* 30(1):61–71 (1994).

[8] James, William. *The Principles of Psychology*, New York: Holt (1890).

[9] Gaunt, P., Ed. *Beyond Agendas: New Directions in Communication Research*, Westport, Connecticut: Greenwood Press (1993).

[10] Schmalz, M.S., D.D. Dankel, K.S. Nielsen, and W.E. Walker. "An efficient, nonrecursive pipelined architecture for the automated understanding of technical texts", *Proceedings of FLAIRS '93, Ft. Lauderdale, FL, 18–21 April 1993*, pp. 228–232.

[11] Schmalz, M.S., D.D. Dankel, D.R. Rhodes, L.M. Muzzi, K.S. Nielsen, and W.E. Walker. "Knowledge-based linguistic support for automated requirements capturing", submitted to *FLAIRS '94 Conference on Knowledge-Based Systems, Destin, FL* (October, 1993).

[12] Levine, D.S. and S.J. Leven, Eds. *Motivation, Emotion, and Goal Direction in Neural Networks*, Hillsdale, NJ:Lawrence Erlbaum (1991).

[13] Simon, B. "Shame, stigma, and mental illness in ancient Greece", in *Stigma and Mental Illness*, P.J. Fink and A. Tasman, Eds., pp. 29–39; Washington, DC: American Psychiatric Press (1992).

[14] Nelson, J.R. "The role of religion", in *Prevention, Powerlessness, and Politics: Readings on Social Change*, G.W. Albee, J.M. Joffe, and L.A. Dusenbury, Eds., pp.421–432; Newbury Park, CA: Sage Publications (1988).

[15] Ogunnaike, B.A. *Process Dynamics, Modelling and Control*, New York:Oxford Univ. Press (1994).

[16] Bloom, F.E. *Neuroscience: From the Molecular to the Cognitive*, New York: Elsevier (1994).

[17] Zeig, J.K. and S.G. Gilligan. *Brief Therapy: Myths, Methods, and Metaphors*, New York: Brunner/Mazel (1990).

[18] Hendricks, J.E. and B. Byers, Eds. *Multicultural Perspectives in Criminal Justice and Criminology*, Springfield, IL: Charles Thomas (1994).

[19] Yen, J. and N. Pfluger. "A fuzzy logic based extension to Payton and Rosenblatt's command fusion method for mobile robot navigation", *IEEE Transactions on Systems, Man, and Cybernetics* **25**(6):971–978 (1995).

[20] Ortony, A. *The Cognitive Structure of Emotions*, New York: Cambridge University Press (1988).

[21] Miron, M.S. and A.P. Goldstein. *Hostage*, New York: Pergamon (1979).

ACQUIRING DOMAIN KNOWLEDGE THROUGH ANALYSIS OF TEXT FROM TECHNICAL DOCUMENTS

Soliman Edrees Howard Beck
Agricultural and Biological Engineering,
University of Florida,
P.O. Box 110570
soliman@agen.ufl.edu
beck@agen.ufl.edu

Ahmed Rafea
Computer Science Dep.,
ISSR, Cairo University,
Cairo, Egypt.
rafea@esic.eun.eg

Abstract

This paper presents a methods for acquiring, building, and augmenting a knowledge base. It uses a natural language system to acquire the knowledge from technical domain texts. The acquired knowledge is used to define new concepts added to the domain schema and build instances of domain schema. Machine learning techniques (concept formation and concept clustering) are used to define a new concept or a new cluster, or update and integrate with an old one. Citrus diseases technical documents have been used in the analysis process.

INTRODUCTION

One of the main problems in developing knowledge base systems is the acquiring of domain knowledge. Although several techniques have been invented to overcome this problem, it is still the bottleneck. That is because of the dependence on the domain experts to acquire knowledge. This paper introduces a study of automatic acquisition of domain knowledge from domain text. The process of automation depends on analyzing domain text, generating a conceptual model of the domain, extracting instances of the domain model, and building the knowledge base

A linguistic analysis of the domain text in the area of plant diseases of citrus has been done. The use of text documents reduces the human expert dependency in acquiring knowledge, and there are many useful relationships between entities expressed in the text which help in the process of analysis. The results of the analysis lead us to linguistic components (domain lexemes, phrasal grammar) and the domain model. The domain lexemes represent the domain vocabularies with their syntactic and semantic structure. The phrasal grammar represents the syntactic structure of the phrases in the domain. The domain model represents the entities of the domain and the relationships between these entities.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/339 ©1996 FLAIRS

Machine Learning techniques (Concept Formation, and Concept clustering) are used to identify new concepts, create a cluster, update an existing cluster, and finely incorporate these into an existing domain model. The next sections give an abstract for these topics.

RELATED WORK

Although there are several research attempts to build a knowledge base by processing natural language text such as **KUDZU [3]**, **PUNDIT [4]**, and **SCISOR [6]** they are different from this research. These systems process domain text and instantiate instances from domain schema. This research tries to organize the acquired knowledge by formulating, clustering concepts, and generalizing the entities which have common properties.

DOMAIN ANALYSIS

The results of technical document analysis have lead us to the preliminary lexicon, the phrasal grammar, and the domain model. Fig(1) shows a sample text of the domain.

The lexicon represents all the lexemes found in the available electronic technical documents. It is indexed by the lexeme entry. The structure of the lexicon consists of two types.[7]

The first one represents the noun and adjective lexemes. Each lexeme contains a concordance of phrases that have this lexeme. The concordances belonging to each lexeme are structured in clusters according to their semantic. Fig(2) shows the structure of lexeme *citrus* attached with its concordance.

The second one represents the verbs lexemes. It contains all the semantic components and their structure for each sentence that has this verb. Each verb has one or more semantic category which depends on the syntactic structure of the sentence. The structure of these relations are organized into groups according to their semantic category. Also, it contains more generalized syntactic information for each

sentence structure. Fig(3) shows the structure of the lexeme *develop* including its concordance and the structure of each relation in each semantic category.

BROWN ROT. (Phytophthora citrophthora, etc.).
Symptoms.

The rot starts as light brown discoloration of the rind and it remains firm unless invaded by secondary soft-rot organisms. Following periods of high humidity in the field or in storage, a white mold develops on the rotted fruit surface. Infected fruit soon drop so that the presence of this disease can be diagnosed by the numerous rotting fruit under individual trees. Disease outbreaks are sporadic and usually minor unless there are prolonged periods of rainfall. Disease is seen m ore often in the Indian River area than in the Central area of production.

Fig.(1) Sample of Domain Text

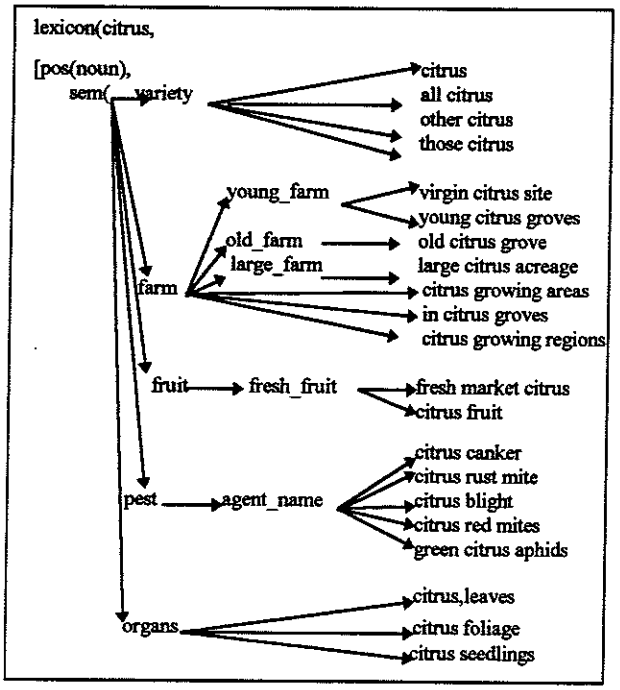
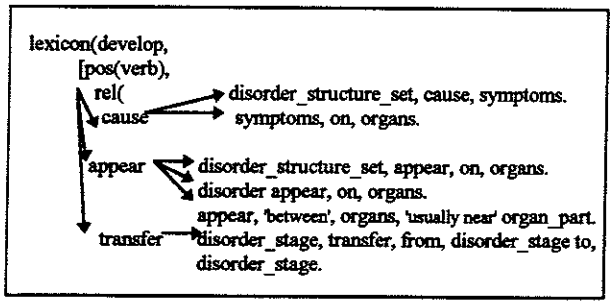


Fig.(2) lexicon structure for citrus



Fig(3) lexicon structure for verb develop

The phrasal grammar [2], [5] is extracted from the domain sentences. It is organized in a phrasal hierarchy. Semantic grammars are organized in the low levels of the hierarchy, and more generalized syntactic patterns are organized in the higher levels. This hierarchy structure permits many different phrase patterns containing the lexeme to be processed easily, and to get their semantic. The following set of rules Fig(4), illustrates the patterns of the phrasal grammar from the domain.

<S> --> <SUBJ> cause <OBJ> on <Host>.

<S>--> <SUBJ> cause <OBJ>.

<S> --> <OBJ> is caused by <SUBJ>
at <LOCATION>.

<SUBJ> --> <ORGANISM> | lack of <MINERAL>
in soil | <INSECT_STAGE> |<CULTURAL_CONTROL>.

<OBJ> --> <DISEASE> | <DEFICIENCY> | <DAMAGE>
<SYMPTOMS>.

<DISEASE> --> 'greasy spot' | xyloporosis | cachexia | 'citrus red mites'....

<DAMAGE> --> 'cracking and shredding of the bark'..

<DEFICIENCY> --> 'cooper deficiency'|'iron deficiency'..

<CULTURAL_CONTROL> --> prune...

<MINERAL> --> magnesium | iron | .....

Fig.(4) Sample of phrasal grammar from domain

The domain model represents all the entities, concepts, and the relationship between concepts, and entities in the domain.

The entities represent the semantic components or the properties of the domain. Each entity has its own instance values. The following are examples of domain entities.

<organs> is an entity, having the following instance values: leaves, stems, fruit, root, flower, bark, leaf, trunk, branch, twig, etc.

<organ_part> is an entity having the following instance values: lower surface of leaf, inside part of bark, the core of fruit, the rind of fruit, oil gland, etc. This entity is integrated with <organs> to formulate a position concept.

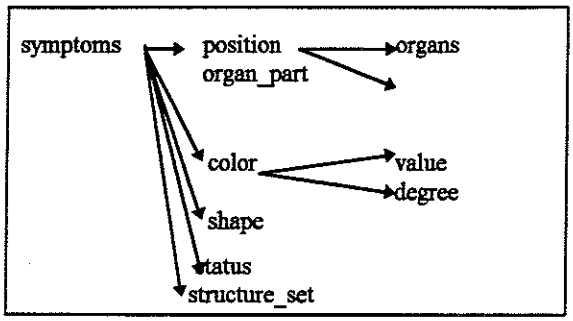
<color> is an entity that represents all color instance values found in the domain, such as yellow, green, brown, black, yellowish, creamy, tan, red, olive, etc.

The entities are aggregated together to formulate a concept. The aggregation process depends on the relationships between entities. Any aggregated concept has at least two entities.

The <organ> is affected by the <agent> infection causing the <agent_symptoms> which appear on <organ_part>. This infection may change organ color, or shape, to abnormal values.

<symptoms> is a concept which reflects the deviation of organs from normal to abnormal situation, such [yellow leaves], [small fruits], [white spots on lower surface of leaf]. The symptoms concept is aggregated from <organs>,

<organ_part>, <color>, <degree>, <shape>, and <symptom_status> entities. Fig.(5) illustrates the <symptoms> concept with its aggregated entities.



Fig(5) symptoms concepts

<disorder> concept is formulated from entities < name>, <status>, <period>, <variety>, < cause>, <symptoms>.

The < disorder > concepts are aggregated using the domain relations. The following are an examples of domain relation.

<cause> is a relation between <disorder> and <symptoms> or <disorder_set> and <disorder>.

<appear> is a relation between <symptoms> and <organs> or <variety>.

ACQUIRING DOMAIN KNOWLEDGE

Processing Technical Documents

Facts and relations concerning a disease agent are extracted from the text. These relations are organized and restructured into facts and rules. Examples of these relations for brown rot agent are shown below. The acquired relations include several types of knowledge. The first one includes a set of facts about the symptoms caused by the agent. The second is a set of facts about the agent causes (i.e. environment or organisms which activates the agent). The third is a set of fact about the varieties (hosts) affected by the agent. Finally is a set of facts about the location where the agent is found.

Formatting New Concept

Concept formation **[8], [10]** defines a new concept for each new acquired agent. The extracted set of facts and relations, Fig(6), are used to form a new concept for the citrus disease *brown rot*. First the concept builder extracts the properties related to each type of knowledge (symptoms knowledge, agent causes knowledge, preferable hosts knowledge, and the location knowledge). For each set of properties types, it tries to match these properties with those attached to the objects in the domain schema. If the matching process find an existing concept, it is retrieved and instantiated with related acquired knowledge. When the matching process fail, a new concept is defined and instantiated with this type of knowledge. After that it forms the concept of the agent according to the instantiated sub-concepts and their relationships.

For example, to form the concepts of brown rot agent, symptoms concepts will be aggregated from relations in Fig.(6). For example *"cause"* includes the *organ_part, color, and shape properties*, *"appear"* includes the *agent_structure_set, and organ_part*. Notice that these forms of symptoms concepts are subsets of the symptoms concepts in the domain schema Fig(5). The new formulated concepts for each agent are instantiated with their values found in the set of sub-concepts. Fig(7) shows the brown rot concept formulated from the facts and rules found in Fig.(6).

```
sem(agent([brown,rot]) ).

sem(agent([brown,rot]), rel(cause), color(degree([light],
value([brown])), shape([discoloration]), organ_part([the,rind])).

sem(agent([brown,rot]), rel(remain), agent_status([firm]),
rel(spread), degree([secondary]), agent_set([soft,rot,organisms])).

sem(agent_structure_set([white mold],rel(appear),
organ_part([rotted, fruit,surface]), farm([field,storage]),
climate_factor([high,humidity])).

sem(organ([infected,fruit],status(drop)).

sem(agent([brown,rot]),rel(appear),location ([indian,river,area])).
```

Fig.(6) A set of Facts and Relations for Brown Rot agent.

Clustering Concepts

The set of acquired concepts from the previous section is used in a clustering technique to form new clusters **[9], [1]**. First it finds the concepts which have shared properties. For each property the value is generalized if possible, creating a cluster which has all the concepts sharing this property value. After that it names the cluster with the property value. An example of a cluster is a group of agents which appear in a certain location, attack certain variety, cause certain symptoms, or affect certain organs, etc.

INCORPORATION WITH EXISTING KNOWLEDGE BASE

The acquired knowledge for a set of disorder is augmented to previously acquired knowledge base. The integration is divided into two levels. The first one is the domain schema level. In this level the new acquired object is added to the existing domain schema.

The second one is the instantiated knowledge base (instance of concepts and new clusters). The new instantiated knowledge may be integrated with the previously acquired knowledge, integrated after modification, or rejected. The rejected knowledge is already existing in the knowledge base. The modified knowledge is done according to the generality and the specificity of the existing knowledge comparable with the new knowledge. For example if the new knowledge is more general than the old, and adding it to the knowledge will contradict with other concepts, it keeps the knowledge which doesn't make any contradiction.

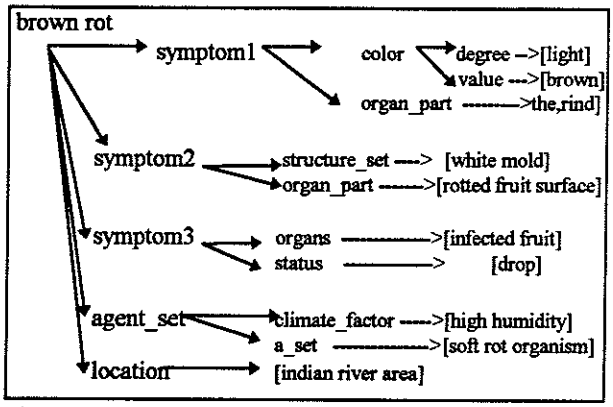


Fig.(7) brown rot instanitated concepts

A match technique is applied to get the similarities and differences between the new knowledge and the old. This technique attempts to minimize these differences by generalizing the properties of instances which are a member of the same class. So a modification of a knowledge base description is done by adding the generality of the differences. The ungeneralized differences which may represent a new property related to a new concept, are added into the appropriate part in the knowledge base (instantiated concepts or clusters knowledge base). The match techniques keeps the knowledge base consistent. For example if we get this new fact "symptom(color value 'yellow' status 'spot', organ_part 'upper surface')" for certain disease, and the existing knowledge base has already this fact "symptom(color degree'pale' value 'yellow' status 'spot', organ_part 'upper surface') " for the same disease, the old fact will be updated to "symptom(color value 'yellow' status 'spot', organ_part 'upper surface')" that is because the yellow value is more general than plae yellow value. In this case the system keeps the ugenerlized value "pale yellow" in the <color> cluster.

IMPLEMENTATION

The system is being implemented on a PC compatible machine using the SICSTUS Prolog language. The developed system has several modules, **INTERPRETER, LEXICON, PRE-PARSER, PARSER, INSTANCES, CON_FOM, CLUSTER, AND KA_INTEG.**

INTERPRETER is the main program in the system. Its function is to coordinate and communicate between the other programs. It reads input from the technical domain text, and passes it to the **PRE-PARSER** module. It passes the output of **PRE-PARSER** to the **PARSER** and gets a parse tree for the input line, activates the instances module to produce **INSTANCES** of the domain schema, activates the concept formation to produce a set of concepts for each disorder structured in form of facts and rules. After that it calls **CLUSTER** to create new cluster(s) and then call **KA_INTEG** which integrates the acquired knowledge in the knowledge base.

LEXICON contains all the lexemes found in the analyzed technical domain texts. Each is implemented as an object indexed by lexeme and which has the concordance. Fig.(8) shows a Prolog implementation for the citrus lexeme shown in Fig(2).

```
lexicon(citrus,variety,
        [[disorder,[[the,citrus,tristeza,virus],
                    [the,citrus,red,mite],
                    [citrus,blight]]],
                    :
        [organ_part, [citrus,macrophylla]]
        ]).
```

Fig(8) Prolog implementation of lexeme citrus

PRE-PARSER divides the input text into a set of phrases. First, it tags the punctuation to get sentence phrase(s) having a verb. These are divided into the main verb and subject and the object of the verb. Continuing the parsing process, the subject and object are divided into sub-phrases according to the grammar rules.

For example the sentence *"The rot starts as light brown discoloration of the rind and it remains firm unless invaded by secondary soft-rot organisms."* from Fig.(1) will be divided into three phrases *"The rot starts as light brown discoloration of the rind"* ,*"it "remains firm"* and, *"invaded by secondary soft rot organisms"*

The first phrase has subject and object, the subject has one phrase *"The rot"* and the object has two phrases *"light brown discoloration"* and *"the rind"*. These phrases are used by parser to get the semantics.

```
parse_rel(start, rel(appear), Sbj, SSbj, Obj, SObj):-
                    sem_of_con(set, Sbj, Sem),
                    get_con([of], Obj, O1, O2),
                    sem_of_con(symptoms, O1, Sem1),
                    sem_of_con(organ_part, O2, Sem2),
                    SSbj=..[sem,Sem],
                    SObj=..[sem,Sem1,Sem2].
```

Fig.(9) Sample of Prolog implementation of the parser

PARSER produces the semantic interpretation for each input phrase. It uses a set of parsing rules Fig(9). It communicates with the **PRE-PARSER** module to obtain sub-phrases. Also,

it communicates with the **LEXICON** to get the semantic of lexemes. For Example to parse "The rot starts as light brown discoloration of the rind", it activates the parse rule in Fig(9) and produces the semantic interpretation *"sem(set(rot), rel(appear), symptoms([light,brown, discoloration]), organ_part([rind]))"*

INSTANCES uses the semantic interpretation produced from the parser and builds instances of domain schema (set of facts and relations). An example of the instantiated instances of a domain schema is shown in Fig.(6).

CON_FORM consists of two modules. The first one generates the set of sub-concepts from the set of agents facts and rules Fig(6). The output of this module is a set of Prolog facts for the agent sub-concepts. The second module generates the agent concepts from the previous set of sub-concepts. One agent may have more than one sub-concept defining it. Fig.(10) shows a Prolog output of brown rot concepts.

```
agent([brown,rot]):-
        symptoms(color(V,Degree, organs(O)).
agent([brown,rot]):-
        symptoms(organs(O), status(St))
agent([brown,rot]):-
        symptoms(struct_set(SS), organ_part(OP)).
agent([brown,rot]):-
        symptoms(a_set(S), climat_factor(CF)).
color(V, Degree):-
        color_value(brown),
        degree(light).
        :
climat_factor([high,humidity]).
```

Fig(9) Prolog output for brown rot concepts.

CLUSTER consists of two modules. The first module finds the shared sub-concepts. A shared sub-concept (class) is one which is shared with other concepts and its value is the same or similar (i.e. light brown and brown are similar). The other module integrates all the concepts under the specified class. For example the following is a cluster for the concepts which shared with the same climate_factor values.

"class(climate_factor(humidity, degree(high), [[brown,rot], [algal,spot], [greay,spot]])."

KA_INTEG integrates the acquired knowledge (objects, concepts and clusters) with the previously acquired knowledge. It consists of a duplication checking module which is applied first to check if the new knowledge already exists or not. The consistency checking module tries to keep the knowledge base consistent.

CONCLUSION

This research introduced a approach which uses an NLP system and Machine Learning system to acquire a knowledge base. The system is implemented and tested on the analyzed text. The lexicon has about 800 lexemes implemented as shown in Fig.(8), and the parser has about 300 rules implemented as shown in Fig.(9). The output from processing this system is a knowledge base in form of Prolog facts and rules. The CLUSTER and KA_INTEG modules are not completely implemented or verified. The next step is to run this system on another technical domain text and also, to investigate the output syntax (i.e. a Prolog Form, or Expert Systems shell form).

References

[1] H. Beck, T.Anwer, and S. Navathe, A Conceptual Clustering Algorithm for Database Schema Design. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 6, No. 3, page No. 396-411, June 1994.

[2] M. Brent, From Grammar to Lexicon: Unsupervised Learning of Lexical Syntax. *Computational Linguistics* Vol. 19, No. 2, page No. 243-262, 1993.

[3] L. Boggess, J. Hodges, and J. Cordova, Automated Knowledge Derivation: Domain Independent Techniques for Domain-Restricted Text Sources. *International Journal of Intelligent Systems*, Vol. 10, page No. 871-893, 1995.

[4] L. Hirschman, M. Palmer, J. Dowding, D. Dahl, M. Linebarger, R. Passonneau, F. Lang., C. Ball, and C. Weir , The PUNDIT Natural Language Processing System. *Proceedings of the Annual AI Systems in Government Conference*, page No. 57-66, 1989.

[5] J. Hodges, and J. Cordova "Automatically Building a Knowledge Base Through Natural Language Text Analysis. International Journal of Intelligent Systems, Vol. 8, page No. 921-938,1993.

[6] P. Jacobs and L. Rau, SCISOR: Extracting Information from On-line News. *Communications of the ACM*, Vol. 33, No.11, page No. 88-97, 1990.

[7] P. Jacobs and L. Rau, Innovations in Text Interpretation. *Artificial Intelligence* Vol. 63, page No. 143-191, 1993.

[8] J. Martin, Acquiring and Combining Overlapping Concepts. *Machine Learning*, Vol. 16, page No. 121-155, 1994.

[9] A. Nevins, A Branch and Bound Incremental Conceptual Cluster. *Machine Learning*, Vol., 18, page No. 5-22, 1995.

[10] S. Wrobel, Concept Formation During Interactive Theory Revision. *Machine Learning*, Vol., 14, page No. 169-191 1994.

AN ANALYSIS OF THE HOPFIELD MEMORY FOR STORAGE OF NATURAL LANGUAGE SENTENCES[†]

Nigel Collier
Department of Language Engineering
UMIST
P.O.Box 88, Manchester M60 1QD, United Kingdom
E-mail: nigelc@ccl.umist.ac.uk

Abstract

This paper looks at how the Hopfield neural network can be used to store and recall patterns constructed from natural language sentences. Using linguistic arguments and numerical simulations we explore the differences between these and randomly generated patterns, used in much of the previous research. We also look at the time and space complexity of the algorithm used. Our results show that the storage ability of the network is better than expected and conclude that this is due to correlations among the patterns, which reduce the amount of information which needs to be stored.

1 INTRODUCTION

We propose here to explore the functionality of the discrete Hopfield neural network [7] when trained with natural language sentences. In particular we are looking at the accuracy of the model for storage and recall of sentences. Results are also given which show the time and space complexity of the trained model.

Suggestions have been made (e.g [4] [8]) for using connectionist architectures in natural language processing (NLP). Connectionist NLP however has not been fully exploited because of a lack of transparency in the models. These models often seem to be constructed *ad-hoc* on the basis of heuristics gathered from analogies to the human brain. Despite this, connectionism can offer the advantages of a bottom-up paradigm together with noise tolerance and an ability to generalise, i.e. to go beyond the linear relationships in the training data.

The Hopfield model is mathematically analysable and its behaviour can be elegantly explained (see [7]) using an energy function which converges when a stable state, corresponding to a result, has been reached. Recently much work has been done in theoretical physics (e.g. [2] [5] [9]) after it was found that the model could be used to represent a spin glass system. Analysis and numerical simulations in this area have extended our knowledge of the model far beyond what was known ten years ago. It is unfortunate that much of this work gives us no insight into how the model will behave when trained on natural language sentences. This is due to the assumption by physicists that the training patterns will be randomly generated.

In this paper we examine the storage capacity through auto-association simulations. We argue from a linguistic perspective that the properties of natural language patterns are significantly different to randomly generated ones to justify new investigations into the properties of the model. Particularly those of sparse and biased networks.

As an end in itself, this work contributes towards the task of image completion given noisy input. The task is well known in vision processing but also has applications in NLP. An example is sentence recognition in spoken language processing, where time

[†]While any mistakes in content are entirely my own I would like to thank my supervisor Professor Tsujii for his guidance and Professor Matsumoto of NAIST in Japan for his help in compiling the Asahi corpus. I would like to express my gratitude to Asahi Shinbun of Japan for their kind permission to use the editorial corpus.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/344

is a critical factor. Given the output of a spoken language recogniser we must decide which of the word combinations is most likely given contextual association knowledge.

2 MOTIVATION

One of the advantages of the Hopfield memory is its scaleability. This is important in NLP work because the size of the models are often proportional to the number of words in the training data. Even in sublanguage models the number of words in a domain can easily reach 10,000.

Several important assumptions have been made about the patterns being stored:

- The patterns consist of strings of randomly generated bits.
- Each bit has the same probability of being on or off.
- Patterns have no internal correlation - i.e. bits in one part of the pattern do not effect the probability of bits in another part of the same pattern.
- Patterns have no external correlation - i.e. bits in one pattern do not effect the probability of bits in a later pattern.

These assumptions have arisen from the analogy between the discrete Hopfield network and the Ising model and allow the behaviour of the Hopfield model to be constrained and simplified. In training the network to recall natural language sentences none of these assumptions hold true. We must therefore re-examine the behaviour of the network.

If we train the network on consecutive sentences taken from a text corpus then we see that:

- The patterns are not random, the distribution of words obeys that of the sublanguage in the corpus and is influenced by the underlying grammar of the English language.
- Words occur with different probabilities.
- The likelihood of a word occurring in a sentence is effected by the other words in that sentence (syntactic and semantic influences), therefore the patterns are internally correlated.
- The likelihood of a word (or phrase) occurring in a sentence is effected by the words in other sentences (pragmatic influence), therefore the patterns are externally correlated.

3 THE MODEL

The discrete Hopfield model is a feedback network where units are connected to each other by variable analogue weights held in a matrix W. The network is fully connected and symmetrical, so that $W_{ij} = W_{ji}$. No self interaction is permitted between a unit and itself, so $W_{ii} = 0$.

Given N units, each unit has inputs corresponding to

$$H_i = \sum_{j, j \neq i}^{N} W_{ij} V_j + I_i \tag{1}$$

where I_i is an external input to unit i. V_i is the output from unit i,

$$V_i = \begin{cases} 0 & if \quad H_i < U_i \\ 1 & if \quad H_i \geq U_i \end{cases} \tag{2}$$

where U_i is a threshold value.

Once trained, the stored patterns can be recalled by presenting the network with a noisy version of a training pattern and updating units randomly and aysnchronously until the network converges to a stable state. This is shown by the convergence of an energy function,

$$E(\{V\}) = -\frac{1}{2} \sum_{i}^{N} \sum_{j}^{N} W_{ij} V_i V_j + \sum_{i}^{N} U_i V_i - \sum_{i}^{N} I_i V_i \tag{3}$$

Previous theoretical work has linked the storage capacity of the Hopfield network to a storage ratio

$$\alpha = \frac{n}{N} \tag{4}$$

in which n is the number of patterns to be stored.

Hopfield [7] studied the storage capacity of the network using simulations and found that auto-associative recall began to degrade when α exceeded a critical value α_c, where $0.1 \leq \alpha_c \leq 0.2$. Analytical approaches such as that used by Amit *et al* [1] have given us a value of $\alpha_c \approx 0.14$ at which recall degrades discontinuously.

Further studies by Grensing *et al* [6] have shown that if we accept a small amount of error in recall, say 0.005% then $\alpha_c \approx 0.15$.

The critical storage ratio limits the number of patterns which we can store to

$$n \leq \alpha_c N \tag{5}$$

This limitation in storage capacity has been one of the biggest draw-backs for using the Hopfield network.

One consideration which is not shown in the storage ratio is the correlation between patterns. The assumption was made that because the patterns were randomly generated they were orthogonal and required separate stable points in the energy landscape.

If patterns are correlated, i.e. overlap, then there is less information to store. We need only store the part of the pattern which discriminates it from its closest neighbour. This consideration has not received much consideration in theoretical work with randomly generated patterns, but is highly relevant to natural language patterns. The implication is that we may be able to store many more 'correlated' patterns than the theoretical limit indicates for uncorrelated ones.

4 TRAINING

One way of storing sentences is to let the words be the basic units. We ignore the relevance of word ordering and represent the context of the sentence through its component lexical items. A sentence then becomes a vector ξ_i^μ where $\mu \in [1, n]$, and $i \in [1, N]$. n is the total number of sentences we want to store and N is the size of the lexicon. $\xi_i^\mu = 1$ when a particular word i is in sentence μ, otherwise $\xi_i^\mu = 0$.

The knowledge structure for the network is a matrix of weights W

$$W_{ij} = \frac{1}{N} \sum_{\mu=1}^{n} \xi_i^\mu \xi_j^\mu \tag{6}$$

between words i and j.

This training prescription produces very sparse networks which by themselves cannot properly inhibit as well as excite units. We introduce a small negative constant for cases of zero weights which is $-N^{-1}$. This small negative weight seems to provide enough inhibition to recall patterns.

The training text we use is the Asahi corpus (see [3]) of newspaper editorials. Grammatical words are removed as having no contribution to context. The sentences have an average of 11 content words.

The subject matter in the Asahi corpus is mainly international news stories. We may consider that the corpus contains a sublanguage of international and domestic events because the range of stories is quite narrow. For example, trade issues between the US and Japan and the Japanese budget are frequent topics. The corpus lexicon growth curve reflects this with closure occurring at around 16000 sentences.

We have constructed several training matrices $W1$ to $W4$ from the Asahi corpus. Each smaller matrix contains a subset of sentences from larger matrices. Statistics for each matrix are given in Table 1. Values for α and bias are also given, where bias is

$$bias(\xi) \equiv P(\xi_i^\mu = 1) = \frac{\sum_\mu^n \sum_i^N \xi_i^\mu}{n \times N} \tag{7}$$

	W1	W2	W3	W4
N	260	673	921	1357
n	41	121	167	273
α	0.16	0.18	0.18	0.20
$b(\xi)$	0.037	0.014	0.011	0.008

Table 1: Matrix characteristics

The maximum number of units we are using here is 1357 from the first 273 sentences in the corpus. Although there are plans to conduct much larger experiments, which better reflect the scale of the NLP problem, this will need improved software and a faster computer. The programs here are built in C and run on a Sun Sparc station 20.

5 RESULTS

5.1 Pattern Completion

We conducted auto-association tests to find the mean fraction of bits, $\overline{F}_B$, in stored patterns which were in error, where

$$F_B = \frac{1}{N}\sum_{i}^{N} |\tilde{V}_i^{(\mu)} - V_i^{(\mu)}| \qquad (8)$$

$\tilde{V}^{(\mu)}$ is the stable state achieved from presenting the network with a noisy version of training pattern ξ^{μ}. $V^{(\mu)}$ is the stable state we want - the *nominated* state.

To ensure statistical accuracy $\overline{F}_B$ has been calculated for 50 sentences in 10 trials and is therefore the mean value of 500 simulations. Initial network states are subjected to increasing noise levels, m_0, from 0.0 to 1.0 in increments of 0.1.

We would expect from previous discussions (e.g. [1] [2]) that recall would not be possible with such a sparse matrix and with such a high value of α.

How can we tell when recall has failed? We might hope to see a rapid degredation in $\overline{F}_B$ when the α_c limit had been reached but actually this is not so. As Bruce *et al* [2] comment, a sudden discontinuity in error is only likely to be seen for very large N. All we can do at present is to look at the quality of recall and judge on the basis of our application whether recall is acceptable and at what level of noise recall becomes unreliable.

For this reason we have divided the results between error in bits which should be 1 - shown in Figure 1 - and error for bits which should be 0 - shown in Figure 2.

Clearly for our patterns the weight of knowledge is in the 1 bits which are only a small fraction of N. We envisage that a post-processing component should be able to map the minimally noisy recalled pattern $\tilde{V}^{(\mu)}$ to the nominated pattern $V^{(\mu)}$.

From Figures 1 and 2 we see that $\overline{F}_B$ increases with m_0, and that the robustness of the network in resisting noise corresponds to α every case.

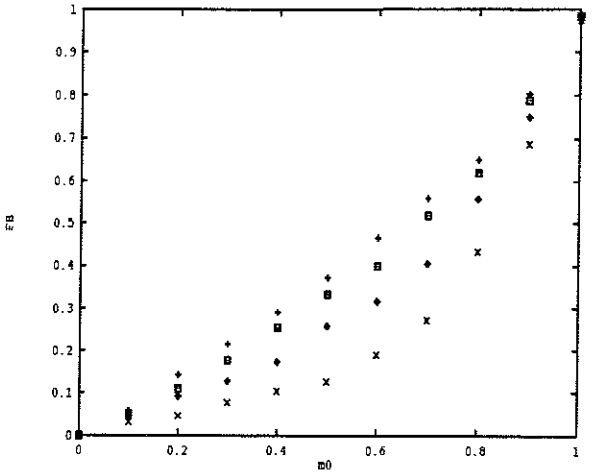


Figure 1: Mean error free fraction of bits $\overline{F}_B$ for 1 bits only against initial noise, m_0. W1 $\times$, W2 $\diamond$, W3 $\square$, W4 $+$.

5.2 Complexity

Table 2 shows the size of the matrix in non-zero elements as *space*. By using a 0,1 representation in the training patterns we generate very sparse weight matrices which have space requirements much less than $O(N^2)$. We see from Table 2 that the storage requirement had an upper bound of $O(N^{\frac{3}{2}})$.

Looking at the cpu. time to convergence results in Table 2 we see that the network stabilized in time with

	W1	W2	W3	W4
N	260	673	921	1357
space	4130	12659	22882	39185
$N^{\frac{3}{2}}$	4192	17459	27950	49988
time	220	726	1607	5260

Table 2: Matrix time and space complexity upper bounds. Note that *space* is given as number of non-zero elements in W and *time* corresponds to maximum cpu. seconds to convergence.

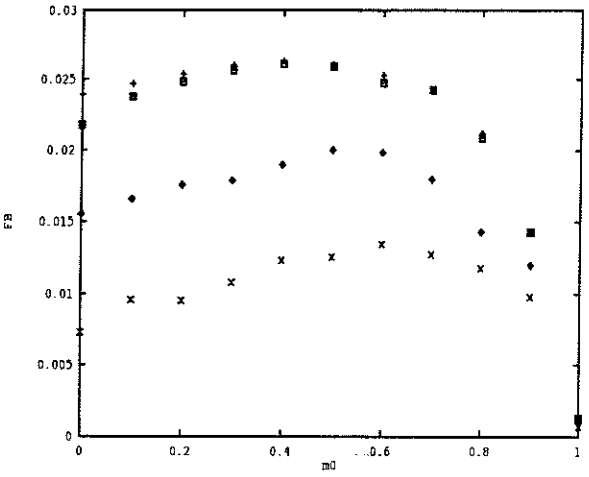


Figure 2: Mean error free fraction of bits $\overline{F}_B$ for 0 bits only against initial noise, m_0. W1 $\times$, W2 $\diamond$, W3 $\Box$, W4 $+$.

an upper bound that appears to be polynomial on N.

6 CONCLUSION

We have shown the strong influence of α on the mean fraction of bits in error, $\overline{F}_B$ observed by many researchers in other fields. The interesting result is that the Hopfield network was trained on natural language sentence patterns and the matrices had much higher values of α than were previously reported for the critical value α_c. In previous experiments with artificial patterns (e.g. [11]) it was found that correlations improved recall. We can confirm this with the natural language patterns we used.

The amount of storage required was seen to be bounded by $O(N^{\frac{3}{2}})$ which is acceptable for NLP work, but the time complexity is a cause for concern.

Future work will look at much larger matrices and see if the behaviour is consistent with those reported here as well as better storage prescriptions such as those cited by Tarassenko *et al* in [10].

References

[1] D. Amit, H. Gutfeund, and H. Sompolinsky. *Phys. Rev. A*, 1987.

[2] A. Bruce, E. Gardner, and D. Wallace. Dynamics and statistical mechanics of the hopfield model. *Journal of Physics A*, 20:2909–2934, 1987.

[3] N. Collier and K. Takahashi. Sentence alignment in parallel corpora: The asahi corpus of newspaper editorials. Technical Report 95/11, Centre for Computational Linguistics, UMIST, Manchester, October 1995.

[4] S. Gallant. A practical approach for representing context and for performing word sense disambiguation using neural networks. *Neural Computation*, 3:293–309, 1991.

[5] E. Gardner. Optimal basins of attraction in randomly sparse neural network models. *Journal of Physics A*, 22:1969–1974, June 1989.

[6] D. Grensing, R. Kühn, and J. van Hemming. Storing patterns in a spin-glass model of neural networks near saturation. *Journal of Physics A*, 20:2935–2947, 1987.

[7] J.J. Hopfield. *Proceedings of the National Academy of Science*, 79:2554+, 1982.

[8] N. Ide and J. Veronis. Very large neural networks for word sense disambiguation. In *ECAI90: 9th European Conference on Artificial Intelligence, Stockholm, Sweden*, pages 366–368, August 6–10 1990.

[9] T. Ozeki and H. Nishimoto. Noise distributions in retrieval dynamics of the hopfield model. *Journal of Physics A*, 27:7061–7068, 1994.

[10] L. Tarassenko, B. Seifert, J. Tombs, J. Reynolds, and A. Murray. Neural network architectures for associative memory. In *First IEE International Conference on Artificial Neural Networks*, 1990.

[11] W. Whyte, D. Sherrington, and A. Coolen. Competition between pattern recall and sequence processing in a neural network storing correlated patterns. *Journal of Physics A*, 28:3421–3437, 1995.

DISCOVERING A "MINIMAL" LANGUAGE MODEL

Leona F. Fass

Abstract

We describe a formal, context-free language learning problem we constrained, to obtain a unique result. We show that through appropriate knowledge representation, a finite minimal syntactic model of linguistic knowledge may be discovered. Even if the language it characterizes is infinite, once the finite model is acquired we consider the language to be learned. While stressing syntactic modeling and formal languages, we also discuss relationships with other linguistic aspects and applications to other language domains.

1 INTRODUCTION

An intelligent system that learns or acquires a language may emphasize lexical, semantic or syntactic aspects, but must deal with any such issue effectively. We might seek a learning system or device that acquires precisely a specified body of linguistic knowledge as its *behavior.* Then the design and successful construction of the system may depend on characterizing this specified behavior by finite means. A designer may impose constraints so that, from within a domain of potential learning devices, a system with selected properties may be obtained. If an intelligent system learns a language effectively, it might even be constrained to produce that behavior in some specified, unique fashion.

Here we review our theoretically-oriented research in the development of learning systems that convey a linguistic behavior finitely. We describe our perspective on language learning and discuss a formal, context-free, language learning problem we constrained, to obtain a unique result. Some extensions and possible applications also will be described.

2 LANGUAGE LEARNING AS LINGUISTIC KNOWLEDGE MODELING

Whether addressing formal, programming or natural language learning problems, we may find that the behavior, or body of knowledge, to-be-acquired by an intelligent system is infinite. For example, it might consider learning all of the specific strings in the formal language $a^n b^{2n}$; all of the statements, including all possible programs, that could ever exist in Pascal; or all of the sentences that have been produced, and that yet may be produced, in an ever-growing natural language such as English. Obviously, no intelligent system would or could realistically acquire any of the languages just described by "memorizing" (storing) its entire (possibly, potentially) infinite body of knowledge. Rather, the system could learn by acquiring a characterizing *model* that finitely represents exactly the knowledge in the infinite (string-,statement-, or sentence-) set. A successful learning system might discover the requisite properties of a model by accessing the information available in an appropriate, finite, linguistic knowledge sample. Of course the actual construction of a language model must be completed finitely.

If a finite model of linguistic knowledge exists and can be effectively obtained, we consider the language it represents to be *learnable.* If an intelligent system obtains the model it has *learned* the language, for it has precise information about how it must always *behave.*

3 OUR SPECIFIC LANGUAGE-LEARNING PERSPECTIVE

We have taken a strongly syntax-oriented approach to problems of linguistic knowledge acquisition, considering construction of *syntactic* language *models* by means of grammatical inference. From our perspective, acquisition of a generative model – e.g., a grammar producing exactly a language, or of a recognitive model – e.g. a recognizer accepting exactly a language, is equivalent to acquisition of the language itself. "Knowing" the model enables the generation or processing of everything *in* the language–as opposed to "everything else," that is *not.* An intelligent system that possesses such knowledge has *learned,* in a sense not unlike Valiant's [14], for it can determine the specific elements of the language's string-, statement- or sentence-set. A critical goal of such knowledge acquisition research is determining a representation that conveys all of a (possibly infinite) language's constructs in a finite fashion that is "acquirable," or learnable, as we have defined, above. A finite model must distinguish correct linguistic behavior from all such behavior that is incorrect. A system designer must have the facility to convey to the intelligent system an effective decision procedure to determine what syntactic structures comprise the language, and what do not.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/349 ©1996 FLAIRS

Seeking a solution to the particular problem of context-free language acquisition we determined a representation of linguistic knowledge that for our purposes, proved optimal in satisfying the criteria just described. Not only did we find a finite syntactic characterization, we found a representation of any such context-free language[1] (CFL) that, in a very precise sense, is unique and *minimal*. We constrained the learning process so that an effective procedure for establishing linguistic membership could be defined for any CFL, even if infinite. This enabled us to discover a minimal language model and (with constraints) to verify whether or not a *potential* such model might be correct.

We describe below the evolution in our thinking that led us to obtain these results.

4 THE PROBLEM WE FIRST CONSIDERED

When we first considered the problem of context-free language acquisition, we sought a technique by which a learning system might produce a syntactic model (grammar or recognizer) for a CFL, given a sentence sample.

For example, given the formal language sample: **abb,aabbbb, aaabbbbbb**; an intelligent system might have the means to guess that the sample is representative of the CFL $\{ a^n b^{2n} \mid n \geq 1 \}$. The system might then produce what it believed to be a characterizing grammar:

$$\sigma \rightarrow abb$$
$$\sigma \rightarrow a\sigma bb.$$

If it had guessed correctly, the system would be able to produce any and all of the $a^n b^{2n}$ language from that grammar, and would have acquired that body of knowledge in our syntactic-modeling sense.

Now suppose the system also received as part of the sample: **ab, aabb, aaabbb** and **abccba**.

We would expect an *intelligent* system to change its guess and produce an expanded syntactic model, to account for the new linguistic knowledge. But, how would the system know when it had expanded the model enough? Or, too much?

Now suppose that instead of the sample shown above the intelligent system were given a sample: **abc, aabbcc, aaabbbccc,...** Could it possibly guess that these strings are representatives of the context-free language $\{ a^i b^j c^k \mid i = j \text{ or } j = k; i,j,k \geq 1 \}$? If so, could it guess that each of the sample strings shown *must* be ambiguously derived, and thus produce a suitable characterizing grammar or recognizer? This seems to be quite a lot for a system, no matter how intelligent, to guess (and can involve questions relating to inherent ambiguity that are *known* to be undecidable).

[1] Due to space limitations, throughout this paper we assume the reader to have a basic familiarity with CFLs, recognizers and grammars, and formal language theory.

While we had originally sought techniques by which a system could construct a syntactic model for a CFL from a language sample, we soon revised our plan. For, it turned out that properties of arbitrary CFLs, illustrated in the examples above, prevented us from achieving our original, desired results. Central recursion or phrase-embedding, and inherent ambiguity, made it impossible to discover precisely characterizing generative rules, given sentence samples, in the general CFL case. Non-deterministic operation of traditional pushdown-automaton recognitive devices made it impossible, in the *general* CFL case, to determine structure of a finite characterizing recognizer, given a sample of sentences it is to accept.

Based on suggestion of Levy and Joshi [9] we changed our means of language representation. We then circumvented the above-noted problems and established the existence of finite, learnable CFL models. We also showed these models can be effectively obtained from specific (for each language) suitably-represented finite samples of linguistic knowledge.

5 THE REVISED APPROACH WE CHOSE

Levy and Joshi [9] proposed that grammatical constructs could be reduced if language itself conveyed some structural information. In the case of CFLs they suggested representation of sentences in a skeletal structured form (as derivation trees, from a known CF grammar for the language, but with syntactic category names deleted from interior nodes). The phrase structure of sentences was thus conveyed by their "shapes".

Given a CF grammar for a CFL [9] showed a recognitive device for the structured language S could be obtained, and converted to a deterministic device, if necessary (they defined for the purpose a class of skeletal automata, bottom-up recognitive tree processors, accepting sentence structures with "the right shapes"). Levy and Joshi also showed that a CF grammar for a structured language could be obtained from a *given* skeletal automaton recognitive device.

The structured representation of sentences seemed likely to resolve those problems we'd encountered, seeking generative or recognitive models of CF linguistic knowledge from sample sentence *strings* (e.g.; discovery of embedded phrases, or, multiple structurally-different derivations, representing strings, or sentences, that must be ambiguously derived). We chose to represent CFL sentences structurally, and were able to show that a finite structure sample characterized the entire body of linguistic knowledge. The sample determined finite recognitive and generative models through which the knowledge could be acquired. Furthermore, the recognitive and generative models defined by the sample could be determined in a unique, minimal way.

6 SOME OF OUR RESULTS

As noted above, Levy and Joshi [9] showed that a recognitive device, call it M, for a structured CFL could be obtained from a given CFG G, and vice versa: a G can be found from an M accepting the language structures.

What we have shown is that one need not begin with a *given* grammar or recognitive device, for an "optimal" representation of the linguistic knowledge is obtainable directly from the language structures. As we now describe, we developed these results by analyzing, adapting and generalizing similar results from what we now classify as *classical* regular language and sequential machine theory.

The work of Myhill [11], Nerode [12] and Moore [10] establishes the relationships among regular grammars, finite state sequential machines and unique classes of language strings. Furthermore, Moore [10] gives us the foundation to experiment finitely with sequential machines and the strings they characterize, to synthesize or construct a minimal *regular* language model.

What we have shown [2,3] is that corresponding results may be determined for the recognitive and generative models of a context-free language, *as long as the CFL is represented structurally.*

Given a CFL with structural representation S (S can be the skeletal derivation trees – i.e., with syntactic category names deleted – for all derivations of sentences in the CFL, obtained from any know CF grammar for that CFL) we have shown that S, itself, defines a unique skeletal recognitive device *M* and a corresponding unique generative grammar *G*. Generalizing Myhill [11] and Nerode [12] we have shown that congruence classes of structures within S define the recognitive states of *M*, and the generative syntactic categories of *G*. Furthermore generalizing, Moore's [10] gedanken-experiments we have shown that in either case, *M* or *G*, we have a "minimal deterministic" result.

Finally, in the style of [11,12] we have shown that for *any* set of skeletal structures S we theoretically have a recognitive or generative model. However, *only if* S is the structured representation of a CFL will the resultant *M* or *G* be *finite*. Adapting a result on experiment size (in our case, experimental *sample* size) established by Thatcher and Wright [13], we have shown that the resultant finite *M* or *G* is obtainable from a finite structure sample (representatives of the finite set of congruence classes that define the minimal linguistic model). In both cases we have defined, for any CFL with specified structure S, a specific linguistic sample and technique, from which an optimal model, representing the body of linguistic knowledge S, may be obtained. For any such linguistic knowledge S we may *learn* it by discovery of this minimal language model.

In our earlier work we have given proofs of existence, and techniques for construction, of a learnable model for any structured CFL [2,3]. We believe that our results, particularly with respect to a learnable generative grammar *G*, are relevant to other aspects of language-learning (and not just the syntactic modeling of CFLs). We will discuss these adaptions of our results in Section 7, below. But first we present an additional, foundational result.

Theorem: Let S be the structural representation of a CFL as described above. There is a CFG *G* for the CFL that defines the structures S and that has the following properties (corresponding results can be stated for the recognizer *M*):

(1) *G* is backwards-deterministic: no two distinct productions have the same right-hand-side.

(2) *G* unambiguously generates the CFL *structures* S.

(3) *G* is the smallest possible such grammar (it has the least number of syntactic categories and productions needed to generate a structurally-equivalent language, backwards-deterministically).

(4) Given a G claimed to be the optimal *G* just described, a specific finite linguistic sample is defined to test G for incorrectness and possibly, by default, to verify it as "correct."

Proof sketch:

G is the learnable model we have discussed.

(1) In the construction method outlined above, and detailed in our earlier work [2-6], the components of *G* are obtained from congruence classes of linguistic structures. In particular, each variable, or syntactic category name, becomes a *representative* for the congruence class of structures it derives. No two distinct variables can represent the same congruence class, or generate the same language structures. Hence no distinct productions will have the same right-hand-side.

(2) Is a consequence of (1)

(3) It can be shown that since the components of *G* are determined from congruence classes of structures within the structured CFL S, it contains no useless symbols or rules, and no redundant ones. It can be shown that any other structurally-equivalent, backwards-deterministic CFL is either isomorphic to *G* (its components represent precisely the same congruence classes of structures as do the components of *G*) or (mapping through the congruence classes) there is a homomorphism from the useful components of the alternate CFG onto *G*.

(4) The imposition of structural information onto the CFL allows us to define a linguistic *complement* of a language, and to *decide* whether a structure is in the structured CFL S, or its complement

(relative to a structural domain). If *G* is completely determined by a specific sample of the structures S, a candidate grammar can be effectively tested (as defined in [1]) by seeing if it produces all of the characterizing sample of S and *none* of the relative (finite) sample of its complement.

For example, suppose the minimal language model *G* has n variables (representing n classes of linguistic structures). We have shown [2,3,6] that the finite sample of all depth ≤2n structures of S (skeletal derivation trees) comprises a sufficient knowledge sample for access by an intelligent system that, given the sample, will *discover* *G*. The sample sufficient for testing an n-variable, backwards-deterministic *potential* grammar, claimed to model S, will be all depth ≤2n structures in S, and all depth ≤2n structures that are *not* (again, relative to an appropriately defined domain of structures.)

E.g. a grammar for the CFL:

$\{ a^n b^n \cup a^n b^{2n} \mid n \geq 1 \}$ is

$\sigma \rightarrow ab$	$\sigma \rightarrow abb$
$\sigma \rightarrow a\tau b$	$\sigma \rightarrow a\rho bb$
$\tau \rightarrow a\tau b$	$\rho \rightarrow a\rho bb$
$\tau \rightarrow ab$	$\rho \rightarrow abb$

The *minimal* backwards-deterministic structurally-equivalent grammar for the CFL is

$\sigma_1 \rightarrow ab$	$\sigma_2 \rightarrow abb$
$\sigma_1 \rightarrow a\sigma_1 b$	$\sigma_2 \rightarrow a\sigma_2 bb$

(we note it has two initial sentence symbols).

All depth ≤4 skeletal structures that would be produced by the minimal grammar would provide sufficient knowledge to a system that could *discover* *G*. All *complementary* depth ≤4 skeletal structures provide the additional examples needed for testing (these would be 2-,3- and 4-branching structures, labeled on terminal nodes with a's and b's, that do *not* belong to the structured language defined by *G*). A 2-variable backwards-deterministic *potentially* structurally-equivalent grammar would be established as correct if it derived *all* of the former finite linguistic sample and *none* of the latter.

Because we have constrained our language acquisition problem and imposed structure on the linguistic knowledge: we can define a suitable language sample; we can define a complementary sample; and we have the procedures to decide which elements are members of which set.

Thus, we have described two means of discovering a minimal language model, applicable in the case of an intelligent system that seeks to learn a language that is context-free.

7 SOME POTENTIAL APPLICATIONS

We found a minimal representation of linguistic knowledge by constraining the specific problem we first considered: learning of CFLs through the acquisition of syntactic models. As we noted above, an intelligent system might deal with lexical or semantic aspects, and not just syntax. It might deal with programming or natural languages, and not just the formal language domain. We believe our techniques and results have applications in these area, which we now briefly describe.

Our work is related to the issues of lexical knowledge representation and acquisition as discussed in [8] in the general sense that syntactic, semantic and lexical aspects of language interact. It is also related in the specific sense that our representation for syntactic linguistic modeling can support lexical and semantic components. The minimal model *G* described above (and detailed in the theorem of Section 6) has components that correspond to classes of structures (skeletal trees and subtrees, from the structured language S.) Terminal vocabulary lexical items can be incorporated into the structures and will be represented minimally by *G*. *Semantics* may be incorporated through, say, the use of attribute grammars (attaching meaning at each of the structure nodes). The backwards-determinism of *G* and its unambiguous characterization of language structures can resolve some problems of polysemy and ambiguity with which a language learning system might deal. In any case, *G* provides a minimal, in our view, *optimal* framework to represent and acquire the body of linguistic knowledge. As noted above, it also provides a foundation for testing a potential language model, to determine if it is correct.

Our results can bear on issues of *programming language* representation and processing, since many such languages are syntactically context-free. We have long believed that our determination of minimal CF language models can have utility in automated, minimal compiler design.

Finally, our work relates to natural language considerations, if we agree that much (if not all) of natural language can be described as context-free. Given the skeletal structures obtained from *any* representative CF derivations in a natural language, samples of increasing depths will *eventually* identify a great portion of the language in-the-limit (subsequent models will not change) [7]. At any fixed point in time, we may accept that the ever-growing natural language is minimally identified [4-6], and modelled approximately.

References

[1] Cherniavsky, J.C.,R. Statman and M. Velauthapillai, "Testing and Inductive Inference: Abstract Approaches", Georgetown University Department of Computer Science Series. TR-5 1987. Also appears in Proc. of the First Workshop on Computational Learning Theory, Morgan-Kaufmann, 1988.

[2] Fass, L.F., "Learning Context-Free Languages from their Structured Sentences", *SIGACT News*, Vol 15, No. 3 (1983), pp. 24-35.

[3] Fass, L.F., "A Minimal Deterministic Acceptor for Any (Structured) Context-Free Language", presented at the *1990-91 Annual Meeting of the Linguistic Society of America*, Session on Computational Linguistics, Chicago, January, 1991; 35 pp., abstracted in *Meeting Handbook*, p.17.

[4] Fass, L.F., "Applying Some CFL Learnability Results to Natural Language Learning", presented at *AAAI-Stanford Spring Symposium Series, Symposium on Machine Learning of Natural Language and Ontology,* Stanford, March, 1991. Appears in *Symposium Notes*, pp. 48-52. Notes also released as German AI Institute, Kaiserslautern TR DFKI D-91-09.

[5] Fass, L.F., "Canonical (CF) Grammars and Natural Language", presented at the *1993 Annual Meeting of the Linguistic Society of America*, Los Angeles, January 1993. Research Overview, 15 pp., abstracted in Meeting Handbook. p.23.

[6] Fass, LF., " 'Correct' Language Models" presented at the *Tenth International Congress on Logic, Methodology and Philosophy of Science*, Section on Linguistics, Florence, Italy, August 1995. Abstracted in *Congress Volume of Abstracts*, p. 563.

[7] Gold, E.M., "Language Identification in the Limit," *Inf. Contr.*, Vol. 10 (1967), pp. 447-474.

[8] Klavans, J. [Ed], *Notes of the AAAI - Stanford Spring Symposium on the Representation and of Acquisition of Lexical Knowledge: Polysemy, Ambiguity, Generativity,* Stanford, March, 1995, released as AAAI TR SS-95-01.

[9] Levy, L.S., and A.K. Joshi, "Skeletal Structural Descriptions," *Inf. Contr.*, Vol 39 (1978), pp. 192-211.

[10] Moore, E.F., "Gedanken-Experiments on Sequential Machines," in *Automata Studies*, J. McCarthy and C.E. Shannon (Eds.), Princeton University Press, Princeton, N.J. 1956, pp. 129-153.

[11] Myhill, J. "Finite Automata and the Representation of Events, " *WADC Tech. Rept.*, 57-624, Wright-Patterson AFB, 1957.

[12] Nerode, A., "Linear Automaton Transformations," *Proc. AMS*, IX(1959), pp. 541-544.

[13] Thatcher, J.W., and J.B. Wright, "Generalized Finite Automata Theory with an Application to a Decision Problem of Second Order Logic," *Math. Sys. Th.*, Vol 2 (1970), pp. 57-81.

[14] Valiant, L, "A Theory of the Learnable," *Comm. ACM*, Vol. 27 (1984), pp. 1134-1142.

Leona F. Fass received a B.S. in Mathematics and Science Education from Cornell University and an M.S.E. and Ph.D. in Computer and Information Science from the University of Pennsylvania. Prior to obtaining her Ph.D. she held research, administrative and/or teaching positions at Penn and Temple University. Since then she has been on the faculties of the University of California, Georgetown University and the Naval Postgraduate School. Her research primarily has focused on language structure and processing; knowledge acquisition; and the general interactions of logic, language and computation. She has had particular interest in inductive inference processes, and applications/adaptations of interference results to the practical domain. Dr. Fass may be reached at

Mailing address:
Post Office Box 2914
Carmel California 93921

A GEOMETRIC REPRESENTATION FOR FUNCTIONAL RECOGNITION

Ellen L. Walker
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180
walkere@cs.rpi.edu

Abstract

Functional recognition is a paradigm that represents objects by their functional properties rather than by their geometric structure. The use of this paradigm constrains both the representation of object models and the reasoning necessary to recognize them. Common representation and reasoning techniques for structural recognition, such as CAD-based vision, are unsuited to functional recognition. This paper discusses issues in representation and reasoning with functional models, and describe a general reasoning framework that addresses many of these issues.

1 INTRODUCTION

The functional paradigm of object recognition models and recognizes objects by their functional properties, rather than by their geometric structure. This paradigm is embodied in systems such as FUR [Dima89] and Gruff-2[Star94]. Functional object models are based on geometric constraints implicit in the functionality of the objects, rather than on explicit structural descriptions of the object's geometry. Thus, functional models describe larger, more natural object classes than similar structural models do.

In this paper we describe some issues in geometric representation and reasoning with functional models, and describe a geometric reasoning framework that addresses many of these issues.

The discussion in this paper is limited to objects whose functions can be determined from their static shapes; we don't consider recognition of objects based on their dynamics (e.g. scissors), or their ability to change configuration (e.g. a sofa-bed will be treated no differently from a sofa). Another issue not considered here is material properties; no distinction, for example, is made between soft surfaces and hard surfaces.

2 FUNCTIONAL MODEL COMPONENTS

The first requirement of a functional representation is to represent an object's component parts. Typically, functional object recognition systems (e.g. [Rivl94]) first decompose the object model into functional parts, and then match these to the structural parts of the object being recognized.

Both representation and matching are more difficult because the mapping between functional and structural parts is many-to-many, and not all structural parts are relevant to the object's function. The many-to-many mapping is necessary because multiple non-contiguous geometric features can serve a single functional purpose. An example is the legs of a chair together providing support for the seat. On the other hand, a single part can provide multiple functions, as when wheels of a car provide both support and mobility.

Although the necessity for a many-to-many mapping is recognized in the literature [Rivl94], currently implemented systems are limited to a mapping each functional component to a single structural primitive [Rivl94] or have only limited ability to group structural parts into functional parts [Star94].

Rivlin's system [Rivl94], requires that each functional part be mapped to a single superquadric, which enables functional parts to be directly extracted from range images, but limits the shape of functional parts to shapes that can be described by a single superquadric. In particular, non-contiguous objects working together to solve a functional purpose cannot be recognized as a single functional part.

The Gruff-2 system [Star94] is not part-oriented in the sense of labeling object parts with functional labels, but instead executes procedures to determine whether specific surfaces satisfy given constraints. Some of these procedures allow surfaces to be grouped together to form a "virtual surface" for the sake of functionality, but grouping is not addressed in the general case.

A more general solution is to use hierarchical models, so that each functional part maps to either a structural primitive or a component that contains either structural or functional subparts. The work of Thadani [Thad90] describes methods for grouping primitive functional components into hierarchical functional descriptions based on higher-level models and specified connections among subparts. Similar hierarchical grouping based on geometric relationships has been done in structural object recognition [Moha92]. The methodology described in this paper combines functional and structural methods of hierarchical grouping.

The second component of a functional representation is a set of constraints on and among the object's functional parts. For the class of objects whose function depends on geometry, a small number of primitive geometric constraints have been sufficient to describe functional parts and their relationships to each other. [Star94]. These are: *shape*,

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/354 ©1996 FLAIRS

size, relative *pose* (position and orientation), *support*, and *clearance*. The first three constraints are common in all forms of structural models. Support has been considered for some types of recognition [Mulg92, Mann94], and although clearance has received little attention in structural modeling, it is important for applications such as assembly planning [Sand90]. All of these constraints are needed to model a simple piece of furniture such as a bed (Figure 1).

In summary, a functional model must describe the object's functional parts and the geometric constraints that define the functional characteristics of the object without imposing structural constraints other than those required for functionality. It must allow irrelevant structures without allowing structures that violate functional constraints.

3 REPRESENTING FUNCTIONAL MODELS

Several classes of model representations have been traditionally used in computer vision, including declarative models, procedural models, and frames. This section considers the applicability of each of these model classes for the representation of functional models.

Declarative structural modeling methodologies such as constructive solid geometry and boundary representations [Requ80] are common, standard implementations are easy to find, and matching for such models is well-understood [Grim90]. However, with this type of model, an object cannot be represented without describing its complete structure. In functional representations, objects of many different shapes can have the same function. To represent a given object using a declarative structural model would require assumptions to be made about its shape. Each structural variation of an object (e.g. square vs. round table) needs a separate model. Thus, declarative structural models are too specific for functional representation.

Procedural models use sequences of function calls to describe objects. Since procedures can test for exactly the required functional constraints, no additional structure is imposed on the objects [Star94]. Matching in this case is a matter of applying the model's tests to the observed object's structure.

Procedural models cannot be developed without knowing how the models will be used. In [Dima89] three *modes of activity* are presented: *search mode* to find an instance of an object, *verify mode* to determine whether a given object satisfies a functional model, and *generate mode* to generate a geometric object for a given functional model. All three modes require the same information, but a given procedure can only be used in a single mode. As an example, the bed model in Gruff-2 [Star94] determines whether an object can serve as a bed [verify mode], but a different models would be required to complete a partially-specified bed [generate mode].

Frame-based models [Fike85] combine features of both declarative and procedural models. Procedures can implement functional constraints, but they are attached to declarative frames, which structure the constraints and schedule procedure activation based on the system's current knowledge.[1] Thus, a single model can be applied to different tasks. An example is the 3D Frame-based Object Recognition and Modeling system (3D FORM) [Walk88].

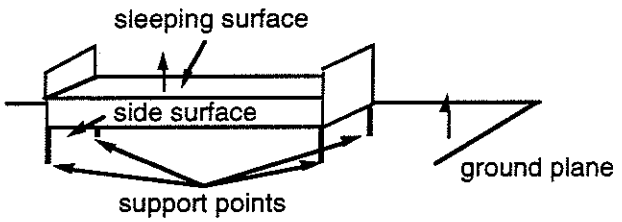


Figure 1: Parts of a bed

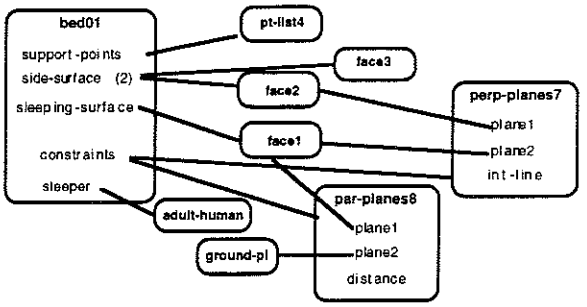


Figure 2: Instance of a bed frame with two geometric relationships

4 REPRESENTATION IN 3D FORM

Each 3D FORM model consists of three parts: a primitive geometric feature (where appropriate), a list of object parts (themselves 3D FORM models), and a set of geometric relationships among the primitive and parts that must be satisfied. The model is implemented as a network of frames where each object is represented by a single frame containing slots for primitive features, links to its parts (pointers to other frames), and specifications for its geometric relationships. Examples of models are called *instances* and are themselves frames. (See Figure 2). In particular, observed objects, when recognized, become instances of the appropriate model. 3D FORM also provides inheritance through a class hierarchy, as does Gruff-2.

In the current implementation of 3D FORM, the primitive features are numeric ranges, points, lines and planes. Each primitive feature is represented as a frame, with slots to express its constraints. The slots of points, lines, and planes are designed so that each primitive can be specified in terms of lower primitives that it contains and/or higher primitives that it is contained in. For example, the slots of a line segment frame include points on the segment, its endpoints, its orientation vector, and normals of planes that it lies in.

Empty slots are permitted in frame instances, and demons are provided to automatically compute slot values where appropriate. Structurally incomplete objects are simply represented using empty slots.

Geometric relationships (also frames) are specified within the object definition. When a relationship is evaluated, the system ensures that the relationship is not inconsistent with the given arguments, and, if necessary, hypothesizes missing arguments that are consistent with the relationship. A perpendicularity relationship, for example, could verify the perpendicularity of two planar surfaces, or hypothesize one given the other. In this way, the reasoning performed by 3D FORM depends on current knowledge.

The abilities of 3D FORM to represent structurally incomplete objects (including hypotheses), and to perform

[1] Technically, the models used by Gruff-2 are also frames, but do not make use of the declarative nature of slots, nor of demons (or message passing) to control behavior dynamically.

The sleeping surface is long enough and large enough to support the sleeper.

range-geq
comp-larger (value (DIAMETER SLEEPING-SURFACE))
comp-smaller (value (HEIGHT SLEEPER))

range-geq
comp-larger (value (AREA SLEEPING-SURFACE))
comp-smaller (value (AREA SUPPORT-POLY SLEEPER)

The sleeping surface is parallel to the ground and has clearance above it.

parallel-planes
rel-pl1: (value (geom-feature SLEEPING-SURFACE))
rel-pl2: (value (geom-feature GROUND-SURFACE))
rel-dist: *king-bed-height*

clearance
clear-surface: (value SLEEPING-SURFACE)
clear-extent: *king-bed-top-space*
clear-obstacles: (value PARTS)

There is clearance to each side of the bed

every SIDE-EDGES some SIDE-SURFACES

perpendicular-planes
rel-plane1: (value (geom-feature (SLEEPING-SURFACE)))
rel-plane2: (value (geom-feature (SIDE-SURFACES))
rel-intline: (value (geom-feature (SIDE-EDGES)))

every SIDE-SURFACES

clearance
clear-surface: (value SIDE-SURFACES)
clear-extent: *king-bed-side-space*
clear-obstacles: (value PARTS)

The support vertices are bottom-most relative to the normal to the ground plane, and they lie on the ground plane

every SUPPORT-VERTICES every PART-VERTICES

ordered-pts
ordered-pt1: (value (geom-feature (SUPPORT-VERTICES))
ordered-pt2: (value (geom-feature (PART-VERTICES))
ordered-line: (value (norm (geom-feature (GROUND-SURFACE)))

every SUPPORT-VERTICES

points-on-plane
onpl-pts: (value (geom-feature SUPPORT-VERTICES))
onpl-plane: (value (geom-feature GROUND-SURFACE))
onpl-dist: *support-tolerance*

The projection of the center of mass lies within the support convex hull (formed by the support vertices)

convex-hull
ch-vertices: (value (SUPPORT-VERTICES . all))
ch-polygon: (value (SUPPORT-CONVEX-HULL))

point-in-polygon
pip-poly: (value SUPPORT-CONVEX-HULL)
pip-point: (value CENTER-OF-MASS-PROJ)
pip-tol: *support-polygon-tolerance*

Figure 3: 3D FORM constraints for a bed

flexible knowledge-dependent reasoning make it particularly suitable for functional recognition.

5 FUNCTIONAL MODELS IN 3D FORM

This section describes the representation of a functional model using 3D FORM frames. Sections 5 and 6 will use the example of a bed (Figure 1).

The first requirement for representing a functional model is the ability to represent functional parts. In 3D FORM, parts are represented by frames attached to slots of the parent frame (Figure 2). These frames can represent single geometric primitives or grouped structures.

Because the 3D FORM representation is hierarchical, each functional part can itself have subparts — frames that describe either functional or structural components. Examples include multiple non-adjacent planar faces grouped into a sleeping surface, and multiple support chains (themselves functional objects) grouped into a support structure. Hierarchical groupings effectively create a one-to-many relationship between functional and structural components.

Because the representation is not entirely dependent on extracted features, object parts can be geometric constructions that are useful for reasoning, although not observable. For example, the side surfaces of a bed are not required for functionality; neither a hammock nor a platform bed has any. However, these surfaces are still useful to bound the side clearance polygons.

contains a frame for the range $[6,\infty]$. Alternatively, shape and size constraints can be expressed by relating quantities from one frame to another. This alternative is shown by the first two constraints in Figure 3, using the SLEEPER slot to represent the user of the bed.

Relative pose constraints are imposed explicitly in the object model, either between parts of the object in question or between object parts and global objects, e.g. the ground plane. The relative pose of the sleeping surface of a bed and a fixed ground plane is expressed by the relationship template:

```
(parallel-planes
     (rel-pl1 (value (geom-feature sleeping-surface))
     (rel-pl2 'ground-plane)
)
```

Support and clearance constraints also are imposed by relationship templates. The support relationship constrains the relative locations of the object's support polygon and center of mass (computed slots in the object frame). A clearance relationship defines a volume of space relative to the object that must be empty, taking the complete set of object parts as one argument, and the required clear volume as the other.

The complete set of relationship templates for the idealized bed frame is shown in Figure 3. Note the use of quantifiers "every" and "some" to apply constraints appropriately to multi-valued slots.

6 FUNCTIONAL RECOGNITION IN 3D FORM

In its most general form, the problem of functional recognition is a matching problem: given geometric information derived from an object's image and one or more functional models, determine correspondences between scene geometry and the functional models. This definition subsumes several types of questions, such as "Where is the bed in the image?", "Is the object a (better) bed or a chair?", and "Why is the object not a bed?". Depending on the input image(s), the data could be a complete or partial three-dimensional object description.

The basic step in 3D FORM functional recognition is to apply a functional model to a generic object. In [Walk89], this process, called *specialization*, is described in detail. First, an object frame is created to represent geometric information from the image, including structural relationships among object parts. Next, the frame is defined as the functional type (BED) and an assignment of

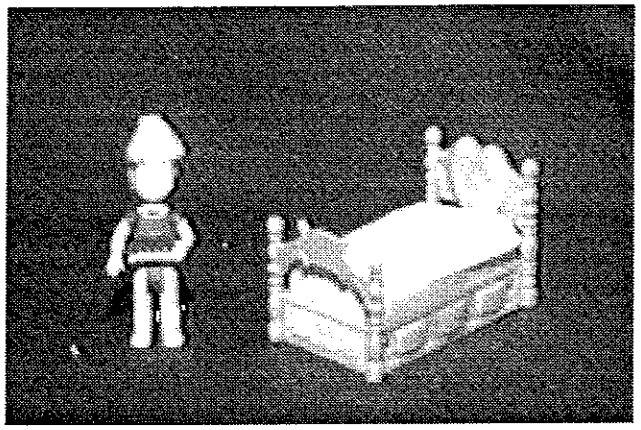


Figure 4: Bed scene

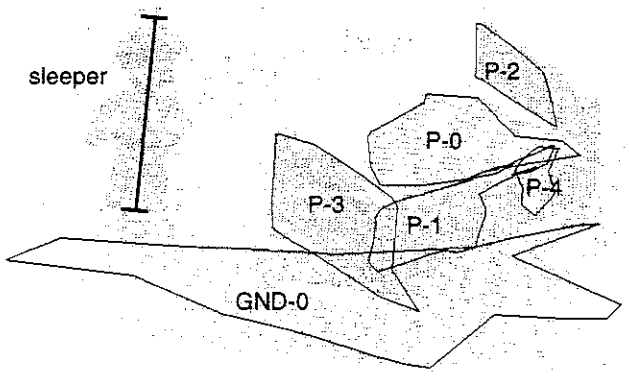


Figure 5: Labeled range image of bed scene

```
(object1
  (instance 3d-object)
  (parts P-0 P-1 P-2 P-3
P-4)
)
```

```
(object1
  (instance bed)
  (parts
     P-0 P-1 P-2 P-3 P-4)
  (sleeping-surface P-0)
  (side-surfaces
     P-1 P-2 P-3)
  (side-edges edge29 edge28
     edge27))
```

Figure 6: Object frame before and after specialization

object parts to functional parts is made. This assignment must satisfy all constraints in the object. Finally, the relationships of the new object are evaluated, including hypothesizing missing parts. Parts that are irrelevant to the object's function are not used in specialized slots Parts that prevent functionality cause constraints to fail. Parts that work together to serve a function are grouped together as recursive subparts of the object.

As an example, consider the scene in Figure 4. A doll (the sleeper) is standing next to a bed, which is to be recognized. This scene was imaged by a GRF-1 range finder, then, using a very simple implementation of plane extraction by region-growing, several planar surfaces from the bed (P-0 through P-5) were extracted, as well as a large portion of the ground plane (GND-0) The range image with bounded polygons and labels is shown in Figure 5.

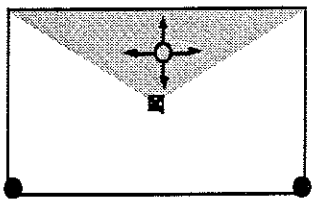


Figure 7: Bed (top view) with two known legs (solid circles) and hypothesized third leg (open circle must lie in shaded region).

For this simple case, the ground plane was assigned to extracted plane GND-0. This assignment is based on the fact that the orientation of the range sensor in the world is roughly known. A 3D-OBJECT frame was created with all other surfaces as its parts (Figure 6a). Next, the plane was specialized to BED. By type-checking, any of the faces was viable for the SLEEPING-SURFACE. However, surface P-0 was chosen because it is the only surface parallel to the ground plane. Note that the correct surface was chosen even though the segmentation was extremely crude.

Once P-0 was assigned as the SLEEPING-SURFACE, surfaces perpendicular to it became potential SIDE-SURFACES. Since P-4 lies in a plane close to that of P-1, they are grouped together to form a composite side-surface (replacing the old P-1).

In reality, only the P-1 surface should be labeled as a side-surface. Although the headboard and footboard of the bed impede access to the sleeping surface, both P-3 and P-2 are labeled side-surfaces, because as the boundary between two separate clearance polygons, they violate neither. A more sophisticated access functional-object (see Section 7) would address this problem.

Each intersection a side-surface with the sleeping-surface was computed as a SIDE-EDGE of the bed. The result is shown in Figure 6b. The side-edges of the bed are examples of hypotheses for parts that were not previously specified in the data. In a closed-loop system, computed parts such as these can be fed back to the lower-level segmentation process to update the segmentation.

When no hypotheses are generated, the evaluation result corresponds to a *functional* or *non-functional* result, using the language of [Star93]. The equivalent of a *possibly functional* result includes hypotheses for the unknown information. An interesting example is shown in Figure 7, where a third leg is hypothesized to stabilize a surface supported by two visible legs.[2] This hypothesis is generated by geometric reasoning within the *support* relationship frame, resulting in an (incomplete) object specification for the missing point. This hypothesis would be a reasonable answer to the question "What would be needed for this object to be a bed?" which no existing functional recognition system can do.

7 GENERALIZING FUNCTIONAL MODELS

The bed frame used as an example so far can be considered an "ideal bed"; it describes a standard piece of furniture but not everything that can be slept upon. To generalize this definition, one must consider that constraints can be satisfied to some degree, and the degree of satisfaction of all the constraints determines the desirability of the particular object as a bed. This section presents

[2] Although any chain of support from an appropriate support point to the sleeping-surface would satisfy the constraint, the simplest hypothesis (a vertical leg) is assumed as a heuristic.

examples of relaxing the constraints of a bed and suggests how they could be implemented within the 3D FORM framework.

With regard to the sleeping surface, the shape constraint can be relaxed to allow shorter beds (e.g. a sofa used as a bed), or to allow multiple surfaces to be grouped into a sleeping surface (e.g. two chairs pushed together). The relaxed constraint could be expressed in terms of the degree to which the surface can support a reclining sleeper, and can be enforced or measured using the area of the support polygon(s) for the best hypothesized position of the sleeper on the sleeping surface. The area constraint in Figure 1 is an approximation to this constraint, assuming a polyhedral approximation to the sleeper in the reclining position, and using the support polygon relative to the sleeping surface (which would be further defined inside the sleeper frame).

The orientation constraint for the sleeping surface can be relaxed to be mostly horizontal to allow a hammock or a reclining chair to be included. Again, with better geometric primitives to model the sleeper as an articulated object with comfort ranges for the articulations, the orientation constraint could be relaxed even further to measure the comfort of the articulated sleeper when placed in the best hypothesized position on the sleeping surface.

Similarly, the constraint on the height of the sleeping surface and clearance to the side of the sleeping surface exist to ensure that the sleeper can get in and out of the bed. These constraints can be relaxed (with appropriate penalties to functionality) to allow beds that are too low (e.g. a sleeping bag or air mattress) or beds that are too high (e.g. the top bunk) provided that access is provided in some other way (e.g. a ladder). This constraint would be implemented as an additional slot of the bed frame to be filled with a new type of functional object (access-provider). This slot would be optional in general, but required in a bunk bed.

To implement these generalizations, the requirement that the object meet all constraints to be recognized should also be relaxed. Instead of a functional / non-functional result, each computation would return a fuzzy measure of the degree to which each constraint is met (as in [Star94]) and these values would be combined into an overall measure of functionality. These measures of functionality would be useful in automatically deciding how to use each object and/or which object(s) to use for a given purpose.

8 CONCLUSION

This paper described some of the issues in developing functional models for object recognition, and also a frame-based system for representing such models. The frame-based system can recognize objects using functional models and complete data, and can also hypothesize what is missing when applying functional models to incomplete data. Recognition and functional part labeling of a simple bed object was demonstrated, and extensions to the basic model were described that can model a wide range of objects that can function as beds.

I gratefully acknowledge Roddy Collins and Wendy Abbott for their help in preparing the recognition example for this paper.

References

DiMa89 DiManzo, M., Trucco, E., Giunchiglia, F., and Ricci, F. FUR: Understanding Functional Reasoning. *International Journal of Intelligent Systems* **4**, 1989, pp. 431-457.

Fike85 Fikes, R., and Kehler, T. The Role of Frame-based Representation in Reasoning. *Communications of the ACM* , **28**, 3, Sept. 1985, pp. 904-920.

Grim90 Grimson, W.E.L, *Object Recognition by Computer: The Role of Geometric Constraints*, MIT Press, 1990.

Mann94 Mann, R. and Jepson, A. 'Support' in Support of Vision, *IEEE Computer Society Workshop on the Role of Functionality in Object Recognition*, June 1994.

Moha92 R. Mohan and R. Nevatia, Perceptual Organization for Scene Segmentation and Description, *IEEE Transactions on PAMI*, **14**, 1992, pp. 616-635.

Mulg92 Mulgaonkar, P.G., Cowan, C.K., and DeCurtins, J. Understanding Object Configurations Using Range Images. *IEEE Transactions on PAMI* **14**, 2 , Feb. 1992, pp. 303-307.

Requ80 Requicha, A.A.G. Representations of Rigid Solids. *ACM Computing Surveys* **12**, 4, Dec. 1980, pp. 437-464.

Rivl94 Rivlin, E., Dickinson, S. and Rosenfeld, A., Recognition by Functional Parts, Technical Report CAR-TR-703, Center for Automation Research, University of Maryland, Feb. 1994.

Sand90 Sanderson, A.C., de Mel llo, L.S.H., and Zhang, H. Assembly Sequence Planning. *AI Magazine* **11**, 1, Spring,1990, pp. 62-81.

Star93 Stark, L., Hoover, A.W., Goldgof, D.B. and Bowyer, K.W., Function-based object recognition from incomplete knowledge of object shape. *IEEE Workshop on Qualitative Vision*, June 1993.

Star94 Stark, L. and Bowyer, K. Function-Based Generic Recognition for Multiple Object Categories. *CVGIP: Image Understanding* **59**, 1, Jan. 1994, pp. 1-21.

Thad94 Thadani, S., *Constructing Functional Models of a Device from its Structural Description*, Ph.D. Thesis, Department of Computer and Information Science, Ohio State University, 1994.

Walk88 Walker, E.L., Herman, M., and Kanade, T. A Framework for Representing and Reasoning about Three-dimensional Objects for Vision. *AI Magazine* **9**, 2, Summer 1988, pp. 47-58.

Walk89 Walker, E.L., *Frame-Based Geometric Reasoning for Construction and Maintenance of 3D World Models*, Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, 1989.

DERIVING COMPILED STATES FROM A PHYSICAL SYSTEM FUNCTIONING DESCRIPTION

Jean-Yves Djamen
Claude Frasson
Marc Kaltenbach

Université de Montréal
djamen@iro.umontreal.ca

Abstract

This paper shows how qualitative values assigned to a component in a physical system functional description can be transform to compiled states (i.e. states at an appropriate level of detail) in order to facilitate knowledge transfer in a tutoring context. *

1 INTRODUCTION

One of the aim of designing intelligent tutoring systems (ITSs) is to facilitate knowledge communication of a given domain, between a trainee (the human learner) and a trainer (simulated by the computer that incorporates pedagogical and domain knowledge of human trainers as well as mechanisms to communicate them to another human).

Some difficulties are inherent in building such systems. For example functional perspectives are still missing in the representation of the domain knowledge encoded in the computer or cannot be represented using known AI techniques.

We propose hence to view a physical sysem as a set of components with some well chosen *attributes* and their values. The state of a component is the current set of values of its attributes. Attributes are assigned to components with regards to their *roles* in the physical system. The role of a component consists of its *behavior* and its *function*. The behavior is described by its state changes. The function is described by effects of its state changes. The behavior of the whole physical system is defined as a set of state changes fostered by effects of human actions across components, and /or effects of state changes within components.

This paper makes a case for a particular way of defining component states to achieve what we call *compiled states* that prove particularly useful in producing comments for several reasoning tasks such as simulation, diagnosis, functional deduction, etc. [4]

In the following we show how compiled states can be automatically derived from the description of a physical system functioning. The paper first recalls the idea of quantity space (section 2), then shows the relationship between qualitative values and compiled states (section 3). It further insists on the process of automatically deriving compiled states from the description (or the modification) of a physical system functioning (section 4). It then concludes with an example (section 5).

2 QUANTITY SPACE AND QUALITATIVE VALUES

The quantity space is used as a standard feature in qualitative physics to simplify the modeling task [5]. This space, which has been defined primarily for a discrete representation of a continuous space [6], holds three qualitative values, mainly $+$, 0 and $-$, where a nonzero landmark can always be accommodated by defining a quantity $y = x - a$ whose zero is the value $x = a$ [3].

An extended approach considers the quantity space as a partially ordered set of landmarks values, where a quantity is described in term of its or-

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/359 ©1996 FLAIRS

dinal relations with the landmarks, hence allowing the description of qualitative distinctions such as increasing, decreasing, stable oscillation [7]. Another extension considers partial states in a quantity space as a subset of state variables in order to contribute to the causal explanations of some physical system mechanisms [2].

However, while describing a physical system functioning to facilitate knowledge communication between the computer and the human, the classical use of the $\{+, 0, -\}$ semantics may not facilitate tutoring activities such as assessment, student reasoning analysis, etc.

For instance, let us consider the following description (Cf. [8], page 114):

***Rule 1* : States of a battery and state changes**
Battery
States: Charged or Discharged.
If the battery is discharged and if it has a voltage applied to it, then it becomes charged; otherwise it remains discharged.
If the battery is charged and if there is a path with no resistive elements across the battery, then it becomes discharged; otherwise it remains charged.

This description incorporates both changes (i.e. the battery can change from *charged* to *discharged* and vice-versa) and changes effects (i.e. the values in *voltage* and *resistive* play a role in this change), hence rending difficult the communication of such rules to another human. On the other hand it will be difficult to handle new design prospects. For example adding other components can lead to some difficulties such as the scaling problem.

More generally, it is not always clear whether values assigned to components (e.g. the battery in Rule 1) will be useful for knowledge communication or they suffice to describe components' roles (e.g. the voltage and the resistive in Rule 1) vis-à-vis the functional mechanisms.

3 QUALITATIVE VALUES AND COMPILED STATES

A qualitative value assigned to a component is a compiled state if the behavior of the component can be described without a reference to another component [4] (principle 1). Before seeing how such a value can be derived from a description, let us define the following notions: Value Types, Component Current States, Events and Qualitative Directions.

Principle 1 : State changes
In describing the behavior of a component, state changes are listed and no external reference to another component must hold.

Values assigned to a component represent its *set of possible states* (SPS). Two types of values can be assigned to a component: enumerable and interval. *Enumerable type values* of a component are described by a set of states that can be listed and do not need to be classified as landmark. For instance 'red', 'yellow', 'green', etc. These values are used when there is no need to define infinitesimal states between two states v_1 and v_2. *Interval type values* of a component are described as within the $\{+, 0, -\}$ semantics.

At any moment, a component can take only one state from its SPS. This state is called its *current state* (CS).

State changes within a component take place with regard to *external events* (actions of a human across the component) or *internal events* (effects of some state changes) [4]. We will note $s_{\rightsquigarrow} v$, a signal produced by an event, and sent to a component that should take the value v as its new current state.

Qualitative directions are respectively defined for enumerable and interval type values as follows †:

Enumerable

$$dir(x, s_{\rightsquigarrow} v_i) = \begin{cases} stable & \equiv & CS(x) = v_i \\ different & \equiv & CS(x) \neq v_i \end{cases}$$

Interval

$$dir(x, s_{\rightsquigarrow} v_i) = \begin{cases} increasing & \equiv & CS(x) < v_i \\ stable & \equiv & CS(x) = v_i \\ decreasing & \equiv & CS(x) > v_i \end{cases}$$

4 DERIVING COMPILED STATES

Compiled states can automatically be derived from the description of state changes and their effects, by assigning maximum and minimum values to each component.

4.1 Assigning Maximum Values

Maximum values can be assigned to a component by applying Theorem 1.

Theorem 1 : Adding a compiled state
Let u be a component, with a set of assigned values $V = \{v_1, \ldots, v_i, \ldots, v_n\}$. If the description of the behavior of u violates the principle 1 with $CS(u) = v_i$, then v_i is an abstract state that must give at least two values, v_{i1} and v_{i2}. The new set $V^+ = \{v_1, \ldots, v_{i1}, v_{i2}, \ldots, v_n\}$ replaces V.

† $dir(x, s_{\rightsquigarrow} v_i)$ stands for qualitative direction of x while receiving the signal s holding the value v_i. $CS(x)$ stands for current state of x.

Given some components $x, y_1, \ldots, y_m$, the general form of a rule describing the behavior of x that violates the principle 1 is as shown in Rule 2, where a state change within a component is described along with some conditions that its environment must satisfy. (The environment is made of external and/or internal events.) This rule can be extended to more external (i.e. non x) components. Hence the violation of principle 1 in such a rule can mean reference to only one external component or several components.

Rule 2 : *if* $CS(x) = v_i$ *and* $s_{\rightsquigarrow} v_j$ *and* $CS(y) = w_a$ *then* $CS(x) = v_k$ $(v_i, v_j, v_k \in SPS(x))$

4.1.1 One External Component in Behavior

The external component described in Rule 2 may have enumerable or interval type values.

Enumerable Type Values: Two qualitative directions of x can be considered respectively when $CS(y) = w_a$ and $CS(y) = w_b$ $(w_a \neq w_b)$. The reference to $CS(y) = w_a$ in Rule 2 assumes that Rule 3 can logically be deduced when x receives the signal $s_{\rightsquigarrow} v_j$. (It is worth noting that if v_l was equal to v_k, then the reference to y should be superfluous.)

Rule 3 : *if* $CS(x) = v_i$ *and* $s_{\rightsquigarrow} v_j$ *and* $CS(y) = w_b$ *then* $CS(x) = v_l$ *(with* $v_l \neq v_k$*)*

Given Rule 2 and Rule 3, some behaviors of x can be defined. In occurrence, the current state of x changes from v_i to v_k or v_l, depending on the qualitative direction of y. If there is no other value assigned to x, then $SPS(x)$ should respectively be $\{v_i, v_k\}$ and $\{v_i, v_l\}$.

Foreseen cases, when x changes its current state are then listed as follows:

- if $v_k = v_j$ $(v_k \neq v_i)$, then $CS(y) = w_a$ has the effect of forcing the change in $CS(x)$.
- if $v_k \neq v_j$, then $v_k = v_i$ and $CS(y) = w_a$ has the effect of forcing a value preservation of $CS(x)$.
- if $v_l = v_j$ $(v_l \neq v_i)$, then $CS(y) = w_b$ has the effect of forcing a value change in $CS(x)$.
- if $v_l \neq v_j$, then $v_l = v_i$ and $CS(y) = w_a$ has the effect of forcing a value preservation of $CS(x)$.

All these changes can be represented by the following rules:

Rule 4 : *if* $CS(x) = v_i$ *and* $s_{\rightsquigarrow} v_j$ *then* $CS(x) = v_k$ $(v_k = v_j$ *or* $v_k = v_i)$

Rule 5 : *if* $CS(x) = v_i$ *and* $s_{\rightsquigarrow} v_j$ *then* $CS(x) = v_l$ $(v_l = v_j$ *or* $v_l = v_i)$

But $v_k \neq v_l$. So, two cases are possible. Either in Rule 4 and Rule 5, v_j is not the same, or rather, v_i is not the same. We have deduced Rule 3 on the basis of the receipt of the same signal. We have indeed the same v_j and the only difference is about v_i which must be specialized in accordance with the qualitative direction of y.

Both Rule 4 and Rule 5 can be specialized as follows ($v_i w_k$ stands for the current state of x being v_i, taking into account that an external component has an attribute value w_k):

Rule 6 : *if* $CS(x) = v_i w_a$ *and* $s_{\rightsquigarrow} v_j$ *then* $CS(x) = v_k$ $(v_k = v_j$ *or* $v_k = v_i w_a)$

Rule 7 : *if* $CS(x) = v_i w_b$ *and* $s_{\rightsquigarrow} v_j$ *then* $CS(x) = v_l$ $(v_l = v_j$ *or* $v_l = v_i w_b)$

As a component will react to a signal by changing or preserving its current state, those rules can be written respectively as follows:

Rule 8 :
if $CS(x) = v_i w_a$ *and* $s_{\rightsquigarrow} v_j$ *then* $CS(x) = v_k$ $(v_k = v_j$ *and* $v_k \neq v_i w_a)$
if $CS(x) = v_i w_a$ *and* $s_{\rightsquigarrow} v_j$ *then* $CS(x) = v_i w_a$

Rule 9 :
if $CS(x) = v_i w_b$ *and* $s_{\rightsquigarrow} v_j$ *then* $CS(x) = v_l$ $(v_l = v_j$ *and* $v_l \neq v_i w_b)$
if $CS(x) = v_i w_b$ *and* $s_{\rightsquigarrow} v_j$ *then* $CS(x) = v_i w_b$

With regard to the signal $s_{\rightsquigarrow} v_j$, the only possible combinations are described by the two following mutual exclusive rules:

Rule 10 :
if $CS(x) = v_i w_a$ *and* $s_{\rightsquigarrow} v_j$ *then* $CS(x) = v_k$ $(v_k = v_j)$
if $CS(x) = v_i w_b$ *and* $s_{\rightsquigarrow} v_j$ *then* $CS(x) = v_i w_b$

Rule 11 :
if $CS(x) = v_i w_b$ *and* $s_{\rightsquigarrow} v_j$ *then* $CS(x) = v_l$ $(v_l = v_j)$
if $CS(x) = v_i w_a$ *and* $s_{\rightsquigarrow} v_j$ *then* $CS(x) = v_i w_a$

Interval type values: Three current states can be considered according to three qualitative directions of y: $CS(y) = w_0$, $CS(y) = w_-$ $(w_- < w_a)$ and $CS(y) = w_+$ $(w_+ > w_a)$. Thus, the reference to $CS(y) = w_a$ within Rule 2 assumes that Rule 12 and Rule 13 can logically be deduced when x receives the signal $s_{\rightsquigarrow} v_j$. (It is worth noting that in the best case, we have rules described in the previous section, and in the worst case, more values are yet to be discovered).

Rule 12 : *if* $CS(x) = v_i$ *and* $s_{\rightsquigarrow} v_j$ *and* $CS(y) < w_b$ *then* $CS(x) = v_l$ $(v_l \neq v_k)$

Rule 13 : *if* $CS(x) = v_i$ *and* $s \rightsquigarrow v_j$ *and* $CS(y) > w_b$ *then* $CS(x) = v_p$ $(v_p \neq v_k$ *and* $v_p \neq v_l)$

Each of the above rules is then developed as in the previous section.

4.1.2 Several External Components in Behavior

Rule 2 can be (re)written as follows:

Rule 14 : *if* $CS(x) = v_i$ *and* $s \rightsquigarrow v_j$ *and* $CS(y_1) = w_{11} \ldots CS(y_m) = w_{mi}$ *then* $Cs(x) = v_k$

Values of external components can be *enumerable* or *interval*, or a mixture of both types. In the worst case, value changes of x are defined by the set of vectors described by the current states of external components. The form of this set is as follows (by noting that each external component y_j has three qualitative directions $CS(y_j) = w_{j-}$, $CS(y_j) = w_{j0}$ and $CS(y_j) = w_{j+}$):

$$
\begin{aligned}
&CS(y_1) = w_{1-} \ldots CS(y_m) = w_{m-},\\
&CS(y_1) = w_{10} \ldots CS(y_m) = w_{m-},\\
&CS(y_1) = w_{1+} \ldots CS(y_m) = w_{m-},\\
&\vdots\\
&CS(y_1) = w_{1+} \ldots CS(y_m) = w_{m+}
\end{aligned}
$$

On the basis of the formed set of vectors, it becomes possible to define behavior rules of x as shown in the following general rule, in which compiled states can be generated as in previous cases (* represents 0, - or +):

Rule 15 : *if* $CS(x) = v_i w_{1*} \ldots w_{m*}$ *and* $s \rightsquigarrow v_j$ *then* $CS(x) = v_k$ $(v_k = v_j$ *or* $v_k = v_i w_{1*} \ldots w_{m*})$

4.2 Assigning minimum values

Minimum values can be assigned to a component by applying Theorem 2. In fact, the new set of possible states of a component is only made of values describing state changes, along with values described in all signals that the component can receive [4]).

Theorem 2 : Suppressing a state

Let u be a component, with a set of assigned values $V = \{v_1, \ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_n\}$. If v_i is not used neither in its behavior nor in its function, then v_i must be removed from V, and the new set $V^- = \{v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n\}$ replaces V.

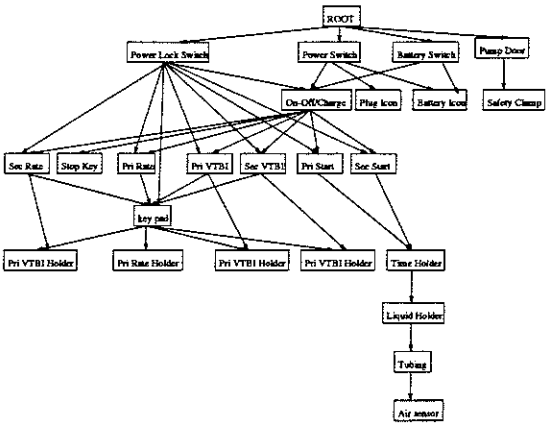


Figure 1: Baxter pump's functional dependency graph.

5 AN EXAMPLE

Let us consider a part of the Baxter pump functioning [1] described below:

...By default, the pump is 'Plugged In' ("Plug Icon" appears in the message display component, and the battery is charging) or else operates on battery power ("Battery Icon" appears). The "On-Off/Charge" (OOC) key turns the pump 'On' and 'Off'. The internal battery charger remains 'On' regardless of the "On-Off/Charge" key as long as the pump is 'Plugged In'. When the pump is 'On', the keys "Pri Rate" and "Pri VTBI" allow programming respectively the primary rate and the primary volume. The "Numerical Keypad" can then be used...

5.1 Some Components and their Roles

The above description enables some specific tasks such as "Making a primary infusion with rate 250ml/h and volume 500ml". Main components used by the operator in such a task are "On-Off/Charge", "Pri Rate", "Pri VTBI", "Start", "Key 0", "Key 2" "Key 5", which are described as sensitive buttons in the operator's manual. But other components, such as "Pri Rate Holder", "Pri VTBI Holder" and "Power Switch" (PS), are also important. The figure 1 shows functional relations between the Baxter pump constituents. In this figure, rectangles represent Baxter pump constituents that are useful for the current functioning description. An arrow between x and y means that a state change in x can affect the current state of y.

Let us assumed the following description of some state changes and their effects:

Rule 16 **: Some state changes**
if CS(OOC) = Off and s⇝On and CS(PS) = On then CS(OOC) = On
if CS(OOC) = On and s⇝Off then CS(OOC) = Off
if CS(Pri rate) = Off and s⇝On and CS(OOC) = On then CS(Pri rate) = On
if CS(Key 2) = Off and s⇝On and (CS(Pri Rate) = On or CS(Pri VTBI) = On) then CS(Pri rate) = On

Rule 17 **: Some effects of some state changes**
if "PS" changes from 'Off' to 'On' then "OOC" is able to receive an action (a_r_a), otherwise (i.e. if "Power Switch" is not "on") "OOC" is not a_r_a
if "OOC" changes from 'Off' to 'On' then "Pri Rate" and "Pri VTBI" are a_r_a, otherwise (i.e. if the current state of "OOC" is not 'On') they are not a_r_a
if "Key 2" changes from 'Off' to 'On' and if "Pri Rate" is 'On' then "Pri Rate Holder" must hold its old value concatenated with digit '2'
if "Pri Rate" is 'On' then "Pri VTBI" must be 'Off'

Let us also consider the initial values of "PS", "OOC", "Pri Rate", "Pri VTBI", "Key 0", "Key 2" and "Key 5" to be 'off'; and those of "Pri Rate Holder", "Pri VTBI Holder" to be '0'.

5.2 Assigning minimum and maximum values

Let us focus on one description that violates the principle 1.

The following rule is used to described how "OOC" can change from 'Off' to 'On' (values type of "OOC" and "PS" are *enumerable*):

Rule 18 : *if CS(OOC) = Off and s⇝On and CS(PS) = On, then CS(OOC) = On*

As seen above, Rule 19 can be deduced according to the qualitative direction of "Power Switch".

Rule 19 : *if CS(OOC) = Off and s⇝On and CS(PS) = Off, then CS(OOC) = Off*

But in Rule 19 "Off" before and "Off" after the environment of "OOC" (the signal and "PS" direction) are different. So that when "PS" is 'Off', the 'Off' of "OOC" means 'Not Activable' (i.e. "OOC" is not *a_r_a*); and when "PS" is 'On', it means 'Activable' (i.e. "OOC" is *a_r_a*).

The abstract state 'Off' [4] will then lead to two compiled states 'OffOn' ('Activable') and 'OffOff' ('Not Activable'), in accordance with the following rules:

Rule 20 :
if CS(OOC) = Activable and s⇝On, then CS(OOC) = On
if CS(OOC) = Not Activable and s⇝On, then CS(OOC) = Not Activable

Possible states of "OOC" are updated from { Off, On } to { Activable, Not Activable, On }. In this context, 'On' is already a compiled state and does not need to be modified (Cf. Rule 21).

Rule 21 *if CS(OOC) = On and s⇝Activable, then CS(OOC) = Activable*

Acknowledgments

This work is done as part of the SAFARI project from a grant of the Quebec Ministry of Industry, Trade, Science and Technology.

References

[1] Baxter. *Operator's Manual, Flo-Gard 6201, Volumetric Infusion Pump.* Baxter Healthcare Corporation, Deerfield, IL 60015 USA, 1992.

[2] B. Chandrasekaran. Functional representation and causal processus. *Advances in computer*, 38:73–143, 1993.

[3] Johan de Kleer and J. S. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–84, 1984.

[4] Jean-Yves Djamen, Marc Kaltenbach, and Claude Frasson. Qualitative comments with physical systems for intelligent tutoring systems. In *Proc. of the Eighth Florida Artificial Intelligence Research Symposium*, pages 304–308, Melbourne Beach, FL., April 1995.

[5] Kenneth D. Forbus. Qualitative physics: Past, present and future. In Daniel S. Weld and Johan de Kleer, editors, *Qualitative reasoning about physical systems*, pages 11–39. Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[6] P. J. Hayes. The naive physics manifesto. In D. Michie, editor, *Expert system in the microelectronic age.* Edinburgh University Press, 1979.

[7] Benjamin Kuipers. Commonsence reasoning about causality: Deriving behavior from structure. In D. G. Bobrow, editor, *Qualitative reasoning about physical systems*, pages 169–204. The MIT Press, Cambridge, MA., 1985.

[8] Barbara Y. White and John R. Frederiksen. Causal model progressions as a foundation for intelligent learning environments. *Artificial Intelligence*, 42:99–157, 1990.

MEASUREMENT INTERPRETATION OF MEDICAL LABORATORY DATA BASED ON RS-QUAD

Shusaku Tsumoto and Hiroshi Tanaka
Department of Information Medicine
Medical Research Institute,Tokyo Medical and Dental University
tsumoto.com@tmd.ac.jp, tanaka@cim.tmd.ac.jp

Abstract

Understanding medical laboratory data is equivalent to understanding the relation between structure and function of biological systems, which is one of the most important problem of functional reasoning. In this paper, based on these two methods, QUAD and ACAM, we introduce a measurement interpretation system, called MIS-QUAD (Measurement Interpretation System based on RS-QUAD), to interpret results of laboratory examinations chronologically and to reason the hierarchical level which explains the derived information most plausibly.

1 INTRODUCTION

Understanding medical laboratory data is equivalent to understanding the relation between anatomical structure, such as a kidney and function of biological systems, such as a renal function, which is one of the most important problem of functional reasoning. The difficult problem is that biological systems are composed of several kinds of hierarchical systems, where the interaction of functional units is indispensable to maintain their functions.

Especially, in medical diagnosis, abnormality of laboratory examinations can be viewed as the breakdown of maintenance of biological functions. This is why the ordinary methodologies[3, 6] are not sufficient to understand the behavior and function of complex biological systems.

In the former paper[5], we focus on how to represent a hierarchy in biological function, especially in medical domain. Then we introduce the previously reported methodology, qualitative dynamics of diseases' progression(QUAD), and a simulation method which complements the above reasoning, augmented cell automaton model(ACAM) to bridging between hierarchies. As a result, this approach enables us to describe temporal progression of disease mechanisms and to make more natural causal explanation about clinical manifestations.

In this paper, based on these two methods, QUAD and ACAM, we introduce a measurement interpretation system, called MIS-QUAD (Measurement Interpretation System based on RS-QUAD), to interpret results of laboratory examinations chronologically and to reason the hierarchical level which explains the derived information most plausibly.

The paper is organized as follows: Section 2 illustrates a biological function. Section 3 discusses how medical experts interpret medical data. Section 4 presents representation of biological functions. Section 5 shows briefly how MIS-QUAD works and Section 6 discusses related work. Finally, Section 7 concludes this paper.

2 BIOLOGICAL FUNCTION

When we model functions in living things, we have to consider the following four characteristics: multiple-level-hierarchies, flow of information between hierarchies, description of interactions at the same hierarchical level, and time-scale of the same hierarchical level. In this paper, we specially focus on the former two characteristics and illustrate them by using renal function as an example. Medical functions are almost always hierarchical. For example, let us consider the function of a kidney (Fig. 1). It is well known that a kidney functions as a filter that removes the waste from the blood. If readers could not image its function, readers could regard a kidney as a chemical filtering device composed of chemical absorbent.

As like a filtering device, a kidney consists of millions of functional unit, called a nephron, which

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/364 ©1996 FLAIRS

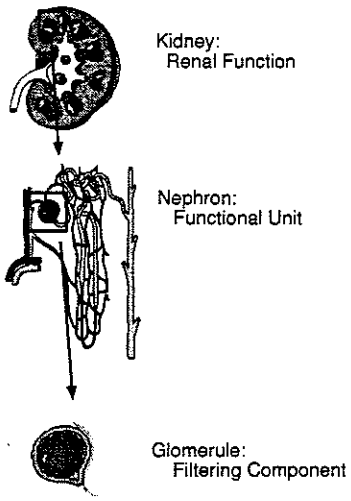


Figure 1: Morphological Hierarchy in Renal Function

is composed of several different subunits, such as glomerular, proximal tube, and distal tube. Glomerular functions as a filter which removes all kinds of small chemicals, such as albumin. However, it removes not only the wastes but also the substances are needed for maintenance of the living things. The proximal tube reabsorbs such chemicals. And distal tube reabsorbs water, which is regulated by the water regulation mechanism of living things.

Interestingly, those subunits consists of a large number of functional sub-subunits. For example, glomerular consists of hundreds of glomerular cells. Then, each cell consists of different components, such as nucleus, cytoplasm, ribosome.

As shown in the above description, the following two characteristics can be observed. First, renal function is hierarchical. This issue is also precisely discussed in [5]. Second, there are two types of connection between a lower level and an upper level in the hierarchy. One is that an upper functional unit consists of different functional subunits, such as a nephron. We refer to this type as *heterogeneous bridging*. The other is that an upper functional unit consists of many functional subunits, such as a kidney or a glomerule. We refer to this type as *homogeneous bridging*. Based on these concepts, renal function can be represented as shown in Fig. 2.

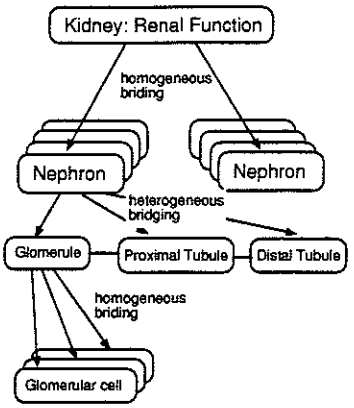


Figure 2: Hierarchical Representation in Renal Function

It is notable that these morphological hierarchies are closely related with measurement interpretation of medical experts. As discussed below, they firstly interprets medical data in an organ function level(Kidney). Then, medical experts reason how the lower level components, nephrons are damaged under the constraints generated by the upper level (Kidney). Finally, they predict the severity of the components of a nephron.

For example, in a case when a patient has a proteinuria, medical experts will diagnose that substantial parts of nephrons are damaged. Then, they imagine that the proximal tube of nephrons, which functions as reabsorption of albumin, will be damaged. It is also notable that these reasoning sequences are rather spatial reasoning than temporal reasoning. Thus, we call these procedures **local measurement interpretation**.

3 HOW TO INTERPRET DATA

3.1 Medical Expert's Interpretation

When medical experts meet clinical data of a patient, they interpret them in the following way. If they do not have any *a priori* information on the patient, then medical physicians interpret medical data, diagnoses the functional damage, and selects several diagnostic candidates under the uncertainty. After the interpretation, they follow the clinical courses of the data and gather information to remove the uncertainty. Thus, medical experts firstly interprets data locally, and interprets them

more globally after getting more information. Otherwise, medical experts interpret medical data by using the information about the patient both locally and globally. For example, let us consider a case when a medical expert finds that the serum creatine of a patient is a little higher than normal. If the expert does not know that he suffers from DM, a medical expert follows the clinical course of serum creatine and other medical data. Their decision is firstly made locally and then made globally. Otherwise, if the expert knows it, then a medical expert diagnoses that he suffers from DM nephropathy and that the functional damage is much severer, since the arterioles of glomerules are damaged. In this case, medical experts generate the scenario of its clinical course, which can be viewed as global knowledge, and then they interpret data at each time locally, although their reasoning is based on global knowledge. If the laboratory examinations violates the generated scenario, medical experts reconstruct the scenario.

Thus, in order to interpret medical function, a past history is one of the principal factors, or one of the important constraints for measurement interpretation, as global knowledge.

3.2 Generation of Context

In order to generate the context, MIS-QUAD introduces two kinds of inputs. One is the past context generated by MIS-QUAD, and the other one is a set of laboratory examinations or physical examinations obtained. Using these two inputs, MIS-QUAD applies "Rules for Context" to summarize the context, to which we refer as a **local measurement interpretation** procedure.

For example, "Rules for Context" for DM nephropathy in the organ-function-level are defined as:

$$[PastHistory = DM]\&[1.1 < Creatine < 2.0] \rightarrow [Status : 3.0 < Creatine < 4.0 in Normal]$$
$$[PastHistory = DM]\&[Creatine > 2.0] \rightarrow [Status : Creatine > 4.0 in Normal]$$

The above rule means that a DM patient whose serum creatine is between 1.1 and 2.0 should be treated as a non-DM patient whose creatine is between 3.0 and 4.0. From this consequence, the following rule about "3.0 ¡ Creatine ¡ 4.0" will be applied.

$$[Status : 3.0 < Creatine < 4.0 in Normal] \rightarrow [MildRenalFailure]$$
$$[Status : 3.0 < Creatine < 4.0 in Normal] \rightarrow [Cell - level : GlomerularDamage : Severe]$$

So, in the case when the serum creatine of a DM patient is 1.3, this status is evaluated as a case of mild renal failure, but in the cell-function level, the damage of glomerular units may be severe. This rule makes a constraint for the lower-level, cell-function-level. After the evaluation of all the rule, the interpretation procedure proceeds into the lower-level, that is cell-function-level.

4 REPRESENTATION

4.1 Symptomatological Level

Representation of the water regulation (symptomatological level) is represented as in Fig. 3, respectively.

The second attribute, denoted by "as Primitives", gives information on what kind of primitives will be in the upper level for this function. The third one shows what kind of primitives construct the function. In the case of heterogeneous bridging, several components are described. On the other hand, only one component is described in the case of homogeneous bridging. And the fourth to the eighth attribute provide knowledge needed for simulation, such as kind of activity, damage, and compensation. In addition to those attributes, we introduce the ninth attribute "Context" in order to incorporate measure interpretation. In the context, path history of the patient will be input in this attribute, as shown later. This attribute serves not only as a constraint to evaluate the values of examinations, but also as a constraint to reason the status of the lower hierarchical level. For example, "Diabetes Mellitus (20 years, poor control)" means that a patient suffers from Diabetes Mellitus(DM) for 20 years, but that the therapy of a patient does not go well. This context suggests that renal function should be evaluated very carefully: that is, even if the creatine value, a marker for renal function (organ function level), is a little abnormal, the function of the patient is more damaged than that of a patient who does not suffer from DM. This context also gives a constraint that the lower level (cell-function level), nephrons, suffer from arteriosclerosis of each vessel, which is characteristic to the DM nephropathy. The final, 10th attribute

Function:	Water Regulation
	(Symptomatological Level)
as Primitives:	Filter
	(Symptomatological Level)
Primitives:	Renal function,
	Circulation
Numbers:	One for Each
Activity:	Filter function
Damage:	Renal Function
Time Scale:	Weeks to months
Compensation:	Connective Tissues
:	Repair
Dynamics:	Acute
Context:	Diabetes Mellitus
	(20 years, Poor Control)
	/* History */
Index:	Edema, Auria
Diagnostic Rules:....	
Rules for Context:....	

Figure 3: Representation of Water Regulation

Function:	Renal function
	(Organ-Function Level)
as Primitives:	Filter
	(Symptomatological Level)
Primitives:	nephrons
Numbers:	Unknown (>1,000,000)
Activity:	Filter function
Damage:	Immunological Reaction
Time Scale:	Weeks to months
Compensation:	Hyperfunction
:	Repair
Dynamics:	Acute
Context:	Diabetes Mellitus
	(20 years, Poor Control)
	/* History */
Index:	Serum Creatine
Diagnostic Rules:.....	
Rules for Context:.....	

Figure 4: Representation of Renal Function

shows what kind of examination will serve as the marker of a function. In the above example, several manifestations, such as edema is used as the marker of renal function.

4.2 Organ-function Level

Representation of renal function (organ-function level) for QUAD is represented as in Fig. 3 and Fig. 4, respectively.

In addition to those attributes, we introduce the ninth attribute "Context" in order to incorporate measure interpretation. In the context, path history of the patient will be input in this attribute, as shown later. This attribute serves not only as a constraint to evaluate the values of examinations, but also as a constraint to reason the status of the lower hierarchical level. For example, "Diabetes Mellitus (20 years, poor control)" means that a patient suffers from Diabetes Mellitus(DM) for 20 years, but that the therapy of a patient does not go well. This context suggests that the value of creatine should be evaluated very carefully: that is, even if the creatine value is a little abnormal, the renal function of the patient is more damaged than that of a patient who does not suffer from DM. This context also gives a constraint that the lower level, nephrons, suffer from arteriosclerosis of each vessel, which is characteristic to the DM nephropathy. The final, 10th attribute shows what kind of examination will serve as the marker of a function. In the above example, serum creatine is used as the marker of renal function.

5 MIS-QUAD

Using the formerly introduced two methods QUAD and ACAM[5], we introduce a measurement interpretation system, called MIS-QUAD (Measurement Interpretation System based on RS-QUAD), to interpret results of laboratory examinations chronologically and to reason the hierarchical level which explains the derived information most. In this system, QUAD and ACAM serve as generators of global knowledge. That is, these methods are used to interpret the chronological course of each function, which we refer to **global measurement interpretation**. Before global interpretation, MIS-QUAD evokes a **local measurement interpretation** procedure, which interprets the status of function at a certain time.

MIS-QUAD is developed under the object-oriented scheme. Each function is defined as a class, and the three-level hierarchy is defined as a subclass of this function. For example, renal function

is defined as a class, and the components of renal function, nephrons are defined as a subclass. As to heterogeneous and homogeneous bridging, MIS-QUAD provides a meta object of symptomatological level.

In order to implement temporal aspects of local measurement interpretation, each state is defined as an instance of an object. When an instance is generated, rules for context are applied, and the context will be calculated. After the calculus, the results are propagated into the subclass, and an instance of the subclass will be generated.

In the case when a past history is not given, the above local interpretation procedure will continue for several instances. After several instances generate, a meta-object interprets the clinical course of measurements and evokes QUAD and ACAM, and generates global knowledge.

6 RELATED WORK

In the literature of artificial intelligence in medicine, the idea of Coiera's qualitative disease histories(QDH)[2] is closely related with that of MIS-QUAD. He points out clearly that deep representations take a computational cost and that a intermediate representation between shallow and deep saves this cost. Furthermore, he proposes that QDHs are good intermediate representations lying between shallow disease patterns and deeper qualitative models. He also classifies multi-level systems into two major categories, hierarchies based on variable granularity and ones based on variable representation type.

The idea of QDH is very similar to our interpretation model, local interpretation and global interpretation. QDH bridges between local model and global model, which can also be viewed as the qualitative scenario.

The main differences between Coiera's approach and ours are the following two points. First, while his approach is oriented to variable representation type, ours is to granularity. His deeper model is based on qualitative model using the framework of QSIM and his qualitative superposition[1].

So, our model corresponds to his QDH model based on a granularity model. Concerning with this granularity based system, Coiera[2] discusses that one of the problem of hierarchies based on granularity is information lost from bottom to top, and this is exactly what we mentioned in Section 2. In order to recover this information lost, we introduce QUAD and ACAM[5]. And we think that these two methods are sufficient to resolve the incompleteness of hierarchical systems based on granularity.

7 CONCLUSION

In this paper, based on these two methods, QUAD and ACAM, we introduce a measurement interpretation system, called MIS-QUAD (Measurement Interpretation System based on RS-QUAD), to interpret results of laboratory examinations chronologically and to reason the hierarchical level which explains the derived information most plausibly.

References

[1] Coiera, E.W. Monitoring diseases with empirical and model-generated histories. *Artificial Intelligence in Med.* **2**, 1990, 135-147.

[2] Coiera, E.W. Intermediate depth representations. *Artificial Intelligence in Med.*, 1992, **4**, 431-445.

[3] Buchnan, B.G. and Shortliffe, E.H.(eds.) *Rule-Based Expert Systems*, Addison-Wesley, MA, 1984.

[4] Sticklen, J. and Bond, W.E.(eds.) A special issue on Functional Reasoning and Functional Modeling. *IEEE expert*, **6**, 1991.

[5] Tsumoto, S., and Tanaka, H. Functional Reasoning based on Morphological Hierarchies and Qualitative Dynamics of Diseases' Progression, Proceedings of Workshop on Reasoning and Representation about Device Function, AAAI Technical Report Series, pp.143-153, AAAI Press, CA, 1994.

[6] Weld, D. and de Kleer, J.(eds.) *Readings in Qualitative Reasoning*, Morgan Kaufmann, CA, 1994.

Extending Functional Models For Non-Local Adaptations In Engineering Device Design

Sattiraju Prabhakar
School of Computing Sciences
University of Technology, Sydney
Email: prabhakar@socs.uts.edu.au

Ashok Goel
College of Computing
Georgia Institute of Technology
Email: goel@cc.gatech.edu

Abstract

In complex engineering devices, often, several levels of interactions are present among different structural elements. Many functional models are limited to capturing the local interactions among structural elements. We extend the Structure, Behaviour and Function (SBF) model to capture these non-local interactions by incorporating two kinds of teleological relationships. The first one represents the roles of subdomains within a device in giving rise to its function. The second one is the interactions among the models of subdomains in giving rise the function of the device. Our SBF model is uniform across several subdomains of a device. We present a computational process that adapts the extended SBF models to design new devices with multiple domains. This computational theory has been tested on a number of engineering examples.

1 INTRODUCTION

Adapting models of engineering devices to new design problems requires localisation of interaction between structural elements so that these interactions can be reused in new contexts. Many functional models support localisation of such interactions and hence adaptation is possible. But in many complex devices, non-local interactions in between the structural elements may be present. Majority of the work in functional modelling has focussed on modelling local interactions and very little work has been done on modelling non-local interactions for adaptation. In this paper we present a functional model which is an extension of Structure, Behaviour and Function (SBF) model (Goel 1991) that addresses this issue for the design of a class of complex devices known as interactive devices (IDs). In the interactive devices, the device and its environment interact to give rise to the functionality of the device.

1.1 Adaptation in Conceptual Design

In conceptual design, the design problem is specified in terms of the function or purpose of the device and the solution is described in terms of the structure of the device (Gero 1990, Goel 1991). Conceptual design of devices is a complex task and reasoning from first principles cannot often provide a viable solution.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/369 ©1996 FLAIRS

Our solution is adapting the known functional model of a device to address a new design problem. That is, from a model of a device which describes the relationship between the structure and function of a device by teleological relations, we arrive at a functional model of a new device. This adaptation requires design knowledge of a known device model be transferable to new context of new design problem (Goel 1991, Prabhakar and Goel 1992, Bhatta et al 1994).

One of the major aspects of context is the interactions between structural components within a device model. These interactions capture the changes within the device which form its behaviour and hence the functionality of the device. Adaptation of such a model is possible if the known interactions guide the designer to arrive at the new interactions. These new interactions may come into existence due to introduction of new structural elements. For example, to design a sulfuric acid cooler from a nitric acid cooler model, the interactions that are present between nitric acid and other structural elements need to be transferable to sulfuric acid and other structural elements in the new design.

In order to perform adaptation three issues need to be addressed - *identification* of structural element interactions that are responsible in giving rise to some functional aspects, *modification* of those interactions to meet the new functional aspects required in the new design problem, and *instantiation* of the new structural interactions into the rest of the structure of the device. One computationally viable approach to transfer these interactions to new contexts is to localise functional aspects of a device to a few interactions among small number of structural elements (Goel 1991). The instantiation is also made simpler due to small number of new interactions possible. The functional model should allow predicting these new interactions. By making modifications to local structural aspects it is possible to arrive at new structural descriptions which can realise new functions (Goel 1991). Since interactions among structural elements is captured within the behaviour, this localisation is captured within behaviour.

The Structure, Behaviour and Function (SBF) model captures these interactions between the structural elements as finite states (Goel 199)]. The changes in between the states is captured as a causal network. This acyclical graph of states allows organisation of interactions in a linear fashion. This model of interactions allow incremental local modifications to be performed on the behaviour to arrive at the structure of a new device. This adaptation is limited to models which represent local interactions.

1.2 Non-Local Interactions in Devices

Many functional models allow localisation of functional aspects onto local structural interactions. But several physical systems can have non-local interactions, as illustrated in figure 1, that are required for the function of the device. One example is nitric acid cooler (Goel and Prabhakar 1994) where the nitric acid is cooled by interacting with the cold water. The behaviour that delivers the functionality of the cooler can be represented using flow representations of nitric acid and water. The important point in modelling this device is that two flows interact resulting in the cooling down of nitric acid. The interactions between two adjacent states is not a simple causal interaction. For example, states *a* and *b* can interact through states *c* and *d*. Several physical systems show such non-local interactions. Some examples include, Automated Teller Machine (ATM), refrigerator, air-conditioners, airplanes, etc.

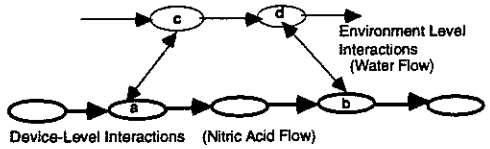


Figure 1. Non-local Interactions within a Device.

1.3 Adaptation of Non-Local Interactions

It is possible to model some of the non-local interactions as local interactions as illustrated in figure 2 for nitric acid cooler.

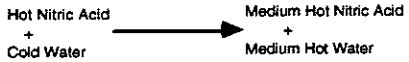


Figure 2. Non-Local Interactions as Local Interactions

This would require explicating the interaction between the substances and losing the information about their individual flows, though it is implicitly assumed. The liquids do not physically touch each other. A problem arises while adapting this behaviour using another substance that can interact with nitric acid and alters its properties. For example, a base can chemically interact with nitric acid. Still another problem is adapting the behaviour in figure 2 to a coolant which does not flow. The above behaviour gives very little information as how to change the flow of nitric acid. In other words, the model shown in figure 2 is not transferable to several new contexts. Further, the water and nitric acid flows may be playing different functional roles, for example water as a coolant and nitric acid as a heat source. Preserving their roles in a functional model can be important for adaptation.

1.4 Environments and Devices

A class of devices, called Interactive Devices, usually show the non-local interactions. The interactive devices are the engineering devices which get their functionalities by interacting with their environments[1].

Some of the examples of interactive devices are ATMs, Refrigerators and Airplanes. Most of the engineering devices can be categorised into three groups depending upon the extent of interaction a device has with its environment. A device is a Low Interaction Device (LID) if its interaction with its environment is limited only to input and output and there are no further interactions. Examples of LIDs are electrical buzzer-circuit, electrical light bulb circuit, etc. In Interactive Devices or Medium Interaction Devices (MIDs), the environment is reactive when interacting with the device to satisfy some goals. An example of an interactive device is an ATM where the environment is the user with her goals, operators and beliefs, whereas the machine has the operators that can perform on a database of bank accounts. Here several interactions can happen between the device and its environment after the initial interaction at input. But the environment and the device play definite roles. In High Interaction Devices (HIDs), the interaction between the device and environment is so complex that it is not possible to discern their roles in the function of the device. Examples of HIDs are distributed computer systems. Many engineering devices we encounter are or can be usefully modelled as MIDs.

A modelling aspect that characterises Interactive Devices is non-local interactions among the structural elements. For example, in an ATM, two databases can interact with each other using database operations. They can also interact with each other through the operators of the user. This non-local interaction is often caused by the environment. The adaptation task is common among interactive devices as they are supposed operate within several environments. Since the engineering devices need to work in large number of environments, the designer's knowledge of the environments is incomplete and the environments are large in number, the adaptation task for different environments is a significant task. Any technique that allows us to transfer knowledge across different environments is useful.

Our solution for this problem has two parts. We propose an extension of SBF model that represents the roles of each subdomain of the device, the teleological representation of each subdomain addresses not only how the roles are satisfied but also how a subdomain interacts with the other subdomain. Then we present an adaptation strategy that can transfer knowledge from such a model to a new model where the subdomains can be different. We present our model and strategy through an example of designing air-conditioner using refrigerator.

2 MODEL FOR NON-LOCAL INTERACTIONS

The model we suggest that enables transfer of knowledge for non-local adaptations has the following features.

1. The model constitutes several behavioural segments which explicate two kinds of teleological relationships between the structure and function.
2. The first kind of teleological relationship is from a set of structural elements to the role they play in the function of the interactive device. These roles are purposes of the subdomain represented by the set of

[1] The *environment* can be defined as having exogenous variables while the device as having the endogenous variables. Our approach is applicable to devices which can be characterised as having multiple domains where each domain has a distinct set of behavioural principles. We will call each of the device and environment as a subdomain to make our ideas uniformly applicable to the case of multi domain devices.

structural elements. These roles are called here as behavioural abstractions. The aggregation of roles of different behavioural segments leads to the function of the interactive device as illustrated in figure 3. Each of the behavioural segment is represented using a flow representation, and may incorporate different behavioural principles and may have states represented at different grain sizes.

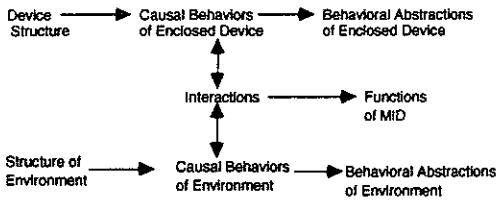


Figure 3. Characterisation of Interactive Devices.

3. The second kind of teleological relationship is structure to the interaction with the other behavioural segments. The interaction behaviour ties up the differences in behavioural principles and grain sizes of the behavioural states to give rise to the function of the interactive device, as illustrated in figure 3.

We use te Structure, Behaviour and Function (SBF) model developed in Goel (1991), to model the behavioural segments, their behavioural abstractions and the function.

2.1 Example: *Refrigerator*

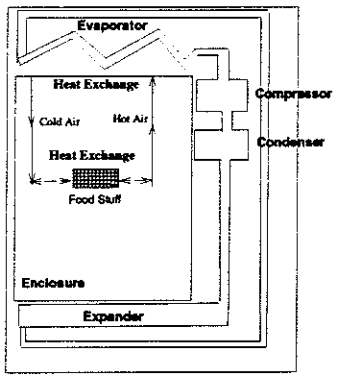


Figure 4. A Schematic of Refrigerator

The refrigerator has the function of keeping food inside a refrigeration enclosure at a constant low temperature. The refrigerator constantly removes heat from the enclosure so that the heat produced by the food items within the enclosure is soon dissipated. Figure 4 illustrates the design of a refrigerator. We consider the enclosure of the refrigerator as its environment and the machines constitute the device. The enclosure not only is made up of exogenous variables as most environments do, but also by considering the enclosure as an environment it can be used for a large number of adaptations.

The design of the refrigerator results in a coordination between the processes in the device and the environments so that the heat exchange between them is continuous and the food remains at a constant low temperature. The device has the behavioural abstraction of continuously removing heat and thus maintaining a low temperature. The environment, which includes both the refrigeration enclosure and the food items in it, has the behavioural abstraction of continuously transporting heat from the food items (the heat sources) to the surface of the device.

Heat is transported by the air currents in the enclosure. The air near the food gains heat from the food items, thus making the food cooler. The hot air rises towards the ceiling of the enclosure as it is lighter than the cooler air occupying same volume of space. At the ceiling, the hotter air exchanges heat with the surface of the device, and becomes cooler. The cooler air comes down from the ceiling by being pushed down by the upward moving hotter air. On its way, it encounters the food items. If the food is hotter than the air, then, again, heat exchange is possible. And, if and when heat is exchanged between the food and the air near it, the above process of air circulation repeats. Note that the setting up of the air currents in the environment is critical to the functioning of the refrigerator.

Now consider the problem of designing a device to keep a room cool - this the function of the air conditioner. Given the design of the refrigerator, and a model of how it works, the issue becomes how to adapt the design of the refrigerator to achieve the new functionality of keeping a room cool. Note that the important difference between the functionality delivered by the given design of the refrigerator and new functionality desired of it lies in the environment. Since the room is much bigger than the refrigeration enclosure, and since the room can potentially contain many different kinds of heat sources rather than just food items, the new environment is more complex than the old one. But the design of the refrigerator suggests a potential direction for finding the solution: modify the heat transport processes in the environment of the refrigerator to fit the new design problem. So the issue becomes how to adapt the behaviours of the environment of the refrigerator. We return to this issue later in the paper.

2.2 SBF Model Of Refrigerator

The function of the refrigerator is given below.

Given:
Food stuff in enclosure is at temperature T1
Makes:
Food stuff in enclosure is at temperature T2

Here the function transforms a behavioural state, the food stuff is at temperature T1, to another behavioural state where the food stuff is at temperature T2, where T2 is the temperature of the enclosure and T2 < T1. The temperature of the enclosure is constant. The temperature of the food stuff will finally be T2 whatever may its initial temperature. It is likely that the food stuff temperature may vary for a short duration due to heat introduced from outside into the enclosure. But then it will revert back to temperature T2.

The behavioural abstraction for environment of refrigerator is:

Given:
Food stuff inside enclosure is at temperature T1
Makes:
Air at the top of enclosure is at temperature T3

The purpose of the enclosure, as modelled for the refrigerator MID is to transport heat from the heat source

(food stuff) to the interaction point with the device. Here the beginning state is that the food stuff is inside the enclosure at temperature T1. This state finally results in the hot air current, with temperature T3 to be at the top of enclosure. This state transition has come about by heat being exchanged between the food stuff and the air current.

The behavioural abstraction for device is:

Given:
Refrigerant inside evaporator is at temperature T4
Makes:
Refrigerant inside evaporator is at temperature T4

The purpose of the refrigerator device is to absorb all the heat from the enclosure. The refrigerator device interacts with the external environment through one of its components : the evaporator. The evaporator is maintained at low temperature of T4 using a refrigerant. The initial temperature of the refrigerant is T4. The final temperature of the refrigerant in evaporator is T4. Because it is located near the enclosure, which can be hotter, the temperature of the refrigerant can go up, but returns back to T4. The behaviour of the refrigerator device is to bring back the temperature of the evaporator to T4.

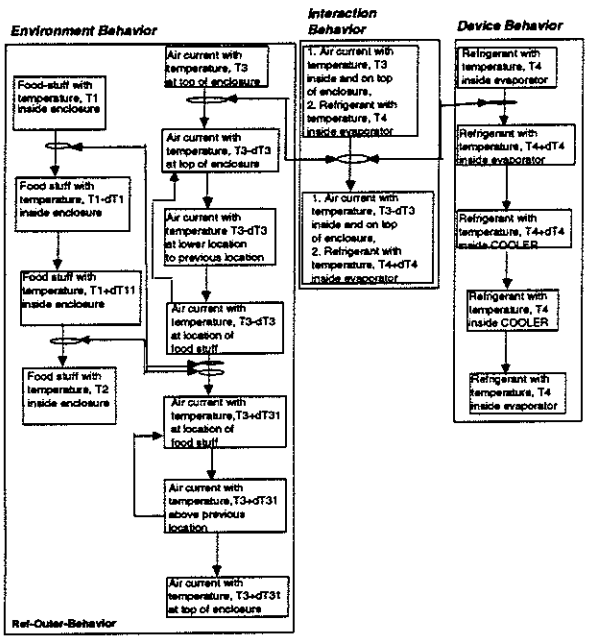


Figure 5. A Partial behaviour for Refrigerator MID.

The behaviour of refrigerator MID is given in figure 5 as a SBF model. It has three distinct behaviours: environment, device and interaction behaviours. The environment is a composition of two causal behaviours. The first one is about the temperature variations of food stuff. The second is about the movement of air currents within the enclosure. These two behaviours interact through a physical principle of heat exchange. The air current behaviour is not shown completely. The air current, after receiving heat from the food stuff, moves towards lower temperature spatial point immediately next to it. In this case, the lower temperature spatial point is above the current location. So the air current moves in the upwards direction.

The device behaviour starts with the refrigerant in the evaporator with temperature T4. It absorbs the heat from the air currents in the enclosure and then passes the heat to the COOLER which consists of condenser, compressor and expander. Then the COOLER cools the temperature of the refrigerant to T4 as a result of which evaporator returns to temperature T4.

The description of the environment illustrated in figure 5 is incomplete. It illustrates only the air current that circulates vertically. But there can be other air currents also. The air currents can be moving at an angle to the base of the enclosure, in addition to moving vertically. This is because of the temperature gradients existing at a horizontal plane. These gradients are set up primarily for two reasons: (i) the two adjacent air currents can differ in temperatures, (ii) the temperature of a spatial point nearby can be different from that of the food-stuff. The refrigerator walls perform an important function of containing the direction of flow of these air currents. These behaviours are not illustrated in figure 5 and can be seen in Prabhakar et al (1995).

3 ADAPTATION

We extend the adaptive modelling technique developed in (Goel 1991, Bhatta et al 1994) for designing interactive devices. Figure 6 illustrates the adaptation strategy that makes use of the SBF model of the interactive devices. This computational process has as inputs the new design problem as a functional description, a known SBF model of interactive device. Then it will adapt the known design to come up with a model for the new design problem.

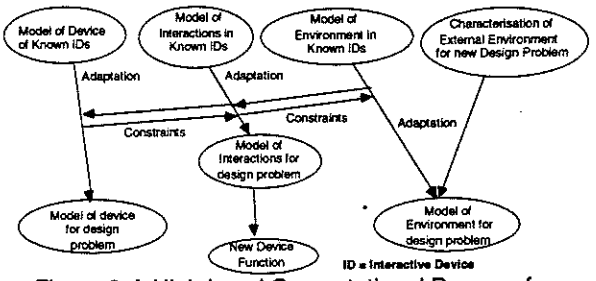


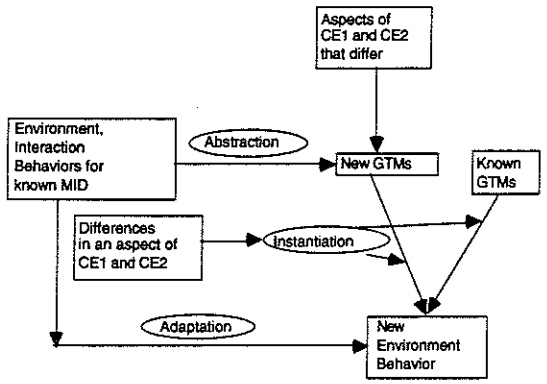
Figure 6. A High Level Computational Process for adapting functional models of interactive devices.

This adaptation has three subtasks - the adaptation of models of device, environment and their interaction. Since the models of device and the environment in the adapted interactive device need to interact, the adaptation subtask is constrained by the other adaptation subtasks. This is achieved by a constraint generation, propagation and satisfaction mechanism. For details of this computational process and some examples the reader is referred to our recent paper (Prabhakar et al 1995).

The different subtasks of adaptation are enabled by the fact that the model of the ID that is being adapted explicitly represents the models of device, environment and their interactions. Further, the roles of these models within a function guide the roles of the models in the ID that is being designed.

Figure 7 illustrates a computational process for the adaptation of environments. In simple adaptations, environment of a known MID can be adapted using one of the strategies of Kritik2 (Bhatta and Goel 1993). In general, this is not possible as a small difference in external environments can give rise to quite different behaviours. The behavioural principles used within the model of environment may not be accessible in a central

repertoire due to wide variations in behaviours among the environments. Hence, it is necessary to learn behavioural principles from the known models of environments. We do this by abstracting teleological mechanisms from the known models of environment (Bhatta and Goel 1993). These teleological mechanisms, called Generic Teleological Mechanisms (GTMs), are then instantiated within the new external environment of the new design problem.



CE1: Characterisation of External Environment for Known MID
CE2: Characterisation of External Environment for New Design Problem

Figure 7. A Computational Process for the design of Functional Model for New Environment.

These behaviours which model the new environment can be substantially different from those of the known environment. Hence they may give rise to new interactions with the device. The constraint posting mechanism across different adaptation subtasks address these modifications in the environment.

3.1 Implementation

The extended SBF model for IDs and the computational processes for designing new IDs have been tested on a number of examples from various domains. Some of these examples are :from Coffee Vending Machine to Automated Teller Machine, from Refrigerator to Air-Conditioner, and from Watermill to Windmill. The implementation of these processes in CommonLISP is still in progress.

4 RELATED RESEARCH

The functional model proposed in this paper addresses several issues of current interest in functional modelling. It extends the Structure, Behaviour and Function (SBF) models (Goel 1991) so that it can be used for non-local interactions caused, especially, by the environment. Our SBF model integrates physical principles with the intended function of the ID as in CFRL (Vescovi et al 1993]. Since our model incorporates several subdomains, and each subdomain can have an unique set of physical principles, a single model may have several sets of physical principles integrated into it. While CFRL allows different kinds of simulations to be performed on it, our model allows only those simulations that allow explication of the function.

Our functional model allows representation of multiple models as in multimodelling approach (Chittaro et al 1993). Our model represents the functional, behavioural and structural models for multiple domains. While multimodelling approach is used for diagnosis our approach is used for adaptation in design. Another stream of research is multilevel flow model (MFM) which is related to our research (Larsson 1996). Similar to MFM we represent roles along with function. In addition, we also represent behaviours for interaction with other behaviours and also the interaction behaviours. Our modelling is uniform, ie we represent same ontologies for different domains. We also use these models for adaptation in design unlike diagnosis in MFM.

5 DISCUSSION

The problem addressed in this paper is to model devices for adaptations of non-local interactions. We presented a functional model as an extension of SBF model, which uniformly represents different domains that may be present within a device. It also represents the teleological relationships between the structure and the function of the device in the form of roles and interactions at different levels. The representation in each subdomain is done not only to represent its role to give rise to the function of the device but also to support interactions with the other subdomains. This model has been tested on a number of examples from different engineering domains.

This model is scalable. For example, subdomains can be present within subdomains. This is not a problem of abstraction. The hierarchical organisation represents the kinds of interactions present at different levels. For example, in a refrigerator, there are interactions among the structural elements of device and the enclosure. These together interact with the environment present in the room where the refrigerator is present, and so on. By allowing multiple levels of new interactions, our model is scalable to more complex physical devices.

Our model assumes that the domains are representable using uniform representation. Representing interaction behaviours can become very hard due to this. Further it may be required to represent the domains using multiple ontologies. This problem is being investigated.

The division of a device into subdomains assumes a theory that allows this division[Simon 1981]. The adaptation of a model to a new design problem should be able to preserve this theory. It is done implicitly by the computational processes of adaptation. For a more flexible adaptation, we may have to reorganise the device into different subdomains. This requires explicating the theory and then modifying it. This work is also still in progress.

References

Bhatta, S, Goel, A and Prabhakar, S.: 1994, Innovation in Analogical Design: A Model-based Approach, *in* J S Gero and F. Sudweeks (eds), *Artificial Intelligence in Design '94*, Kluwer Publishers, Netherlands, pp 57-74.

Bhatta, S. and Goel, A.: 1993, Learning generic mechanisms from experiences for analogical reasoning, in *Proceedings of the fifth annual conference of the Cognitive Science Society*, Boulder, CO., pp. 237 - 242.

Chittaro, L.; Guida, G., Tasso, C. and Toppano, E.: 1993, Functional and Teleological Knowledge in the Multimodelling Approach for Reasoning about

Physical Systems: A Case Study in Diagnosis, in *IEEE Proceedings of Systems, Man and Cybernetics*, 23 (6), pp. 1718 - 1751.

Gero, J.: 1990, Design Prototypes: A Knowledge representation schema for design, *AI Magazine*, 99 (4), pp. 26-36.

Goel, A.: 1991, A model-based approach to case adaptation in *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society,* Chicago, IL, pp. 143-148.

Goel, A. and Prabhakar, S.: 1994, A Control Architecture for Redesign and Design Verification, in Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems, Brisbane, Australia, pp. 377 - 381.

Larsson, J. E. : 1996, Diagnosis based on Explicit Means-end Models, in *Artificial Intelligence*, 80, pp. 29 - 93.

Prabhakar, S. and Goel. A.: 1992, Performance-Driven Creativity in Design: Constraint Discovery, Model Revision, and Case Composition, in (Ed: J. Gero and F. Sudweeks) *Preprints of Second International Round-Table Conference on Computational Models of Creative Design*, Heron Island, Australia, pp. 101 - 127.

Prabhakar, S., Goel, A. and Bhatta, S.: 1995, Adaptive Modelling in the Design of Interactive Devices: Towards a Computational Theory of Engineering Invention, in (Ed: J. Gero and F. Sudweeks) *Preprints of Third International Round-Table Conference on Computational Models of Creative Design*, Heron Island, Australia, pp. 267 - 302.

Simon, H A. : 1981, *The Sciences of Artificial.*. Cambridge, MA: The MIT Press.

Vescovi, M.; Iwasaki, Y. and Fikes, R. : 1993, CFRL: A Language for Specifying the Causal Functionality of Engineering Devices, in *Proceedings of AAAI-93.*

Pushing Constraints in Templates for Mining Association Rules

Jia Liang Han
Department of Mathematics and Computing
University of Southern Queensland
Toowoomba, Queensland 4350 AUSTRALIA
han@usq.edu.au

Abstract

Templates have been used in data mining to display interesting rules. We push constraints in templates into known efficient algorithms for mining association rules. We analyze computational costs of Apriori and OCD algorithms and the effect of pushing the constraints into these algorithms. If constraints in templates on the database may be represented by a factor γ, pushing constraints into these algorithms improves efficiency by $1/\gamma$.

Keywords Data mining, association rules, interesting rules, templates, cost analysis

1 INTRODUCTION

Data mining [9], also known as *knowledge discovery in databases*, extracts implicit, previously unknown information from data. Data mining has wide applications and attracted considerable interests recently. *Association rules* introduced by Agrawal, Imielinski, and Swami [1] are simple but useful knowledge representations that characterize regularities in data. As an example, consider sales data of a supermarket. An association rule could be $bread, cereal \Rightarrow milk$ $[0.2, 0.8]$, which means that (1) 20% of all purchases involve bread, cereal, and milk; (2) a customer who buys bread and cereal is likely (with 80% certainty) to buy milk as well. The following examples are from [1].

- Find all rules that have "diet coke" as the consequent.
- Find all rules with "bagels" in the antecedent.
- Find all rules that have "sausage" in the antecedent and "mustard" in the consequent.

Association rules may help the manager to design schemes to promote sales or to allocate resources.

Because the data size we are interested is normally very large, it is essential to use efficient algorithms for mining association rules (see [5] for an illustrative example). Several fast algorithms have been proposed, including AIS [1], Apriori and related algorithms [2], OCD [7], SETM [5] and DHP [8].

One difficulty common to many data mining areas is overabundance of discovered patterns that are either obvious, redundant, or useless [10]. The number of discovered association rules can be very large even for a not too complicated database, as shown in [6]. Among discovered association rules, often only a small number of them, known as "interesting rules", have significance to applications. It is important to restrict association rules to only "interesting" rules. Otherwise the user has to examine a large number of discovered rules to extract useful ones.

To display only interesting rules, Klemettinen et al. [6] used *templates* [4] and introduced a *class hierarchy* (domain knowledge) to organize items. In addition, they developed graphic user-interface. Their results showed that templates can be effective in restricting the number of association rules. Templates are quite general. For example, suppose there is only one class, C, for all items in the supermarket example. The three rules may be expressed as

$$C+ \Rightarrow diet\_coke \quad (1)$$

$$bagels, C+ \Rightarrow C \quad (2)$$

$$sausage, C+ \Rightarrow mustard \quad (3)$$

The approach implied in [6] first finds all association rules using a fast algorithm, then applies the templates. Templates and classes are constraints on rules. One may compare mining association rules with processing relational queries. In a way, the above approach amounts to delaying constraints in evaluation until the last step. Similar strategies are known to be very inefficient in relational query evaluation. To optimize relational queries, constraints such as selections and projections are pushed into evaluation as early as possible. The same can be done here. In this paper, we examine constraints in templates and classes and push constraints into the known algorithms.

We analyze costs of Apriori and OCD with and without pushing constraints. We show that pushing constraints often improves performance significantly. An item is *specified* by a template if it is given explicitly in the template. For example, sausage in (3) is a specified item. Suppose a template has only one specified item and the probability of its occurrence in a transaction is 5%. Pushing

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/375 ©1996 FLAIRS

the constraint may improve performance by a factor of 20. More than one specified item in a template often imply stronger constraints. Classes in a template are also constraints but such constraints are generally weaker than specified items and the benefit of pushing constraints is less significant. In addition, pushing constraints may reduce the size of transient data, which are intermediate results during processing, thus might eliminate some disk I/O.

This paper is arranged as follows. Section 2 introduces the definitions. Known algorithms are reviewed in section 3. In section 4, we push constraints into known algorithms. We analyze costs of Apriori and OCD and the effects of pushing constraints in section 5. The conclusion is in section 6.

2 DEFINITIONS

Let U be the universal set, i.e., the set of all items. A transaction t is a subset of U, $t \subseteq U$. Consider two itemsets X, W. If $X \subset W \subseteq U$, then W *contains* X and W is an *extension* of X. The cardinality (or the size) of an itemset X is denoted as $|X|$. A *k-itemset* is an itemset of size k. An *m-extension* of X is an itemset $W \subseteq U$ such that $X \subset W$ and $|W| = |X| + m$. For notational convenience, we use XY to denote a set formed by inserting Y into X. A unique identifier *TID* is associated with each transaction. The database D consists of a large collection of transactions. Let $|D|$ denote the number of transactions in D. Let X be a set of items and A an item, $X \subset U, A \in U$, and $X \cap A = \emptyset$. An *association rule*

$$X \Rightarrow A\ [s, c], \quad s \geq s_t, c \geq c_t, \tag{4}$$

is *satisfied* by D if there are $s|D|$ transactions in D which contain $X \cup A$ and of all transactions in D which contain X, c fraction of them also contain A. s is known as the *support* and c the *confidence*. Both s and c are usually expressed as fractions. $c = s_{XA}/s_X$, where s_{XA}, s_X are the supports for XA, X, respectively. Thresholds c_t, s_t are supplied by applications, which eliminate statistically insignificant association rules. An itemset X is a *large (small) itemset* if $s_X \geq s_t (s_X < s_t)$.

Example 1. As a simple example, let us consider the universal set $\{1, 2, 3, 4, 5, 6, 7, 8\}$ and the database

$$\{1234, 12456, 28, 1267\}.$$

For notational convenience, we may use 1234 to denote itemset $\{1, 2, 3, 4\}$. Suppose the thresholds are $s_t = 0.3, c_t = 0.8$. From the data, support for $\{1, 2\}$ is $0.75 > s_t$. Support for $\{1\}$ is 0.75; for $\{2\}$, $s = 1$; for $\{3\}, s = 0.25 < s_t$. Thus, $\{1, 2\}, \{1\}, \{2\}$ are large itemsets while $\{3\}$ is small. The given database satisfies association rule $1 \Rightarrow 2$ since support for $\{1, 2\}$ is greater than s_t and the confidence $c = 1 > c_t$. However, $2 \Rightarrow 1$ is not satisfied since the confidence $c = 0.75 < c_t$. □

To restrict discovered rules to only interesting rules, a class hierarchy and templates were introduced in [6]. First, items are divided into *classes*, which are then arranged into a class hierarchy. For example, courses offered in a university may be divided into disciplines: arts, sciences, engineering, and further divided into levels: basic, undergraduate, graduate courses. A template is an expression of form

$$A_1, \ldots, A_k \Rightarrow A_{k+1}, \tag{5}$$

where each A_i is either an item, a class name, or an expression of form $C+$ or $C*$, where C is a class name. Here $C+$ represents one or more instances of class C and $C*$ for zero or more instances of class C. Similar types of constraints were known in [1] as syntactic constraints.

Two types of templates were considered in [6]: *inclusive* and *exclusive*. An interesting rule has to match the inclusive template and must, however, *not* match the restrictive templates.

3 KNOWN ALGORITHMS

All the previously proposed algorithms use two phases to discover association rules. The first phase finds all large itemsets in the database. The second phase derives association rules using the large itemsets. The second phase is simple. Suppose that W is a large itemset and $A \in W$. Then, $W - A \Rightarrow A$ is a potential association rule. The remaining problem is to determine confidence for this rule, which may be achieved by some simple counting. Thus, the key is to find all large itemsets efficiently.

The order of items in a set is insignificant. However, an implementation of a set is usually ordered. To eliminate redundancy due to permutation, we assume that a set is sorted and extend itemsets in only the ascending order.

The following propositions are essential to efficiency of several previous algorithms.

Lemma 1. *If an itemset X is small, then any of its extensions must also be small.*

Proof: Let t_X be a set of transactions in D that contains X. Since X is a small itemset, we have $s_X = |t_X|/|D| < s_t$. An itemset X is contained in transaction T iff $X \in 2^T$, i.e., the power set of T. If T contains XY, $XY \in 2^T$. Hence, $X \in 2^T$ and T contains X as well. Thus, $t_{XY} \subseteq t_X$ and $|t_{XY}| \leq |t_X|$ for any set Y. Therefore, $s_{XY} = |t_{XY}|/|D| \leq s_X < s_t$. □

Corollary 1.1. *If an itemset X is large, then any of its subsets is also large.*

We will adopt the following notations in [2].

- L_k: large k-itemsets. Each member has two fields: (i) itemset and (ii) support count.
- C_k: *candidate sets* for L_k. Each member has two fields: (i) itemset and (ii) support count.

Candidate sets are those k-itemsets that are potentially large.

(1) L_1 = {large 1-itemsets};
(2) **for** ($k = 2; L_{k-1} \neq \emptyset; k++$) **do begin**
(3) C_k = apriori-gen(L_{k-1});
(4) **forall** transactions $t \in D$ **do begin**
(5) C_t = subset(C_k, t);
(6) **forall** candidates $c \in C_t$ **do**
(7) $c.count++$;
(8) **end**
(9) $L_k = \{c \in C_k : c.count \geq c_t\}$;
(10) **end**
(11) Answer = $\bigcup_k L_k$;

Figure 1: Algorithm Apriori

insert into C_k
select $p.\text{item}_1, p.\text{item}_2, \ldots, p.\text{item}_{k-1}, q.\text{item}_{k-1}$
from $L_{k-1}\ p, L_{k-1}\ q$
where $p.\text{item}_1 = q.\text{item}_1, \ldots, p.\text{item}_{k-2} = q.\text{item}_{k-2}$, $p.\text{item}_{k-1} < q.\text{item}_{k-1}$

forall itemsets $c \in C_k$ **do**
forall $(k-1)$-subsets s of c **do**
if $(s \notin L_{k-1})$ **then**
delete c from C_k;

Figure 2: The apriori-gen function

3.1 Apriori and Related Algorithms

The Apriori algorithm is given in Figure 1. Each iteration (database pass) considers only 1-extensions of L_{k-1} from the previous pass.

The apriori-gen function (Figure 2) first performs a restricted join of L_{k-1} with L_{k-1}. This generates all k-itemsets that can be potentially large.

Lemma 2. *C_k in the apriori-gen function is complete.*

Proof: We prove it by contradiction. Suppose C_k is incomplete, in other words, there exists a large k-itemset $l_k \notin L_{k-1} \bowtie^{-} L_{k-1}$, where $\bowtie^{-}$ denotes the restricted join. We form a $(k-1)$-itemset, l'_{k-1}, by deleting the last item from l_k and another $(k-1)$-itemset, l''_{k-1}, by deleting the next-to-last item from l_k. (Note that items in l_k are ordered; hence, strictly speaking, L_k is a set of lists.) From Corollary 1.1, both l'_{k-1} and l''_{k-1} must be large itemsets. Thus, $l'_{k-1} \in L_{k-1}$ and $l''_{k-1} \in L_{k-1}$. Therefore, $l_k \in L_{k-1} \bowtie^{-} L_{k-1}$. This contradicts the earlier assumption. □

It was observed in [2, 7] that Corollary 1.1 puts strong constraints on the candidate sets, as shown by the example below.

Example 2. Let L_3 be $\{123, 124, 134, 135, 234\}$. Without using the restricted join in Figure 2, C_4 would consist of all subsets of U having 4 items (there are 5 of them). Because of Lemma 2, we only need the restricted join which results in $\{1234, 1345\}$. Since 145 is not in L_3, 1345 cannot be a large itemset. The only valid itemset in C_4 is 1234. This example shows that Corollary 1.1 puts strong restrictions on the candidate sets generated. □

The subset function (line 5) uses hash-tree data structure [3, 2] for C_k.

AprioriTid associates each transaction and its itemsets with a TID. This view changes the representation of database. Candidate itemsets $\overline{C}_k$ become:

- $\overline{C}_k$: *candidate sets* with TID. Each member is of form $< TID, \{X_k\} >$, where X_k is a potentially large itemset present in the transaction with identifier TID.

The database D is viewed as $\overline{C}_1$ and each transaction consists of a set of 1-itemsets. $\overline{C}_1$ is passed only once in AprioriTid. For the kth iteration, $\overline{C}_k$ is passed instead of the database. Such a strategy is complete because of Corollary 1.1. The reader may refer to [2] for more details.

The AprioriHybrid algorithm uses Apriori in the initial passes then switches to AprioriTid when $\overline{C}_k$ fits into main memory.

3.2 OCD Algorithm

The OCD (offline candidate determination) algorithm by Mannila et al. [7] is similar to Apriori. They also observed independently Corollary 1.1 and its power. One main difference between OCD and Apriori is the way C_k is generated. The following condition is specified for the candidate set C_k:

$$C_k = \{X \in U : |X| = k \wedge \forall l \in L_{k-1}(l \subset X)\} \quad (6)$$

This is a statement of Corollary 1.1. Two formulas were given in [7] to compute C_k. The first, eq.(7) below, implies a join operation similar to the apriori-gen function.

$$C'_k = \{X \cup X' : X, X' \in L_{k-1} \wedge |X \cap X'| = k-2\} \quad (7)$$

However, the join in the apriori-gen function is a restricted join and only the last item can be different (the items are ordered). Recall Example 2. If eq.(7) is used instead, additional itemset 1235 would have been created for C_4. Thus, eq.(7) generates a superset of that in Figure 2. Another method proposed in [7] to compute C_k is

$$C'_k = \{X \cup X' : X \in L_{k-1} \wedge X' \in L_1 \wedge X' \not\subseteq X\} \quad (8)$$

This extends L_{k-1} by each item in L_1. L_1 is likely to include most items in the database. Thus, eq.(8) generates a superset of that from eq.(7).

In addition, Mannila et al. tested for m-extensions ($m > 1$) in one pass as well, which was mentioned but not tested in [2].

4 PUSHING CONSTRAINTS

Let us first examine the constraint types due to templates and classes. First, a template may contain one or more *specified* items. For example, in the example in the introduction *diet_coke* in rule (1) is a specified item while *sausage* and *mustard* in rule (3) are also specified. Let

```
(1a)  forall transactions t ∈ D do begin
(1b)    i := i + 1;          // count |D|
(1c)    if t satisfies constraints then begin
(1d)      count 1-itemsets in t;
(1e)      insert t into D';
                  // create a new database D'
(1f)    end
(1g)  end
(1h)  obtain L'_1;          // restricted L_1
```

Figure 3: Push constraints

us introduce the *constraint factor*, γ, to quantify the effect of a template. If the constraint factor of a template is γ, then there are $\gamma|D|$ transactions in D which satisfy the constraints of the template. The constraint due to a specified item is strong, i.e., γ is small. For a template with one specified item a, $\gamma = s_a$. If $s_a = 0.05$, then only 0.05 fraction of all transactions are relevant to the association rules for such a template. More than one specified item in a template usually indicate stronger constraints since $s_{ab} \leq \min(s_a, s_b)$, where a, b are the specified items. However, the constraint factor is not $s_a s_b$ unless statistical independence is assumed, which is uninteresting for applications.

The second type of constraints is due to classes. If a template has a term of class C_i, then any transaction that does not contain at least one item from class C_i is irrelevant. This type of constraints is generally much weaker than the one due to specified items; It is less likely that a transaction contains no item from a class. γ depends on the data and the size of the class. In the extreme case, if a class has only one type of item, γ reduces to that for the specified item constraint.

The third type of constraints is from a template which uses only items and/or classes (no $C+$ or $C*$). Then the total number of items in the rule is fixed. Any transaction which contains fewer items is irrelevant. Whether such constraints are strong or not depends on the data and the template.

We now describe a method to push constraints early in known algorithms. Consider Apriori (also OCD since they are similar). We replace line 1 in Figure 1 by Figure 3 and use D' for D and L'_1 for L_1 in future database passes. A similar strategy may be used for AIS, AprioriTid, AprioriHybrid, SETM, and DHP.

To derive association rules under constraints of an inclusive template and some restrictive templates, we first push the constraints of the inclusive template into a fast algorithm and discover association rules for the inclusive template. Then, we apply the restrictive templates to trim the rules further.

5 COST ANALYSIS

For simplicity, we assume that main memory is large enough to contain C_k, L_k. Consider Apriori in Figure 1. Line (1) requires one database pass. In the pass, 1-itemsets are generated and counted. If a good hashing function is used, matching and searching an item can be implemented efficiently, i.e., constant time. The cost of line (1) is approximately $O(\|D\|)$, where $\|D\|$ is the size of D. At the same time, disk I/O costs $O(C_d \|D\|)$, where C_d is the cost for disk read/write.

Line (3) is realized in Figure 2. In the worse case, the restricted join requires $O(|L_{k-1}|^2)$ operations. The pruning procedure in Figure 2 requires $O(k|C_k||L_{k-1}|)$ operations. Thus, the apriori-gen function costs

$$O(|L_{k-1}|^2, k|C_k||L_{k-1}|) \tag{9}$$

The loop starting from line (4) involves all transactions in D. Let cost of the subset function be f_s. The cost for this loop is in the order of

$$O(f_s|D|, |C_t||D|). \tag{10}$$

f_s depends on C_k and t as well as the data structures and algorithms to implement the function.

Therefore, the total cost for the main loop is in the order of

$$O\left(\sum_k \left\{|L_{k-1}|^2, k|C_k||L_{k-1}|, f_s|D|, |C_t||D|\right\}\right) \tag{11}$$

In addition, each iteration of the main loop requires one database pass. Thus, the total disk read/write is

$$O\left(kC_d \|D\|\right) \tag{12}$$

Among the terms in (11), $f_s|D|$, $|C_t||D|$ are usually the dominant ones because $|D|$ is very large. One way to minimize these terms is to reduce $|C_t|$. C_t depends on $t \in D$ but is contained by C_k, thus, $|C_t|$ is limited by $|C_k|$. Reduction of $|C_k|$ minimizes the terms involving f_s, C_k, or C_t in (9) and (10). Since Corollary 1.1 often reduces $|C_k|$ by a significant factor, its utilization is important to efficiency. This is a key that Apriori and OCD perform well.

Now consider efficiency of the three different methods to generate C_k in Apriori and OCD. The methods from (7) and (8) are relative simple when generating candidate sets. However, since they generate supersets of that in Figure 2 the cost of pruning is higher by some factor.

The AIS algorithm, as well as OCD, considered m-extensions ($m > 1$) in one pass. Such a method has advantages and disadvantages. If one or more database passes are saved, it would have reduced a constant factor in the total cost. The disadvantage is that it may counts small itemsets.

It is difficult to obtain analytic cost formulas for AprioriTid, AprioriHybrid, SETM, and DHP since they require estimates of itemsets. The size of itemsets depends strongly on the data and varies greatly among different iterations. The size of $\overline{C}_1$ is about the same as (a little larger because of the extra field for TID) $\|D\|$. $\|\overline{C}_k\|$ is large when k is small and, depending on the distribution of the data, it could be larger than $\|D\|$ for $k = 2, 3, \ldots$. On the other hand, for large k, $\|\overline{C}_k\|$ is usually much smaller. $\overline{C}_k$ is empty if k is greater than some cutoff.

This explains why AprioriHybrid may perform well. For AprioriTid, AprioriHybrid, SETM, the assumption that transient data resides in main memory is no longer valid and buffer management must considered. We may still see the importance of Corollary 1.1. Corollary 1.1 restricts C_k strongly, thus, also reduces $\overline{C}_k$ significantly.

Let us now consider the effects of pushing constraints on the cost of Apriori and OCD. If the constraint factor is γ, then $|D'| = \gamma|D|$. The formula (11) is still applicable if we use $|D'|$ to replace $|D|$. The dominant terms in (11) all involve $|D|$. These terms are reduced by $1/\gamma$. The constraints also reduce other terms in (11) but the reductions are in general small. The constraints reduce (12) by $1/\gamma$ as well. Thus, the total cost is reduced approximately by $1/\gamma$.

Although we do not have cost formulas for AprioriTid, AprioriHybrid, SETM and DHP, we may still compare the costs with and without pushing constraints for each iteration. Initially, $\overline{C}_1$ is the same as D'. The constraint factor reduces $|\overline{C}_1|$ by $1/\gamma$. For later passes, if statistically independence is used, then the same constraint factor would be applicable and the total cost is reduced by the same factor. However, statistical independence cannot be assumed. On the other hand, it should not be difficult to see that pushing constraints may improve the performance significantly.

6 CONCLUSION

In this paper, we study mining interesting association rules in a large database. Several fast algorithms, AIS, Apriori, AprioriTid, AprioriHybrid, OCD, SETM, and DHP are known. These algorithms do not consider the "interestingness" of association rules, thus, a large number of discovered rules may be obvious, redundant, or useless. A user has to examine numerous discovered rules to find interesting ones. Klemettinen et al. [6] used templates [4] and a class hierarchy to limit discovered association rules. Templates can be inclusive and exclusive. The approach implied in [6] first finds all association rules then applies the templates.

Templates put constraints on association rules. We compare mining association rules with processing relational queries. Applying templates at the last step resembles performing selections and projections for relational query processing at the last step, which is known to be very inefficient. For efficient query processing, selections and projections are pushed as early as possible. We consider a similar strategy for mining association rules with templates. We push constraints in an inclusive template into known algorithms.

We analyze costs of known algorithms, thus, provide a deeper understanding of performance of known algorithms. The dominant terms for Apriori and OCD are proportional to $|D|$ or $||D||$. The complexities for AprioriTid, AprioriHybrid, SETM, and DHP are more difficult to estimate since they depend on the data. Pushing constraints in a template improves performance of mining algorithms significantly. For Apriori and OCD, the improvement is by a factor of $1/\gamma$, where γ is the constraint factor due to the template.

Templates are one type of constraints. Applications, domain knowledge and user-interface may have other types of constraints. In future research, it is interesting to see how to express and push such constraints in data mining algorithms.

Acknowledgements

Useful comments from referees are gratefully acknowledged.

References

[1] R. Agrawal, T. Imielinski and A. Swami. Mining association rules between sets of items in large databases. In *Proc. 1993 ACM-SIGMOD Int. Conf. Management of Data*, pages 207–216, 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 1994 Int. Conf. Very Large Data Bases*, pages 487–499, 1994.

[3] E.G. Coffman and J. Eve. File structure using hashing functions. *Communications of ACM*, Volume 13, pages 427–432,436, 1970.

[4] P. Hoschka and W. Klösgen. A support system for interpreting statistical data. In [9], pages 325–346.

[5] M. Houtsma and A. Swami. Set-oriented mining for association rules in relational databases. In *Proc. of 1995 Int. Conf. Data Engineering*, 1995.

[6] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. In *Proc. 3rd Int'l Conf. on Information and Knowledge Management*, pages 401–408, 1994.

[7] H. Mannila, H. Toivonen and A. I. Verkamo. Efficient algorithms for discovering assoication rules. In *AAAI'94 Workshop on Knowledge Discovery in Databases (KDD'94)*, 1994.

[8] J.S. Park, M.S. Chen and P.S. Yu. An effective hash-based algorithm for mining association rules. In *Proc. 1995 ACM-SIGMOD Int. Conf. Management of Data*, 1995.

[9] G. Piatetsky-Shapiro and W. J. Frawley. *Knowledge Discovery in Databases.* AAAI/MIT Press, 1991.

[10] G. Piatetsky-Shapiro, C. Matheus, P. Smyth and R. Uthurusamy. Kdd-93: Progress and challenges in knowledge discovery in databases. In G. Piatetsky-Shapiro (editor), *KDD-93: Proceedings of AAAI-93 Knowledge Discovery in Databases Workshop.* AAAI Press, 1993.

Context Generation in Information Retrieval

Ian Ruthven and C J van Rijsbergen
Department of Computing Science
University of Glasgow, Glasgow, G12 8QQ
Scotland
email: igr,keith@dcs.gla.ac.uk

Abstract

The interaction between a user and an information retrieval system can be viewed as a dialogue in which both participants are trying to interpret the others' actions in the light of previous experience. The system then must try to generate a context in which to interpret the user's response to the presented material. This notion of context operates on a principle of relevance. Information that the system believes is relevant to the user, or that the user has indicated as relevant will form the basis of the system's notion of the context. This paper presents a way of representing a context that can use both the systems knowledge about itself and the user's response to generate a view of the retrieval session.

1 INTRODUCTION

The notion of relevance is central to Information Retrieval (IR). The system should return documents it assesses as being likely to be relevant to a user's request. This request traditionally takes the form of a query. A more interesting concept is that of pertinence, [Kem74], a definition based on the utility of the information contained within the documents to a particular user with a particular information need. This relates the relevance of a piece of information to a user rather than to a topic. Pertinence is a difficult concept to assess objectively as we have to assume that information needs can be expressed *a priori* in relation to a given database of documents. Also, to assume that needs will not change in relation to what the IR system returns, and that what the user already knows about a particular topic will not affect his assessment of what is pertinent.

Pertinence, however, has a powerful advantage in the development of system, one which motivates the work described here. Although it is difficult for a user to describe what is pertinent, it is easy for her to demonstrate it. Each time a user chooses a document as being of interest he is making a statement about the pertinence of the document and the martial contained within the document, [Kem74] [Cam95]. These statements of pertinence may be used to describe a context within which to describe the user's information needs relative to a collection of information.

In the rest of this paper I aim to describe how some of the features of IR, belief revision and a theory of communication, Relevance Theory, can be to update the system's model of a collection in the light of user's pertinence assessments.

2 RELEVANCE THEORY

The framework proposed here relies heavily on the communication inherent in the user's interaction with the system. The metaphor used here is one of a conversation, the system questions the user by presenting what it regards as germane to the discussion, the most relevant documents. The user answers by selecting the most representative document to her information need.

To realize this dialogue, a formal model of communication is required. This model must be able to cope with the shift in focus as existing information needs and satisfied or as new needs develop in the presence of new evidence.

Two participants in a dialogue may share common terms of reference but it is unlikely that they will share the same connotations for those terms. They both know what they refer to but cannot be sure that the other person is referring to the same object or idea

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/380

in the same sense or through the same experience. In everyday conversation, we can examine each other's background, experiences, views through the dialogue. This may be acceptable in conversations with another human but it would be tedious for the user of a system to have to explain what she means in detail.

Sperber and Wilson [SW82] address the variety of interpretations available to a speaker and hearer, and use them to develop the Relevance Theory approach. Here what is important is not the knowledge a speaker has about what the hearer knows or believes but the relevance of an utterance of the current context of the conversation.

Uttering a proposition within a context will give rise to contextual implications - non-trivial logical implications that could not be derived solely from the content of the propostion nor from the context in which the proposition was expressed. Both existing information and new evidence combine to infer the contextual implications. Any utterance that produces contextual implications will be relevant to some degree, the degree of relevance being determined by a number of factors, including amount of processing involved to produce the implications and the number of contextual implications generated by the sentence.

Sperber and Wilson encapsulate theory theory in their principle of relevance, from 1982

> **Principle of relevance, (1982)**: speaker tries to express the proposition which is the most relevant one possible to the hearer

Relevance Theory can be described in terms of both classical information retrieval techniques and the system-based dialogue model presented here. Assume that there has been an initial statement of interest from the user, this may be a query or some other representation or description of the user's information need. The initial context is the system's model of the collection that is the terms in the collection and an assessed measure of the utility of that term. The system develops the context by trying to derive contextual implications based upon the user's document selection. In other words it decides how the user's selection affects it's model of what the user is looking for.

This means that we want to use the user's previous areas of interest - the previous contexts - as a basis on which to determine any new contexts. Terms that appear in a user-selected document are counted as being pertinent. This may be over-confident, as a term may not be of interest but only appear due to causal co-occurrence with another term. As we cannot say which terms are of interest we treat all as potentially pertinent.

At this point the work here differs from Relevance Theory as described for human conversation in two ways. Firstly Sperber and Wilson [SW86] talk about a strength of a sentence, this approach insteads uses a probability function distributed over the elements of the discourse - the terms that occur in the collections. Secondly in this model, when the user selects a document she does not only affect the elements of the context but also related terms. That is, although a user is only giving feedback on the information within a document we will view this as feedback on related information that is not present in the document. This is an attempt to prime the system to react to a particular section of information contained within the collection. This we treat as analogous to the contextual implications of Relevance Theory.

3 MODEL

Underpinning this model if a formalism for representing the system's knowledge. The change here is regarded as a form of belief revision. The system's beliefs focus on the terms that appear in the collection, and the fundamental belief that an system has regarding a term t, is "the 'belief that a term is the best term to describe the current stage of the search". The value assigned to the belief $\mathrm{b}(t)$ that states that t is useful in retrieval, will be based upon evidence the user has identified t as being relevant, e.g. through a query, or that there is evidence that t may be relevant, e.g. through co-occurrence of t with a relevant term, or t appearing in a user-selected document. This probabilistic model of belief revision asserts that beliefs of this type can be represented by a probability distribution over the terms space. However it requires that complete knowledge of the structure of the space in terms of the conditional relationships. As such, the revision process, when encountering new information, is less of an expansion/retraction of beliefs but a strengthening/weakening of beliefs based on evidence.

I propose two revisions are necessary, firstly the system must revise its belief system in the light of evidence from the user. Secondly it needs to adapt its new belief system to take into account the relation between its modified beliefs and the evidence that the system possesses about each document. This second revision as shall be shown below does not result in a permanent shift in the system's beliefs about the user's interest.

The first corresponds to the context modification of relevance theory. We are attempting to update the system's view of the user's search by incorporat-

ing into the system's belief system new evidence and revising any existing evidence for a belief. For this revision I propose using the form of conditionalisation known as Jeffrey conditionalisation for reasons give in section 3.1.2.

The second revision is different in that it does not result from a change in the system's world. Rather it results from a deliberate attempt to maximise the utility of the system's beliefs in predicting the information most likely to be of interest to the user. It must now try to infer the material in the document from its beliefs. Given a context and a user's response the system is trying to select the interpretations (the documents) that corresponds to what it assesses is the most likely interpretation of the user's utterance. In information retrieval terms what the system is trying to do is assess the documents according to their relevance. The most relevant documents being the most likely interpretations of the system's belief of what the search is for.

For this revision what is required is a general method of revision that can assess the probability of this implication, i.e. $P(b \rightarrow d)$, the probability that a document can be inferred from a set of beliefs. A probability measure is used as Relevance Theory states that only the most likely interpretations are considered. Not all interpretations are equally likely and not all documents are equally relevant to a user. Consequently we must have a way of ranking our interpretations, our measure of a document's relevance, as is also required of an IR system.

A process which fulfills this criterion is known as *logical imaging*, described by Gardenfors [Gar88]. Its application to IR has already been demonstrated and implemented by Crestani and Van Rijsbergen [CvR95, CRSvR95] , and so here I will only attempt to describe the features of imaging necessary to explain the interpretation which I propose using it relative to this model.

3.1 Recasting the Context

Before describing the mechanics of these procedures I want to give some definitions: the **context** is the set of terms in the collection with an associated probability value giving their importance in describing the current search stage, **selected terms** is the set of terms present in the last selected document, **indirectly selected terms** is the set of terms that co-occur with the selected terms, and **unselected terms** is the set of terms that are neither selected nor indirectly selected.

When the user selects a document, the information contained within it should be used to modify the context. The context is revised in the light of evidence from the user. There are three stages in updating the context; first the system's belief in the selected terms should be directly updated. Secondly this change is reflected in the indirectly selected terms. Finally there needs to be a scaling operation to preserve the unit sum of the probabilities.

How this updating on the terms is acheived is threefold depending on which class the term belongs. Selected terms, those that are present in a user-selected document have their probability of importance directly calculated, section 3.1.1. This means that seeing a term in a user-selected document will increase the system's belief in that term's ability to describe the current search stage. Indirectly selected terms, those that have an author-selected relationship will have their probability increased by virtue of being associated with the selected terms. In other words, these terms have their probability increased 'as if it had appeared' in the document. The more selected terms a term appears with the more its probability will be updated 3.1.2. The weakest evidence supplied by the user's selection is for the remaining terms, the unselected ones. If a term does not appear in a user-selected document, nor co-occurs with any terms in that documents, then the user selecting the document does not supply any evidence for the importance of that term. It cannot be said that there is evidence for that term being of no importance, i.e. it probability is zero. What can be said is that there is no additional evidence and so its probability remains unchanged. In fact, as a result of the scaling operation, the probability will be reduced proportional to the changes in the other terms.

3.1.1 Updating

If a term occurs in a set of relevant documents at a level greater than would be expect giving purely statistical relation then we can claim that the term is of importance. The method of how to change the probability of a term should reflect aspects such as the spread of the terms occurrences in the collection, and the frequency within the collection. The actual method chosen is not of primary importance as only scale and relative scale will vary according to the method chosen. When changing the probability we are trying to update our assessment of its discriminatory power to distinguish between relevant and non-relevant documents. A number of methods are being investiagated at the moment including the information radius and the F4 weights, see [Rij79, chapter 6] for an overview of these methods.

3.1.2 Revision

This stage propogates the evidence from updating the selected terms to other terms in the collection. The rationale behind this is that, although we can't directly access the semantic relationships between terms in the mind of the author, or searcher, we can represent the relationships by means of statistical dependencies. If we have no means of directly assessing the probability of importance of a term then it can be determined by using the probability attached to related terms. In theory means propagating belief over all the terms in the term space. In practice we only propagate it to those terms that have an author-selected relationship. This means that I am asserting that the conditional of two terms that do not co-occur is zero, as the probability of co-occurrence is zero.

$$P(term_x \mid term_y) = \frac{P(term_x, term_y)}{P(term_y)} \quad (1)$$

If $term_x$ and $term_y$ do not co-occur then $P(term_x, term_y) = 0$ and consequently $P(term_x|term_y) = 0$. So regardless of individual probability of $term_x$ and $term_y$, seeing $term_y$ alone should allow the assertion that $P(term_x)$ should be zero. Practically, if a term occurs in a document then it will co-occur with every term in that document. This means that $P(term_x)$ will be a factor of the probabilities attached to these terms and the other terms with which it co-occurs.

The system assigns evidence not only according to what is in the collection but but to evidence supplied by the user. The strength of belief in a term is then a measure generated in response to the material selected and to the body of evidence. The beliefs in terms are revised through conditionalisation. The form of conditionalisation I am proposing to use here is Jeffrey conditionalisation. Standard Bayesian conditionalisation gives a means of re-calculating the probability of an event, A, given another event B, The new probability assigned to A, P'(A), is equal to the conditional probability, P(A|B) when P'(B) = 1. Jeffrey conditionalisation generalises this to cases where P'(B) $\neq$ 1. This is appropriate here as we are not interested in the probability of events happening, we are interested in the utility of the event, given that it has happened.

Before applying conditionalisation it is necessary to make certain that three conditions hold over the set of terms; exhausivity, mutal exclusivity and stability of the conditional probabilities between terms. The aim is therefore to demonstrate that is possible to calculate the probability of $P(term_x)$ by the equation,

$$P(term_x) = \sum_{n=1}^{N} P(term_x \mid term_n) P(term_n) \quad (2)$$

where N = terms in collection.

Exhausivity:
All the terms in the collection will have an effect on the probability of $term_x$. However the conditional between non-cooccurring terms is zero, (as above) then a large section of the terms will not have an affect on the belief attached to a term. So, even though all terms are considered in theory, in practice only co-occurring terms need be considered for each term.

Mutual exclusivity:
We are measuring the importance of a term. As we have expressed this as the 'belief that a term is the best term to describe the current stage of the search' then we can assume mutual exclusivity of terms in describing this stage.

Given these two conditions hold then we can assert equation 3.

$$P'(term_x) = \sum_{c=1}^{C} P'(term_x \mid term_c) P'(term_c) \quad (3)$$

where C = set of terms that co-occur with $term_x$.

Conditional probabilities do not change:
If we assume the conditional probability reflects the underlying statistical dependency based on co-occurrence then this is valid. This gives,

$$P'(term_x) = \sum_{c=1}^{C} P(term_x \mid term_c) P'(term_c) \quad (4)$$

The new probability of an indirectly selected term depends on the probability of its associated terms, equation 4. These terms belong to two distinct classes; selected and indirectly selected terms. The probability for the selected terms has already been calculated, section 3.1.1. The question is how to the value for each of the indirectly selected terms.

Example: A user selects a document with $term_1$ and $term_2$ in it. The new probability for these terms are calculated directly. For a $term_3$ that co-occurs with terms $term_1$, $term_4$ and $term_5$, the new probability is calculated by the equation,

$$\begin{aligned} P'(term_3) = {} & P(term_3 \mid term_1) P'(term_{t1}) + \\ & P(term_3 \mid term_4) P(term_{t4}) + \\ & P(term_3 \mid term_5) P(term_{t5}) \end{aligned}$$

As there is no direct evidence for the pertinence of $term_4$ and $term_5$, we use the previous values associated to them when calculating the new probability of $term_3$. The probability associated with $term_3$ will increase due to the change in $term_1$.

3.2 Logical Imaging

So far we have calculated the impact of a document selection upon a context, given previous selections.

The next step is to produce a set of system interpretations of the user selection. That is, a list of documents ranked by assessed likelihood of being relevant to the user.

This selection is based on a form of inference, section 2. The system is trying to infer the documents in the collection from the evidence provided. Here there is no query, *per se*. The method used here is to infer the last selected document from the rest of the document collection. More accurately as we are estimating the likelihood of relevance we measure the probability of this inference. Logical imaging revises a probabilistic belief system by moving probability from terms that do not appear in the document to the closest term that does appear. This shift of probability produces a new probability distribution describing the beliefs in relation to the document. The sum of the probabilities attached to those terms appearing in the intersection of the document under consideration and the last user-selected document gives the estimate of relevance for the document.The relative relevance of documents can then be obtained by ranking.

For each new context the amount of probability associated with each term will change. Consequently the effect of changing the context is to alter the relevance of a document is twofold. One, by changing the probability of the term. Secondly by altering the probability of the terms that transfer their probability to the term when imaging.

4 CONCLUSION

Relevance Theory describes how an utterance can give rise to numerous interpretations dependent upon the context. The context and utterance together form a new context, the form of which can only be described by the utterance and particular context. This paper demonstrates how such a model can be described for IR. The context is the set of terms with an associated probability describing its importance in the search. The user selecting a document will cause an increase in the probability of related terms. The overall change preserves the consistency of the set of terms, only the values of the probabilities change. The value assigned to a term depends on the values of its associated terms.

The context is then used, with the last document selected by the user, to produce the new set of relevant documents. As the probabilities change then the measure of similarity should change according to the terms in each document and how they have changed. Also as there is no query modification as such, following [Cam95], the aproach depends only on the user's choice of documents.

Acknowledgements

This work was supported by the FERMI project (8134) - "Formalisation and Expermimentation in the Retrieval of Multimedia Information Retrieval", funded by the European Community under the ESPRIT Basic Research Action.

References

[Cam95] Iain Campbell. Ostensive support for information needs in an adpative information space. In Ian Ruthven, editor, *Miro 95*. Workshops in Computing, Springer Verlag, September 1995.

[CRSvR95] F. Crestani, I. Ruthven, M. Sanderson, and C.J. van Rijsbergen. The troubles with using a logical model of ir on a large collection of documents. Experimenting retrieval by logical imaging on TREC. In *Proceedings of the Fourth Text Retrieval Conference (TREC-4)*, Washington D.C., USA, November 1995.

[CvR95] Fabio Crestani and C J van Rijsbergen. Information retrieval by logical imaging. *Journal of Documentation*, 51(1):3 – 17, March 1995.

[Gar88] Peter Gardenfors. *Knowledge in Flux: Modelling The Dynamics of Epistemic States*. MIT Press, 1988.

[Kem74] D A Kemp. Relevance, pertinence and information systems development. *Information Storage and Retrieval*, 10(2):37 – 47, January 1974.

[Rij79] C.J. van Rijsbergen. *Information Retrieval*. Butterworths, second edition, 1979.

[SW82] Dan Sperber and Deirdre Wilson. Mutual knowledge and relevance in theories of comprehension. In N.V. Smith, editor, *Mutual Knowledge*, pages 61 – 111. Academic Press, 1982. Chapter 2.

[SW86] Dan Sperber and Deirdre Wilson. *Relevance: Communication and Cognition*. Blackwell, 1986.

AN ARCHITECTURAL DESIGN FOR IMPROVING PARALLEL HEURISTIC SEARCH

R. Craig Varnell, Diane J. Cook, Lynn L. Peterson
Computer Science and Engineering
University of Texas at Arlington
Arlington, TX 76019
varnell@cse.uta.edu, cook@centauri.uta.edu, peterson@cse.uta.edu

Abstract

Many artificial intelligence applications rely on performing heuristic search through large, complex spaces. Iterative-Deepening-A* (IDA*) search has been shown to be useful for large search spaces, because it requires no intermediate state storage and is guaranteed to find optimal solutions. However, the time taken to perform IDA* search on real-world tasks often prevents the everyday usage of AI techniques.

Parallel processing can considerably reduce the time spent in search, and can thereby speed up AI applications. A number of approaches to MIMD heuristic search have been introduced in the literature. In this paper we present a system that can be used to dynamically select between alternative techniques at run time. In particular, our system uses a knowledge-based approach to select the search strategy that is most likely to optimize performance. Experimental results show a substantial performance improvement over serial search algorithms, and indicate an improvement over existing parallel search approaches used in isolation.

1 INTRODUCTION

Heuristic search provides the driving force for many applications of artificial intelligence including problem solving, robot motion planning, concept learning, theorem proving and natural language understanding. Computational complexity is one of the limitations of search. The research community is continually trying to develop efficient search algorithms.

Parallel search algorithms significantly increase the number of nodes evaluated in a given amount of time. Parallel search techniques have been implemented on MIMD and SIMD architectures. Because of the overwhelming size of real-world AI applications, and because of the increasing power and accessibility of parallel computers, there exists a constant need for improvement of parallel search algorithms.

This paper will present a hybrid approach that demonstrates improved performance for parallel heuristic search. We have developed a new architecture combining the benefits of multiple approaches to parallel heuristic search. The C4.5 [9] machine learning system is used at run time to select the best parallel search technique likely to produce the best results.

2 APPROACHES TO PARALLEL SEARCH

A* is a popular search technique that guarantees optimal solutions. Two problems that exist with A* search, however, are memory requirements and time complexity. The amount of memory required to complete a problem is exponential in the depth of the solution, as is the computation time required.

A number of improvements have been made to reduce the memory and time resources required by the search process. IDA* is one example of an algorithm that can overcome the memory requirements of A*[4]. The IDA* search algorithm consists of a series of depth-first searches. On each search iteration a depth limit is set, and when a node's total cost exceeds the limit, the search of that node's subtree is abandoned. As with A* search, the heuristic cost estimating function cannot overestimate the true cost to reach the goal for IDA* search to guarantee an optimal solution path. If the depth-first search runs to completion without finding a goal within the limit, another depth-first search is performed with the depth limit set to the smallest total cost that exceeded the limit of the previous search. This process is continued until a goal node is reached.

Parallel processing methods have been investigated to improve the efficiency of heuristic search. Two examples of MIMD approaches to parallel search are distributed tree search [3] and parallel window search [7]. Distributed tree search assigns disjoint parts

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/385 ©1996 FLAIRS

of the search space to individual processors. Since each processor is looking at a distinct part of the search tree, the execution time can be reduced. In contrast, parallel window search replicates the same search space across all processors but assigns a different cost limit to each processor. Figures 1 and 2 illustrate the techniques employed by distributed tree search (DTS) and parallel window search (PWS), respectively.

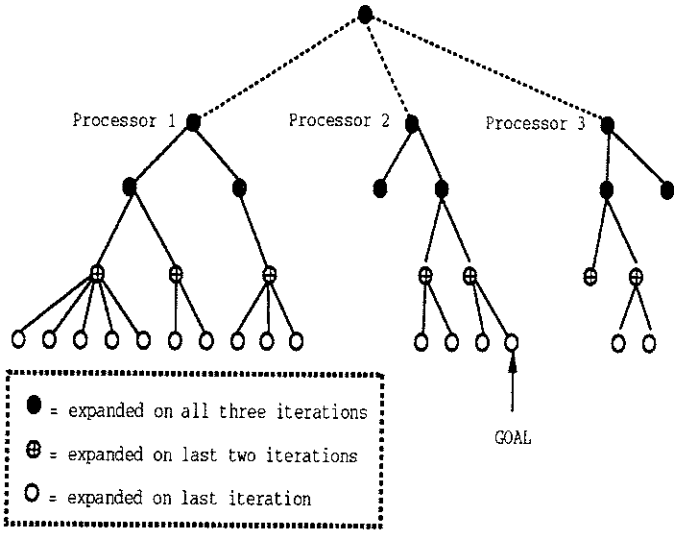


Figure 1: Division of work in distributed tree search

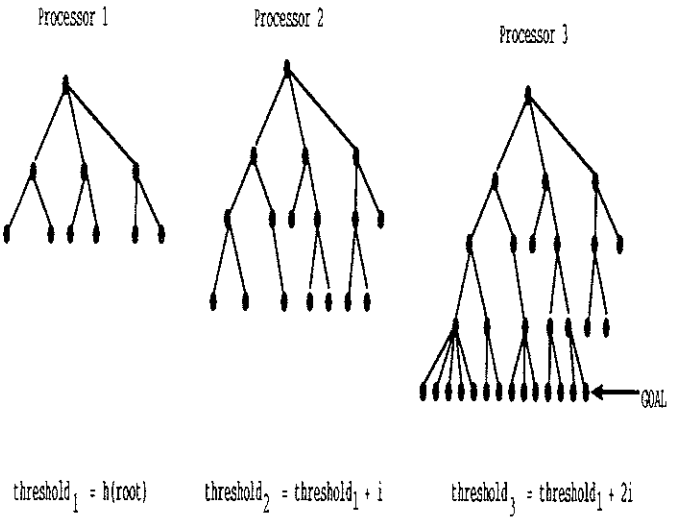


Figure 2: Division of work in parallel window search

A number of MIMD heuristic search techniques employ load balancing to improve performance. A parallel system partitions work among the processors. Since each of these subtasks may be of varying difficulty, some processors will become idle before others and require additional work. A process called *load balancing* addresses the problem of load imbalance in parallel system. A number of load balancing approaches have been successful in parallel search. One example is the nearest neighbor approach [2, 5]. An idle processor sends work requests to its nearest neighbors in a round-robin approach. If a processor has extra work then its current work load is divided into two parts and one is sent to the idle processor.

Another technique that has been shown to improve the performance of heuristic search is ordering. Problem solutions can exist anywhere in the search space. Using IDA* search, the children are expanded in a depth-first search from left to right. If the solution lies at the far right side of the tree, a far greater number of nodes must be expanded than if the solution lies at the far left side of the tree. To alleviate this problem, individual nodes in the tree can be reordered based on their heuristic estimate [8]. Alternatively, the order of operator application that lead to the most promising node at the bottom of a subtree explored in the previous search pass can be stored and used to reorder operator application for the next search pass [1].

3 SEARCH ARCHITECTURE

This section will describe the search architecture that we have developed. The objective as set forth from the beginning of this project was to determine the factors that affect the performance of parallel search techniques, and compare the various approaches. Our architecture merges a variety of techniques for parallel search. As a result, it can be used to compare differing approaches. In addition, this paper describes how a specific strategy can be dynamically selected to optimize performance.

3.1 Rule Generation

In order to choose the right approach for a particular problem, a rule-based knowledge base was required. Human-generated rule bases are often incomplete and incorrect. We therefore employ the C4.5 machine learning system [9] that is capable of building a decision tree from a set of test data. C4.5 uses a classification approach to developing the decision tree. The decision tree is then used to produce a rule set. This section will describe the search architecture and the content of the rule base.

Our approach to combining the benefits of various parallel heuristic search techniques utilizes a rule base generated by a machine learning system. The learner is provided training examples in the form of search

problems using a variety of strategies and the corresponding run times. The learning system outputs a set of rules that selects parameters such as the distribution mechanism (DTS or PWS), node ordering method, and load balancing technique given indicative features of the search problem. For a new search task, we examine the search space to a shallow depth in order to gather information about the search space. This information is fed into the rule base and an appropriate strategy is selected and utilized for the remainder of the search process.

During the rule generation phase the following steps are performed:

1. We run sample search problems using each of the possible search strategies. For each test problem, we note the strategy that performed best. These sample runs serve as training examples for the machine learning system.

2. The C4.5 machine learning system accepts as input the training examples and creates a set of rules indicating situations in which a particular strategy should be employed. These rules are stored for future use by our system.

3.2 C4.5 Rules

In this section we will describe the factors that contribute to the performance of a search problem. The factors that are described here are measurable parameters that can be derived from a search problem. Each one by itself is incapable of characterizing a search space or describing the best approach to a particular problem; however, when combined together they can differentiate a problem by its unique characteristics. Each of these factors is utilized as an antecedent for the rule base.

1. *Goal location* - The goal node of a particular problem can be encountered at anytime during an iteration of IDA*. And since the IDA* algorithm always searches in a depth-first fashion the nodes are expanded in a left-to-right fashion. The goal can be located anywhere from the far left of the search space indicating that it is one of the first of an iteration to be expanded to the far right indicating that it is one of the last to be generated.

2. *Branching factor* - The branching factor indicates the number of children that can be generated from a parent node. This is usually averaged across the entire search space resulting in the average branching factor.

3. *Depth of the goal* - In general, as an optimal goal is placed deeper in the tree, more iterations are required to find the goal.

4. *Optimal vs. sub-optimal solution* - An optimal solution generally requires more time than a sub-optimal solution. By relaxing the optimality requirement, the least cost path to the goal may not be found but a goal somewhere in the search space will be found.

5. *Heuristic branching factor* - The heuristic branching factor is defined as the ratio of the number of nodes generated under a particular threshold with the number of nodes with the next smaller threshold. It reflects the number of nodes which must be expanded below a leaf node of the previous iteration to complete the current iteration.

6. *Tree imbalance* - Many search trees are not uniform in the distribution of nodes. A high level of tree imbalance can increase the need for load balancing during parallel search.

7. *Heuristic error* - The heuristic error reflects the amount of error between the heuristic distance and the actual distance to the goal. This value is averaged across the entire search space to indicate how accurate the heuristic function was for the problem with the goal being to have a low average heuristic error.

The decisions made by the C4.5 system include the following parallel search strategies:

1. *Initial Distribution* - Kumar and Rao have suggested that the root node be placed on processor 0 and other nodes obtain work through load balancing[5]. The Distributed Tree search approach places a disjoint part of the search space on each processor[3].

2. *Percent of stack given away* - When a processor runs out of work it is required to obtain work from other processors. Varying the percent of the stack given away during a load balancing operation can have a significant change in performance. Some problems perform consistently better with one percentage while others perform better with another. Our strategies include giving away 30% and 50% of the stack.

3. *Stack splitting strategy* - The load balancing process divides the work load from one processor into two pieces. One of these is sent to the idle processor. Kumar and Rao [6] indicate that there

are three approaches for selecting which nodes to transfer to an idle processor.

(a) Transfer nodes near the root
(b) Transfer nodes near the cutoff depth
(c) Evenly divide the nodes between the two processors

We examined the first two of these.

4. *Parallel Window Search vs. Distributed Tree Search* - In some cases, the performance of a search problem will be better without any load balancing. The two algorithms that can perform well without load balancing are parallel window search and distributed tree search.

We provided C4.5 with 70 training examples. The machine learning system generated approximately 15 rules for each decision to be made. An example of a C4.5-generated rule is given below.

```
IF Maximum_Depth <= 37 and
   Heuristic_Branching_Factor > 8.42317

   -> 30%   (75.8%)

IF Branching_Factor <= 1.54028 and
   Imbalance > 41.1319 and
   Heuristic_Branching_Factor <= 9.29445

   -> Kumar&Rao (82.0%)
```

3.3 Application of the Rules

Once the rule set is created, then we are ready for an unseen problem. The following steps are performed:

1. A shallow pass is made through the search space capturing the same necessary features of the space.
2. Extract the results of the shallow search pass and pass to C4.5.
3. The results obtained by consulting the rule set are applied to the problem for establishing the best known approach to this particular problem.
4. Once the problem completes and the goal is found, the results of the problem are added back into the test set.

4 TEST RESULTS

In this section we will discuss results obtained by applying our architecture to test cases in the Fifteen Puzzle domain.

4.1 Fifteen Puzzle Domain

One well known search problem is the Fifteen Puzzle. This problem provides a good domain for testing because the problem size can be easily controlled and because results from parallel search algorithm are available. The Fifteen Puzzle consists of a four-by-four frame which hold fifteen movable square tiles and one blank spot. The tiles which are horizontally or vertically adjacent to the empty spot may be slid into the blank spot. The object of the puzzle is to find a sequence of tile movements that will transform the initial tile arrangement into a specified goal tile arrangement.

In this problem, there are four possible moves from any state of the puzzle. They correspond to moving a tile left, up, right, or down where each represents moving a tile to the blank location in that direction. In many places there will be less than four children because some moves are considered illegal.

In our implementation, each move is considered to have a cost of one, so the value of $g(n)$ is equal to the number of moves made to reach node n from the initial state. The optimal heuristic for estimating the distance from any give board configuration to the goal location is the Manhattan Distance Function. It sums the number of horizontal and vertical grid positions that each tile is from its goal position. The value of the heuristic estimate $h(n)$ is the sum of these tile distances.

4.2 Experiments

A set of five problems were used to test the architecture. The following table shows the processing time on a single node of an nCUBE-2 parallel computer.

KORF PROBLEM	SEQUENTIAL TIME
10	27:02:26
54	32:31:28
26	42:43:58
67	45:12:55
21	25:09:08

For each of the factors that we considered, a test was conducted that compared the execution times of each search strategy. The timings for initial distribution strategies are provided below. In addition, we indicate the strategy that was selected by C4.5 for each problem. The speedup achieved by our system over the sequential search and over a random strategy selection (average run time over all strategies) is also provided.

Initial distribution: Kumar vs. DTS

K&R	DTS	C4.5	SP_{seq}	SP_{avg}
1:31:57	1:10:40	DTS	22	1.151
0:48:37	1:05:52	K&R	40	1.177
1:18:09	1:18:38	DTS	32	0.977
1:13:28	2:14:37	K&R	36	1.416
0:26:00	0:27:07	K&R	58	1.021

Percent of stack: 30% vs. 50%

30%	50%	C4.5	SP_{seq}	SP_{avg}
1:10:40	1:14:37	30%	22	1.028
1:05:16	0:35:24	50%	55	1.430
1:18:38	1:10:46	50%	36	1.027
2:14:37	2:05:06	50%	21	1.038
0:27:07	0:25:57	50%	58	1.022

Stack Splitting: Bottom vs. Top

BOTTOM	TOP	C4.5	SP_{seq}	SP_{avg}
1:31:57	1:21:53	TOP	19	1.061
0:48:37	0:55:10	BOT	40	1.067
1:18:09	1:33:27	BOT	32	1.098
1:13:28	2:24:35	TOP	18	0.754
0:26:00	0:30:21	TOP	49	0.928

No Load Balancing:PWS vs. DTS

DTS	PWS	C4.5	SP_{seq}	SP_{avg}
1:08:23	0:15:53	DTS	23	0.616
1:08:56	4:35:02	PWS	8	0.625
1:24:26	2:55:10	DTS	30	1.537
2:56:06	1:28:07	DTS	15	0.750
0:42:44	0:12:43	DTS	35	0.648

These results clearly show that the use of C4.5 assists in the selection of the best approach to parallel search. The overall average speedup of these runs was 1.019.

5 SUMMARY

Search in Artificial Intelligence has a price of exponential complexity. It is a crucial, yet costly, component of virtually all other Artificial Intelligence techniques. Parallel processing helps to reduce this complexity. To reduce the complexity even further, we have designed an architecture that takes a task-specific approach looking at each search problem independent of all others. The same approach can not be used for every search problem due to the nature of the problem. Our system can be used to optimize performance for each unique task, despite dramatic variations in task search spaces. By making use of a machine learning system to construct the rule base, we can select a parallel search strategy that has a high probability of yielding the greatest possible performance for each new search problem.

References

[1] D.J Cook, L.O. Hall, and W. Thomas. Parallel search using transformation-ordering iterative deepening-A*. *International Journal of Intelligent Mechanisms*, 1993.

[2] Shantanu Dutt and Mahapatra. Scalable load balancing strategies for parallel a* algorithms. *Journal of Parallel and Distributed Computing*, 22(3):488–505, 1994.

[3] C. Ferguson and R.E. Korf. Distributed tree search and its application to alpha-beta pruning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 128–132. AAAI, August 1988.

[4] R.E. Korf. Depth-first iterative deepening - an optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.

[5] V. Kumar, G.Y. Ananth, and V.N. Rao. Scalable load balancing techniques for parallel computers. Technical Report 91-55, Computer Science Department, 1991.

[6] V Kumar and V.N. Rao. Scalable parallel formulations of depth-first search. In V. Kumar, P.S. Gopalakrishnan, and L.N. Kanal, editors, *Parallel Algorithms for Machine Intelligence and Vision*, pages 1–41. Springer-Verlag, 1990.

[7] C. Powley and R.E. Korf. Single-agent parallel window search: A summary of results. In *Proceedings of the Eleventh International Joint Conference on Artifical Intelligence*, pages 36–41. IJCAI 89, 1989.

[8] C. Powley and R.E. Korf. Single-agent parallel window search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5), 1991.

[9] J. Ross Quinlan. *C4.5: Programs For Machine Learning*. Morgan Kaufmann, 1993.

Data Mining with Concept Generalization Graphs

Wanlin Pang, Robert J. Hilderman, Howard J. Hamilton, and Scott D. Goodwin

Department of Computer Science
University of Regina
Regina, Saskatchewan, Canada, S4S 0A2
{pang,hilder,hamilton,goodwin}@cs.uregina.ca

Abstract

We present the Path-Based Generalization and Bias-Based Generalization algorithms for attribute-oriented induction using concept generalization graphs. A single concept generalization graph is constructed from multiple concept hierarchies associated with an attribute in a database. Each algorithm efficiently manages the generalization process by avoiding unnecessary regeneralization and by determining which intermediate generalized relations to store for possible future use.

1 Introduction

Attribute-oriented induction has been shown to be an effective method for knowledge discovery from databases [1-8]. A concept hierarchy (CH) associated with an attribute in a database is represented as a tree where leaf nodes correspond to actual data values in the database, intermediate nodes correspond to a more general representation of the data values, and the root node corresponds to the most general representation of the data values. For example, a CH for the Location attribute in a sales database is shown in Figure 1.

Higher level node values can be learned through generalization of the sales data as follows. First, remove any attribute from consideration which does not have a CH. Second, compile statistics about the number of distinct values encountered for the attribute(s) being generalized. Third, replace data values with higher level node values and remove duplicate tuples. Finally, generalize further if the number of tuples in the resulting generalized relation is greater than the specified attribute threshold.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/390 ©1996 FLAIRS

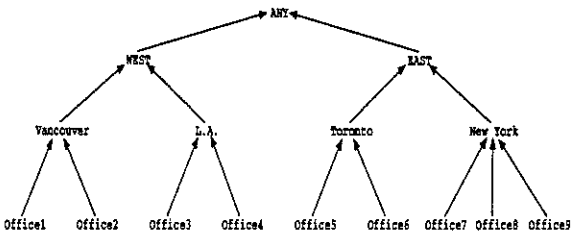


Figure 1: Concept hierarchy

Generally, work on attribute-oriented induction has assumed that each attribute has a single concept hierarchy and that this hierarchy is a tree. Earlier work on the version space algorithm [9] in Machine Learning assumed that if any generalization occurred for an attribute, all values for that attribute would be changed to "?", which corresponds to having a two-level concept hierarchy with only leaf values and a root of ANY. In this paper, we consider how to generalize when several trees are provided as concept hierarchies for the same attribute. Since additional complexity results when we consider multiple attributes, each with multiple trees, we focus here on procedures appropriate to a single attribute with multiple trees.

When knowledge about an attribute can be expressed in different ways (e.g., by multiple CHs), the generalization technique outlined above may not produce a satisfactory result using the selected CH. For example, summarizing using the CH from Figure 1 does not provide any information regarding sales by country. To accomplish this, another CH for tracking sales could be used, such as in Figure 2.

Each level in a CH corresponds to a node in a concept graph (CG). The concept graphs used here are distinct from the rule-based concept graphs in [4], which allow conditions attached to other attributes to affect the generalization of an attribute in a CH. The CHs in Figures 1 and 2 correspond to the more general representations in the CGs of Figure 3(a) and 3(b). We assume the use of unique names for nodes in a CG. If a name is used in multiple CGs, then the

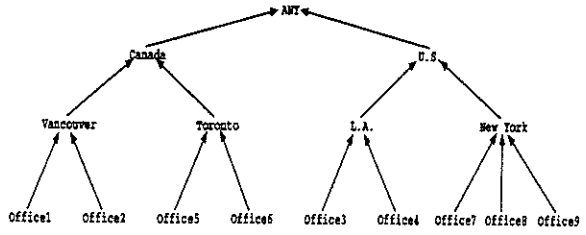


Figure 2: Alternative concept hierarchy

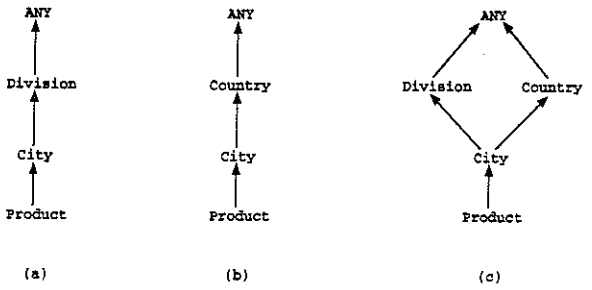


Figure 3: Similar CGs and a CGG

name represents the same partition of the domain for the attribute. Unfortunately, the CHs in Figures 1 and 2 do not identify any relationships that may exist between them. But given our assumptions regarding concept names, the CGs in Figures 3(a) and 3(b) clearly show that these CHs are similar. In this paper, we introduce concept generalization graphs (CGGs) as a method for guiding the learning process when an attribute may have multiple CHs. Figure 3(c) shows the CGG which has been constructed from the CGs in Figures 3(a) and 3(b).

The remainder of this paper is organized as follows. In the following section, we provide a formal definition of CGGs and describe how one is constructed. In Section 3, we introduce two algorithms for generalization using CGGs, and walk through a detailed example for one of them. We conclude in Section 4 with a summary.

2 Concept Generalization Graphs

2.1 Definitions

Given a set $S = \{s_1, s_2, \ldots, s_n\}$ (which could be the domain of an attribute or the Cartesian product of the domains of a set of attributes), S can be divided into parts in many different ways, e.g. $P = \{p_1, p_2, \ldots, p_n\} = \{\{s_1\}, \{s_2\}, \ldots, \{s_n\}\}$, $Q = \{q_1, q_2\} = \{\{s_1\}, \{s_2, \ldots, s_n\}\}$. Let D be the set of partitions of set S, and $\preceq$ be a binary relation defined on D such that $D_1 \preceq D_2$ if for every $d \in D_1$ there exists $d' \in D_2$ such that $d \subseteq d'$. The binary relation $\preceq$ is a partial order relation and $\langle D, \preceq \rangle$ defines a *partial order set*, from which we can construct a *concept generalization graph* $\langle D, E \rangle$ as follows. First, the vertices of the diagram are elements of D. Second, there is a directed arc from D_j to D_i (denoted by $E(D_j, D_i)$) iff $D_j \neq D_i$, $D_j \preceq D_i$ and there is no $D_k \in D$ such that $D_j \preceq D_k$ and $D_k \preceq D_i$. If we let $D_g = \{S\}$ and $D_l = \{\{s_1\}, \{s_2\}, \ldots, \{s_n\}\}$, then for any $D_i \in D$ we have $D_l \preceq D_i$ and $D_i \preceq D_g$. D_l is called the *least element* and D_g the *greatest element*.

For example, given $S = \{0, 1, 2, 3, 4, 5\}$, let $D = \{D_g, D_1, D_2, D_3, D_l\}$, where $D_g = \{S\} = \{\{0, 1, 2, 3, 4, 5\}\}$, $D_1 = \{\{0, 1, 2\}, \{3, 4, 5\}\}$, $D_2 = \{\{0, 1\}, \{2, 3\}, \{4, 5\}\}$, $D_3 = \{\{0, 1\}, \{2\}, \{3\}, \{4, 5\}\}$, and $D_l = \{\{0\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$, then we have the CGG $\langle D, E \rangle$ shown in Figure 4. We call the nodes (elements of D) *concepts*. The least element is the *lowest level concept* and the greatest element is the *highest level concept*. Two concepts, D_i and D_j, are *comparable* if either $D_i \preceq D_j$ or $D_j \preceq D_i$. That is, in the CGG, there exists a path containing both D_i and D_j. For each node D_i in CGG $\langle D, E \rangle$, we define $parents(D_i)$ to be all nodes D_j such that $D_i \preceq D_j$, $children(D_i)$ to be all nodes D_k such that $D_k \preceq D_i$, $in\_degree(D_i) = |children(D_i)|$, and $out\_degree(D_i) = |parent(D_i)|$.

2.2 Constructing a CGG

Given a CG for a domain, we call each level (including root and leaves) a *concept* and the nodes at each level the *instances* of the concept. Every concept is a partition of the instance set of lower level concepts, as well as a partition of the domain. If one or more CGs are given for a domain, all the concepts defined by these CGs comprise a set D of partitions of the domain. Based on the definition in the previous section, a pair $\langle D, \preceq \rangle$ defines a partial order set, and from this partial order set we can construct a CGG $\langle D, E \rangle$ associated with the domain. Each CG corresponds to a path from D_l to D_g in $\langle D, E \rangle$. Note that there may be some paths from D_l to D_g in $\langle D, E \rangle$ such that there is no corresponding CG. In this case, the CGG provides more information from which we can create additional CGs.

3 Generalization Using CGGs

3.1 Basic Idea

CGs define the mapping from the instance set of lower level concepts to that of higher level concepts. If we map the instances of one concept to the instances of another concept (i.e., replace specific values with generalized values), and the result is not satisfactory, we can either map the generalized value to a more gen-

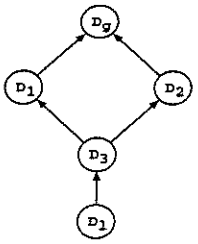


Figure 4: Partial order set diagram

eralized value, or we can use another CG to perform a similar generalization. In the latter case, we need to manage the generalization process to avoid unnecessary re-generalization, and to determine which intermediate generalized relations to store for possible future use. CGGs can be used to achieve both these goals. For example, given the CGG shown in Figure 5, if we have generalized from concept J to I, and we need further generalization from I to either H or F, then we do not need to keep the relation at J. However, if we have generalized to F and further generalization is needed, we need to keep the intermediate relation at I because we may need to generalize from I to H later on. If we have generalized from F to D and the results are still unsatisfactory, we can return to I and generalize from I to H. From the CGG we can see that it is unnecessary to generalize from H to G, because eventually this path will lead to D, which has already been found to be unsatisfactory.

3.2 Algorithms

Two methods for generalization using CGGs are the *path-based generalization* (PBG) and the *bias-based generalization* (BBG) algorithms. Assume that several CGs are given on the domain of an attribute which is to be generalized, and that an attribute threshold is given to control termination of the generalization process. From the given CGs, the CGG shown in Figure 5 has been constructed.

Using PBG, we select a CG (i.e., a path from D_l to D_g) from the CGG and generalize the relations at lower level concepts to relations at higher level concepts until the attribute threshold is reached (i.e., generalization is complete) or until the most general concept is the last remaining concept to be generalized. In the latter case, another CG (i.e., a different path from D_l to D_g) is selected and the above steps are repeated.

Using BBG, all concepts in the CGG are ranked a priori to determine the order in which the edges are traversed during generalization. Note that the rank of a concept must be consistent with partial ordering. That is, if $r(D_i)$ denotes the rank of D_i, consistency

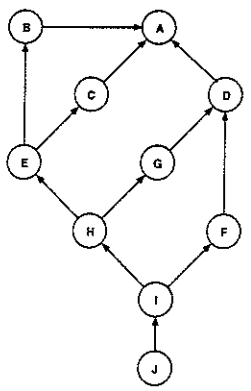


Figure 5: Concept generalization graph

means that if $E(D_i, D_j)$, then $r(D_i) < r(D_j)$.

3.2.1 The PBG Algorithm

Procedures *pbg* and *pbg_reconcile* of the PBG algorithm are shown in Figures 6 and 7, respectively. Given a CGG $\langle D, E\rangle$, function $nodes(\langle D, E\rangle)$ generates the set of all nodes contained in $\langle D, E\rangle$. Function $all\_paths(\langle D, E\rangle)$ generates the set of all paths from D_l to D_g contained in $\langle D, E\rangle$. Function $cardinality(PathSet)$ returns the number of elements in $PathSet$. Function $tuple\_count(C_h)$ determines the number of distinct tuples in a relation. Function $node\_count(CurrentPath)$ determines the number of nodes in the current path remaining to be generalized. Function $generalize(C_l, C_h)$ replaces the values at C_l with the values at C_h. It calls $tuple\_count(C_h)$ and returns *true* if the attribute threshold has been reached, and *false* otherwise. Any of the generalization algorithms presented in [2, 5, 3] may be used to implement this function.

We now walk through a detailed example of PBG. Consider the CGG, with least node J and greatest node A, shown in Figure 5. From this CGG, we first generate the set of all paths contained in $\langle D, E\rangle$ so that $PathSet = \{\langle JIFDA\rangle, \langle JIHGDA\rangle, \langle JIHECA\rangle, \langle JIHEBA\rangle\}$. The concepts in each path of $PathSet$ will be generalized until the attribute threshold is reached (i.e., success) or until there are no more paths to be generalized (i.e., failure). We set $CurrentPath$ to $\langle JIFDA\rangle$ (i.e., the first path in $PathSet$) and remove $\langle JIFDA\rangle$ from $PathSet$. We set C_h to J (i.e., the first node in $CurrentPath$) and remove J from $CurrentPath$. We call $tuple\_count(J)$ to determine whether the attribute threshold has been reached. If so, we return *success* and are done (i.e., the attribute is already generalized). Otherwise, the concepts of each node in $CurrentPath$ will be generalized until the attribute threshold is reached (i.e., success) or until there are

```
1. begin
2.     NodeSet ← nodes(⟨D, E⟩);
3.     PathSet ← all_paths(⟨D, E⟩);
4.     while cardinality(PathSet) > 0 do
5.         CurrentPath ← first path in PathSet;
6.         remove CurrentPath from PathSet;
7.         C_h ← first node in CurrentPath;
8.         remove C_h from CurrentPath;
9.         if tuple_count(C_h) ≤ threshold then
10.            return success;
11.        end if
12.        while node_count(CurrentPath) > 1 do
13.            C_l ← C_h;
14.            C_h ← first node in CurrentPath;
15.            remove C_h from CurrentPath;
16.            if generalize(C_l, C_h) then
17.                return success;
18.            end if
19.            pbg_reconcile(C_l, C_h);
20.        end while
21.        C_l ← C_h;
22.        C_h ← first node in CurrentPath;
23.        pbg_reconcile(C_l, C_h);
24.    end while
25.    return failure;
26. end
```

Figure 6: Procedure *pbg*

```
1. begin
2.     RemoveSet ← ∅;
3.     for each Path ∈ PathSet do
4.         if (C'_l C'_h) ∈ Path then
5.             RemoveSet ←
6.                 RemoveSet∪children(C'_h) from Path;
7.             remove children(C'_h) from Path;
8.             if node_count(Path) = 1 then
9.                 remove Path from PathSet;
10.            end if
11.        end if
12.    end for
13.    while cardinality(RemoveSet) > 0 do
14.        C_r ← first node in RemoveSet;
15.        remove C_r from RemoveSet;
16.        RemoveNode ← true;
17.        for each Path ∈ PathSet do
18.            if C_r ∈ Path then
19.                RemoveNode ← false;
20.            end if
21.        end for
22.        if RemoveNode then
23.            remove C_r from NodeSet;
24.        end if
25.    end while
26.    return;
27. end
```

Figure 7: Procedure $pbg\_reconcile(C'_l, C'_h)$

is only one node remaining in $CurrentPath$ (this will be the most general concept). We set C_l to C_h, set C_h to I (i.e., the new first node in $CurrentPath$), remove I from $CurrentPath$, and generalize from J to I. If $generalize(J, I)$ returns *success*, then we are done. Otherwise, we call $pbg\_reconcile(J, I)$. We remove J from all paths in $PathSet$ containing an edge from J to I and remove any paths in $PathSet$ which now contain only one node (again, this will be the most general concept). We now have $CurrentPath = \langle FDA \rangle$ and $PathSet = \{\langle IHGDA \rangle, \langle IHECA \rangle, \langle IHEBA \rangle\}$. We now generalize from I to F. If this fails, no removals are required from $PathSet$ because there are no paths with an edge from I to F. We now have $CurrentPath = \langle DA \rangle$ and $PathSet$ remains unchanged. We now generalize from F to D. If this fails, again no removals are required from $PathSet$. We now have $CurrentPath = \langle A \rangle$ and $PathSet$ remains unchanged. Since $CurrentPath$ now contains only only one node, we are finished with this path. We set C_l to C_h, set C_h to A, and call $pbg\_reconcile(D, A)$. *pbg_reconcile* removes D and all its descendants from the paths in $PathSet$. We now have $CurrentPath = \langle A \rangle$ and $PathSet = \{\langle A \rangle, \langle IHECA \rangle, \langle IHEBA \rangle\}$. Since $\langle A \rangle$ contains only one node, it is removed from $PathSet$. We now have $PathSet = \{\langle IHECA \rangle, \langle IHEBA \rangle\}$ which corresponds to a new CGG with least node I and greatest node A, as shown in Figure 8. We set $CurrentPath$ to $\langle IHECA \rangle$ and repeat the above procedure.

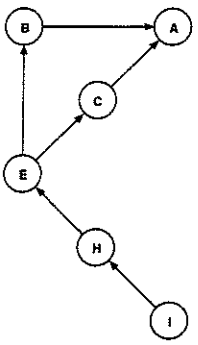


Figure 8: New concept generalization graph

A node is removed from $NodeSet$ when the value of the corresponding concept is no longer needed. For example, if the original data in the database is no longer needed, node J can be removed. Similarly, when an intermediate generalized relation is no longer needed, the node for the corresponding concept can be removed.

3.2.2 The BBG Algorithm

Procedures *bbg* and *bbg_reconcile* of the BBG algorithm are shown in Figures 9 and 10, respectively. Given a CGG $\langle D, E \rangle$, function $all\_edges(\langle D, E \rangle)$ generates the ordered set of all edges contained in $\langle D, E \rangle$ as follows. First, edges are ordered by the rank of the higher level concept (from lowest to highest), Second,

where multiple edges have the same higher level concept, these edges are ordered by the rank of the lower level concept.

```
1.  begin
2.      NodeSet ← nodes(⟨D, E⟩);
3.      C_h ← lowest ranked node in NodeSet;
4.      if tuple_count(C_h) ≤ threshold then
5.          return success;
6.      end if
7.      EdgeSet ← all_edges(⟨D, E⟩);
8.      while cardinality(EdgeSet) > 0 do
9.          CurrentEdge ← first edge in EdgeSet;
10.         remove CurrentEdge from EdgeSet;
11.         C_l ← first node in CurrentEdge;
12.         C_h ← second node in CurrentEdge;
13.         if generalize(C_l, C_h) then
14.             return success;
15.         end if
16.         bbg_reconcile(C_l, C_h);
17.     end while
18.     return failure;
19. end
```

Figure 9: Procedure *bbg*

```
1.  begin
2.      RemoveSet ← C'_l;
3.      for each Edge ∈ EdgeSet do
4.          C_s ← second node in Edge;
5.          if C'_h = C_s then
6.              C_f ← first node in Edge;
7.              RemoveSet ← RemoveSet ∪ C_f;
8.              remove Edge from EdgeSet;
9.          end if
10.     end for
11.     while cardinality(RemoveSet) > 0 do
12.         C_r ← first node in RemoveSet;
13.         remove C_r from RemoveSet;
14.         RemoveNode ← true;
15.         for each Edge ∈ EdgeSet do
16.             C_f ← first node in Edge;
17.             if C_r = C_f then
18.                 RemoveNode ← false;
19.             end if
20.         end for
21.         if RemoveNode then
22.             remove C_r from NodeSet;
23.         end if
24.     end while
25.     return;
26. end
```

Figure 10: Procedure $bbg\_reconcile(C'_l, C'_h)$

4 Conclusion

We presented the path-based generalization and bias-based generalization algorithms for attribute-oriented induction using CGGs. Future research will focus on developing heuristics for improving efficiency of the algorithms. Heuristics are required for selecting and ordering the paths when using PBG, and for determining the rank of a node when using BBG.

References

[1] C. L. Carter and H. J. Hamilton. Fast, incremental generalization and regeneralization for knowledge discovery from databases. In *Proceedings of the 8th Florida Artificial Intelligence Symposium*, pages 319–323, Melbourne, Florida, April 1995.

[2] C. L. Carter and H. J. Hamilton. A fast, on-line generalization algorithm for knowledge discovery. *Applied Mathematics Letters*, 8(2):5–11, 1995.

[3] C. L. Carter and H. J. Hamilton. Performance evaluation of attribute-oriented algorithm for knowledge discovery from databases. In *Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence (ICTAI'95)*, pages 486–489, Washington, D.C., November 1995.

[4] D.W. Cheung, A.W. Fu, and J. Han. Knowledge discovery in databases: a rule-based attribute-oriented approach. In *Lecture Notes in Artificial Intelligence, 8th International Symposium (ISMIS'94)*, pages 164–173, Charlotte, North Carolina, 1994.

[5] J. Han. Towards efficient induction mechanisms in database systems. *Theoretical Computer Science*, 133:361–385, October 1994.

[6] J. Han, Y. Cai, and N. Cercone. Knowledge discovery in databases: an attribute-oriented approach. In *Proceedings of the 18th International Conference on Very Large Data Bases*, pages 547–559, Vancouver, August 1992.

[7] J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. *IEEE Trans. on Knowledge and Data Engineering*, 5(1):29–40, February 1993.

[8] H.-Y. Hwang and W.-C. Fu. Efficient algorithms for attribute-oriented induction. In *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining (KDD-95)*, pages 168–173, Montreal, August 1995.

[9] T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1982.

INTEGRATIVE FRAMEWORKS FOR DECISIONMAKING IN RESOURCE MANAGEMENT

John Bolte
Bioresource Engineering Department, Oregon State University
boltej@ccmail.orst.edu

Abstract

Simulation has historically played a substantial role in the development of analysis tools for agricultural and resource management systems. More recently, artificial intelligence-based technologies have been widely applied in to these systems, particularly in the development of expert systems and heuristic approaches. Additional approaches based on biological metaphors have similarly become commonly used analysis strategies. The maturing of these technologies, coupled with the emergence of object-oriented systems, has provided opportunities to integrate these approaches into decision tools relying on multiple technologies and information sources. These various approaches are reviewed, and a framework for integrating multiple paradigms using an object-based strategy is presented.

1. INTRODUCTION

A number of analysis methodologies, derived from simulation, artificial intelligence, and biological research, have reached a level of maturity which allows their widespread use for decision support for management of complex ecological systems, such as agriculture and natural resource management. Because of the frequently high level of interaction between components of the systems, often encompassing both ecological as well and socioeconomic complexity, analysis of these systems may require the integration of a range of analysis methodologies. A review of several domain-independent methodologies, and the development of an integrative framework based supporting the development of decision systems utilizing these technologies, is provided below.

2. SIMULATION

Mathematical systems models and computer simulations are commonly used by scientists, agriculturalists and resource managers to allocate resources, enhance production management, and understand biological processes. These simulators are constructed in a series of steps as follows: 1) identify the most important features of the system of interest, 2) develop an abstract, conceptual framework describing the system in terms of fundamental component description and interaction between these components, 3) formalize the conceptual framework by developing concrete representations of each of the system components, usually in the form of mathematical representations of the dynamics of that system component with respect to other components of the system, resulting in the formulation of a simulation model,
4) implement the simulation model in a computer representation to execute and test the model's predictive capabilities.

Simulation models, then, become a vehicle for representing in abstract terms a slice of reality consistent with our specific interests in enhancing our understanding of our world. *Representation* here is the key word: simulation models must provide us with an abstract representation of reality. Simulation models can provide decisionmakers with these abstractions to provide *predictive* capabilities for analyzing system response to modifications of the driving factors of the system. By themselves, model provide no *prescriptive* capabilities for determining optimal management strategies; however, these can be provided through coupling with other strategies, as described below.

2. RULE-BASED SYSTEMS

Much of the current effort in developing representational strategies for application in agriculture and natural resource management has focused on modeling the processes by which human experts think and learn. The majority of these efforts has involved developing heuristic systems in which human experience and knowledge is captured in the form of "rules of thumb". These rules are then manipulated using an inference engine typically employing an backward- or forward-chaining deductive strategy to connect the rules and input data collected during an interview session to some set of predefined recommendations or conclusions. Such an approach has been applied to developing rule-based expert systems in a wide range of application domains. The power of

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/395

these systems is largely due to the expressive nature of rules and their ability to efficiently process symbolic information expressing logical relationships. Because substantial quantities of human knowledge can be fit to this paradigm, these systems have generally been successful, particularly in the realms of diagnosis and planning.

Current rule-based systems have a number of limitations. First, expert behavior is only partially explained in terms of inferential reasoning; a large component of human expertise arises from being able to recognize patterns in input situations and react accordingly. Second, these systems lack capabilities for generalizing knowledge to respond to new situations; if the problem's solution space lies outside that specifically defined by the knowledge engineer during the construction of the knowledgebase, the system is generally unable to respond adequately. Finally, rule-based systems lack robust capabilities for automatic learning; the knowledgebases must be hand-crafted and tend to be static and unable to readily incorporate new knowledge as it becomes available.

Despite these limitations, rule-based systems provide a powerful representational approach. Current systems have generally been developed as stand-alone applications; however, the combination of rule-based strategies for higher-level interfacing, diagnosis and planning, coupled with simulation models for lower-level analyses, provides opportunities to substantially enhance the utility of simulation models in a decision support context.

3. NEURAL SYSTEMS

Neural networks offer an alternative to rule-based systems for developing robust intelligent applications. Although these systems have been analyzed for several decades (Rumelhart and McClelland, 1987; Rosenblatt, 1961), it has only been within the past decade that our understanding of neural networks has allowed their successful application to a number of problems in agriculture and resource management. While a number of basic network architectures have emerged, all are based on the same general concept: connecting a large number of very simple processing elements in a massively parallel, highly distributed processing environment. The human brain offers a biological analogy to neural networks. At a simplified level, the brain operates as an extremely large collection of very simple processing elements (neurons) which are highly interconnected and operate in parallel. Individual neurons function as excitatory threshold devices consisting of input connections (dendrites), a cell body containing chemical transmitters which "fire" when the sum of the input excitations exceed some threshold value, and an axonal body serving as a transmission vehicle to send the neuron's output to other neurons through synaptic interfaces. Artificial neural networks operate in a similar fashion. Input connection strengths, calculated according to the activation level of the sending neurons and a connection weight, are summed for each neuron. The sum is fed into an activation function to determine the activation level of the receiving neuron. The activation function may take many forms, but typically a threshold or sigmoidal function is used. Because of the interconnected nature of neural architectures, these calculations proceed in parallel.

Feed-forward (backpropagation) networks are the most commonly used neural networks. These networks can learn to recognize patterns and generalize information available in a training set. Generally, feedforward networks are arranged in three layers: an input layer, a "hidden" layer, and an output layer. The input layer connects the network to the problem inputs. and provides a mechanism for stimulating the network and generating a response. The output layer provides a structure for indicating the final state of a network response, and maps directly into domain-specific problem solutions after input stimulation. The hidden layer serves as a buffer between the input and output layers and acts a pattern repository for the networks.

A primary advantage of feedforward networks is that the can be trained to recognize general patterns between input/output sets. By applying a learning algorithm to the network while exposing it to known input/output combinations, the net can automatically structure itself to reflect general knowledge of the relations mapping inputs to outputs. Thus, these networks operate as general pattern recognizers; however, because the network inputs and output can be arbitrarily mapped to a wide range on information representations, their pattern recognition capabilities can be applied to problems far beyond those typically considered as pattern matching problems. Feedforward networks have been successfully applied to problems domains including handwriting interpretation, image processing, speech recognition, machine diagnostics, stock market analysis, and many others. They provide a readily constructable, highly robust mechanism for addressing problem domains where a given set of input conditions must be mapped into some output recommendation or result.

4. GENETIC ALGORITHMS

Genetic systems are based on the biologically-inspired metaphor of representing problem

solutions as populations of parameters or symbols residing on representational structures analogous to biological chromosomes. Strategies implementing this approach are collectively referred to as genetic algorithms (GA). GA's are fairly recent adaptive optimization technique that attempts to overcome the problems of traditional nonlinear optimization methods (Davis 1992, Michalewicz 1992). The basic concept behind GA's is deceptively simple: a given set of conditions defining a potential solution to a problem are described as a single chromosomal entity, with the individual components of the solution (model parameters, conditions, etc.) corresponding to genes on the chromosome. A population of these solutions, representing initially a diverse range of potential solution options, are generated. Through a process analogous to natural selection (survival of the fittest), pairs of individuals that provide competitive solutions are combined through a crossover event to produce offspring contain characteristics of both parents. Repeated throughout the population, this reproduction of the most fit individuals results in a new generation of individuals, which are again evaluated based on a fitness measure and the most fit enter another crossover cycle. After successive generations, overall fitness of the population tends to increase, dramatically in many cases, and extremely fit individuals containing solution parameters that provide high quality solutions to the target problem generally arise.

Genetic operators of crossover and mutation provide adaptive, convergent solutions by combining building blocks, or *schema*, analogous to coadapted sets of alleles, representing good solutions from diverse chromosomes. The best individual of the final population will likely represent a highly evolved solution to the problem being solved. The link between the problem and the population of chromosomes is an evaluation function which provides a measure of the performance (i.e., fitness) of each chromosome in relation to the problem. The evaluation function thus plays an analogous role to the environment which a natural organism may encounter. Further, the essential features of a traditional GA are generic, in that the main difference among different problems is the evaluation function. That is, manipulation of the populations and reproductive mechanisms are typically problem independent.

Nonlinear optimization involves search of potentially complex, high-dimensionality solution spaces for maxima and/or minima. Traditional approaches have revolved around hill-climbing techniques, which tend to be problematic in regions of solution space with frequent local solutions which are not globally optimal. GA's are less susceptible to problems involving local convergence because they simultaneously search many regions of parameter space, and mutation provides opportunities to explore new regions not otherwise available. In this regard, they are similar to simulated annealing approaches, which use a statistical search to avoid local minima. However, unlike simulated annealing and other pure statistical approaches, GA's learn from previous iterations via the mechanism of crossover. Also, because GA's are based on simultaneous evaluation of populations of solutions, they are more amenable to parallel execution.

Genetic algorithms show promise as a component of a broader decision system in two main respects. First, they have been used to provide general and relatively efficient parameter estimation capabilities for the complex, nonlinear models frequently encountered in biological domains (Bolte and Nath, 1995; Sequeira and Olson, 1995). Second, because GA's provide robust nonlinear optimization strategies in high-dimensionality solution spaces with complex topologies, they provide the prospect of efficient search for optimal solutions to multiobjective resource management problems which are becoming increasing commonplace. Additionally, because genes can encode either numeric or symbolic information, genetic representation strategies can potentially interact with simulation models as well as heuristic systems. However, computational requirements for GA's (and other nonlinear optimization strategies) may limit there widespread use in decision support.

5. OBJECT-BASED SYSTEMS

Object-based, or *object-oriented* systems, are built around the concept of a relatively self-contained entity, the *object*. An object may contain both data reflecting its internal state and functional capabilities, or *methods*. Thus, systems becomes collections of diverse objects, each providing resources and capabilities to the system as a whole to accomplish some requirement. Because objects can mirror reality at both conceptual and implementation levels, object-based software tends to be easier to design and maintain than conventional software. High level support for sophisticated representation, coupled with additional supporting language features, has moved object-based systems into the forefront of current programming paradigms.

There are many implementations of object-oriented languages. A review of these languages, and their respective capabilities, is beyond the scope of this paper. There are however, a few key concepts common to an

object-based approach which merit mention. These include *encapsulation*, *message passing*, *dynamic binding*, and *inheritance* (Budd, 1991; Meyer, 1988; Nerson, 1992). These features work together to provide system designers with the ability to abstract and define interfaces between objects, and consequently the ability to develop standardized frameworks for the construction of interacting, modularized components. This allows the development of frameworks in a manner which provides for integration of diverse representational strategies.

6. FRAMEWORKS FOR INTEGRATION

Decisionmaking in agriculture and resource management is becoming increasing complex, because it often involves composites of biological, physical, environmental, and economic interactions. Consequently, tools to support decisionmaking must be able to integrate information from numerous sources, be they simulation models, expert systems, databases, or any of a number of other types. Decision support software provide capabilities to assist in assembling, integrating, and utilizing knowledge from these diverse sources to make decisions about complex processes. This is a non-trivial task from a software engineering perspective.

Construction of such decision systems can be greatly enhanced through the utilization of high-level object frameworks. An object-based paradigm provides substantial capabilities for the requirements these frameworks by providing for virtualized interface descriptions, dynamic binding to these interfaces, and an extremely modular design approach. These fundamental object constructs provide the building blocks for constructing integrative frameworks for decision support. Such a system should additionally provide a number of high-level services, include 1) standardized public interfaces defining object access and action initiation, 2) high-level communications capabilities for components of the system to communicate with other components in a non-specific manner, 3) standardized methodologies for collecting and transferring information between components of the system, 4) support for standard representational constructs, including simulation constructs, inferencing, and other representational paradigms which are domain independent, and 5) mechanisms for synchronizing the flow of execution among system components. The object paradigm provides potentially useful capabilities in all of these areas. The standardization of public interfaces is a central OOP concept, and is readily implemented through the definition of generic objects with virtualized public interfaces. Virtualizing the interface allows direct communication to specific subclasses without any knowledge of the type of that subclass, a critical requirement for the development of high-level, domain-independent decision support frameworks. Similarly, high-level communications between objects can be accomplished through a similar interface, in a manner which precludes a requirement that caller's have specific knowledge about those objects retrieving the communications. Additionally, interobject communication can be provided by a central blackboard maintained by the system. Synchronization and data collection can also be handled at a high level, with subclasses inheriting much of the behavior necessary to handle parallel execution between objects, data flow and communication.

An important conceptual component of such a system is an *agent*. An agent is a intelligent entity which can provide diagnostic, control or planning capabilities with the system. In a resource management context, agents in the system might embody knowledge about the resources in question, utilizing information derived from models and datasets to make suggestions regarding appropriate management tactics under different circumstances. Agents are also useful for guiding the trajectory of complex systems.

As an example of such a system, we have developed a decision systems for aquaculture production facilities management, where a facility consists of multiple ponds, fish lots, and various supporting resources. At the lowest level, the system relies on simulation models to make predictions about the state of the facility through time. Individual components of the system are dynamically created (instantiated) from object templates reflecting the item in question, include fish ponds, fish lots, pumps, water conditioning units, and other entities important in the operation of actual facilities. A high-level, domain-independent framework provides most of the services described above, including synchronization of simulation objects, interobject messaging, a central bulletin board for addition time-independent, non-directed communication, and centralized data sharing and statistics. The framework, written in C++, takes full advantage of the object paradigm. Within the system, agents are defined to provide expertise in areas of water quality analysis, assessment of fish performance, distribution and stocking of fish lots, pond diagnostics, and other areas roughly paralleling human experts in the field. Analyses of a particular facility are accomplished by running simulations of the system. In additional to "normal" simulation objects, a series of agents may be run in parallel with the simulation to monitor the system state at specific times, obtaining information about specific system components through the standardized interobject

communication facilities provided by the system. The agents can take prescriptive action when problems are identified, or present alternatives to the user to pursue. The agent can also make decisions on the allocation of resources to determine near-optimal pathways through dynamic solution space in these complex environments. Upon conclusion of a simulation, the agents can be queried for suggestions on improving the management strategy employed. An additional interesting benefit is that the effects of management "intensity", i.e. how intently the agents monitor the system, can be examined by varying the frequency with which agent observes the system and action is taken. This system has successfully demonstrated the usefulness of an integrative framework in allowing a mixture of representational paradigms to coexist in a synergistic manner.

7. THE FUTURE

The requirement for tools to synthesize and integrate knowledge sources for agriculture and resource management continues to increase. Fortunately, technologies for dealing with the problems encountered are becoming increasingly available. Many aspects of representation and analysis in these management systems can be abstracted to system-level services and provided through integrative frameworks. Similarly, interface between the components of these can be defined on an abstract basis and dynamically bound at runtime. Indeed, the current trend is towards an object model encompassing these concepts at the operating systems level. A good example of this is the Common Object Model, a cross-platform standard which has been adopted by major operating system vendors as a fundamental part of the operating environment. Such standards will increasingly provide the mechanism for module interface definition, data sharing, and interobject communication. These, coupled with framework-level support for representation strategies and agent-like tools for monitoring and analysis will provide powerful environments for the development of decision support tools.

References

Bolte, J.P. and S.S. Nath. 1995. Decision support for pond aquaculture: Parameter estimation techniques. *PD/A CRSP 1995 Annual Report*. Oregon State University, Corvallis, OR.

Bolte, J.P., J.A. Fisher and D.H. Ernst. 1993. An object-oriented, message based environment for integrating continuous, event-driven and knowledge-based simulation. *Proceedings: Application of Advanced Information Technologies: Effective Management of Natural Resources*. ASAE. June 18-19, Spokane, WA.

Budd, T. 1991. *An Introduction to Object-Oriented Programming*. Addison-Wesley, Reading, Massachusetts.

Davis, L. (Ed.), 1992. *Handbook of genetic algorithms*. Van Nostrand Reinhold, New York. 385pp.

Meyer, B. 1988. *Object-Oriented Software Construction*. Prentice Hall, New York.

Michalewicz, Z., 1992. *Genetic algorithms + Data Structures = Evolution Programs*. Springer-Verlag. 250pp.

Nerson, J.M. 1992. Applying object-oriented analysis and design. *Communications of the ATM*. 35:9 63-74.

Rosenblatt, F. 1961. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanics*. Spartan Books. Washington DC.

Rumelhart, D.E. and J.L. McClelland. 1987. *Parallel Distibuted Processing: Explorations in the Microstructure of Cognition*. Vol1 1. Foundations. MIT Press. Cambridge, MA.

Sequeira, R.A. and Olson, R.A., 1995. Self-correction of simulation models using genetic algorithms. *AI Applications*, 9: 3-16.

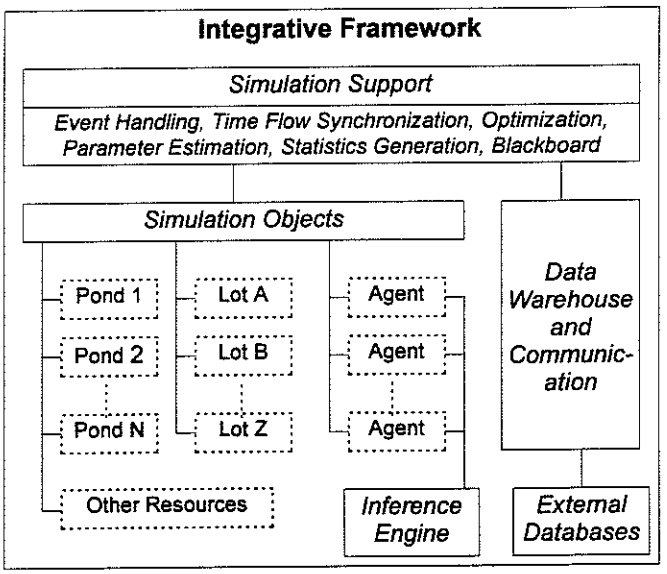


Figure 1. Schematic of Integrative Framework for Aquaculture Facility Management

A Dynamic Diagnostic Expert System for Citrus Diseases Using Visual Basic

R. M. Peart, Graduate Research Professor, Agricultural and Biological Engineering Dept.; Univ. of Florida, Institute of Food and Agricultural Science (IFAS), Gainesville, FL,
L. W. Timmer, Professor of Plant Pathology, Citrus Research and Education Center, Univ. Of Florida, Institute of Food and Agricultural Science (IFAS), Lake Alfred; FL and
L. W.. Miller, Staff Engineer, Agricultural and Biological Engineering Dept., Univ. of Florida, Institute of Food and Agricultural Science (IFAS), Gainesville, FL 32611 USA.

Abstract

A prototype expert system was developed for use by citrus growers in deciding when to apply fungicide to reduce damage from melanose and Postbloom Fruit Drop (PFD). These are fungal diseases which develop in warm, humid conditions and spread with rain splash.

We used Visual Basic to develop a Windows program that growers and managers could use to enter data and receive advice about the danger from the diseases and possible treatment needed. Color pictures were used to show the effects of the disease and the symptoms. A database system was incorporated so the user could keep records on the symptoms and environmental conditions through time to help in future decisions. The program is still a prototype, and is not released for use by the University of Florida, Institute of Food and Agricultural Science.

Introduction

Melanose and Postbloom Fruit Drop (PFD) are caused by fungi and can be costly. PFD can be predicted in time to prevent major losses with a model developed by the second author. Data needed are rainfall in the past 5 days and number of diseased blooms per tree.

Melanose:

Melanose can be distinguished from citrus rust mite injury by the raised pustules it produces (rust mites cause a smooth blemish). Areas of the fruit often crack and produce a pattern known as mudcake melanose.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/400 ©1996 FLAIRS

Fungicide spray treatments are most effective if applied frequently. Copper fungicides are the most widely-used for controlling melanose in most countries, but they are only effective if applied to the fruit surface. Pruning to control melanose is not feasible for older trees. Melanose is not a problem on young trees (less than about 6 years old), because they do not have as many dead twigs, which are hosts for fungal reproduction. The timing and frequency of spray treatments depends on local climatic conditions.

Fruit are susceptible to melanose infection for about 12 weeks after petal fall, but the later the infection occurs, the smaller the resulting pustules. Therefore, melanose infection periods that occur after late June usually do not affect the fruit except in years of a late bloom.

Dead wood in the tree canopy provides a suitable environment for the production of new fungal spores. Therefore, younger groves are less likely to host the disease. Older groves (older than 6 years), however, are more likely to have the fungus present.

Groves with a history of melanose infection tend to have many spores of the fungus that can cause infection during a rainy period..

As most other organisms do, fungi adapt to their environments. Therefore, repetitive use of the same type of fungicide may result in the development of a resistance to the fungicide.

While all species are susceptible to attack from melanose, grapefruit and lemons are particularly sensitive to it. As the growing process slows, fruit will become more and more resistant to infection by melanose.

Postbloom Fruit Drop (PFD)

Postbloom Fruit Drop has occasionally caused losses in yield of 50-90%.

The fungus (a strain of Colletotrichum acutatum) infects citrus flowers and produces orange- to peach-colored spots on the petals or affects entire flower clusters when conditions are favorable.

Subsequently, a persistent calyx (button) develops and is surrounded by distorted leaves with enlarged veins and no fruit develops. Symptoms of the fungus develop within three to four days of innoculation.

Through research at the University of Florida (L. W. Timmer and S. E. Zitko, 1993), it has been determined that one can predict the number of blossoms that will be affected by PFD in 4-8 days if one knows the amount of rainfall during the last 5 days and the number of blossoms presently infected on 20 trees. This "model" is the basis for the PFD Guide in this program.

Postbloom fruit drop of citrus was first described in Belize in 1979. After its discovery in Belize, PFD was reported in Brazil, Argentina, Dominica, and Colombia. It has been observed in most other humid, tropical citrus-growing areas in Central and South America and the Caribbean. Disease severity has varied greatly with location and year, but occasionally growers have reported yield losses of 50-90%.

The optimum temperature for spore germination is about 73 F (23 C), with infection occurring in 12-18 hrs. Flowers are resistant in the pinhead to round bud stage, but they become susceptible when they begin to elongate, and are highly susceptible once open. Dispersal of the pathogen appears to be primarily by rain splash and to a lesser extent by windblown rain. Of the environmental variables studied, only rainfall amounts were highly important in determining disease severity. Temperature and relative humidity were not major factors in explaining disease incidence, perhaps because temperatures during the periods of observation were mainly in the range of optimum for the organism..

Research by Timmer and Zitko suggests that PFD should be treated if the model predicts 20% or more infection 4 days hence, and if there are sufficient blooms to set a crop. The user can see pictures of blooms affected by PFD and healthy blooms in this program.

Use of the Program:

This program may be used in one of two different modes - the record mode, where data is entered each day and saved in a database, or the direct entry mode, where current data and 5-day rainfall accumulation are entered on a one-time basis for recommendation at the current time.

For the record or database mode, the user chooses Edit Records at the bottom of the Introduction screen, Figure 1. The record-keeping screen appears next, Figure 2, and the user enters daily values for Date, Number of Bad Blossoms, and Rain Amount plus the other information on the form. The Back to Intro button takes the user back to the Introduction screen, where either Melanose or PFD are selected.

If PFD is selected, the screen in Figure 3 appears. On this screen, the user selects Data Source. If it is Record Entry, the Block ID is entered and the Calculate Loss button selected, and the data previously entered into the database is used to determine a recommendation based on the PFD rules, including the model described previously. If the Direct Entry mode is selected, the user completes the form on this screen, then clicks on Calculate Loss, and the recommendation, such as shown in Figure 4 appears. The brief prediction of the number of bad blooms in 3-4 days will be expanded in the future to include an appropriate recommendation about treatment. We feel the ability to easily print out this recommendation is an important feature, which is incorporated into this screen.

If Melanose is selected, another screen appears, similar to Figure 3.. As before, the user selects Data Source. If it is Record Entry, the Block ID is entered, and the data previously entered into the database is used to determine a recommendation based on the Melanose rules. If it is Direct Entry, the user completes the form, then selects Calculate Loss, and a recommendation appears.

Program Design and Development:

The program was developed by the third author, Mr. Miller, using Visual Basic, Ver. 4.0, and the literature and updated advice of the second author, Dr. Timmer. Visual Basic is a modern structured

programming language, much more advanced than the original MS-BASIC, from which many programmers received their early experiences. Importantly, it includes the ability to produce Windows screens with input and output boxes, text, and pictures or other graphics for pleasant interaction between the user and the program.

Knowledge Base:

Acquiring the knowledge base from the second author, Dr. Timmer, was simple because of the model he had already developed in his research. This quote from Timmer, et al., (1993) shows the "rule-type" thinking that went into the model:

> "A variety of 'if, then' statements were tested with previous disease, rainfall, and leaf wetness. Because low previous disease resulted in a low percentage of infection even when rainfall was high, the statement, 'IF TD <= 75, THEN TD = 0', was included in the models."

The model used in this Citrus Pathogen Guide is:

% Damaged Blooms 3 to 4 days from today =

-7.15 + 1.28*(TD)^(1/2) + 0.44*(R*100)^(1/2),

where TD = Total no. of diseased flowers on 20 trees today, and
R = Amount of rainfall in past 5 days, mm.

If this percentage is above 20%, the grower is advised to apply fungicide, unless it has already been applied in the past 14 days.

Future Improvements:

One improvement, already mentioned is the expansion of the recommendation for PFD, to give more detailed advice about treatment, taking into account the "over 20%" rule, as well as reminding the user that if fungicide has been applied within the past 14 days, it is probably too soon to re-apply, unless rainfall has been intense enough to wash off the old treatment.

A second improvement would delete the need to re-enter redundant data. For example, when the Block Identification has been entered and the Percent Loss Last Year and Age of Trees have been entered, this should not need to be entered again for the same block.

Acknowledgements:

This is a prototype program, not released for general use yet. It has been developed by University of Florida IFAS researchers with initial programming help from 2 Summer Science Training Programming students, John King, Daytona Beach; and Eric Kline, Pinellas County. The IFAS group included the authors, with support from Dr. J. D. Martsolf, Meteorologist, Horticulture Dept., Gainesville, and Mr. Pete Spyke, Arapaho Citrus Management, Inc., Ft. Pierce, FL.

References:

Timmer, L. W. and Zitko, S. E. (1993). Relationships of Environmental Factors and Inoculum Levels to the Incidence of Postbloom Fruit Drop of Citrus. Plant Disease, volume 77 , (5), 501-504.

Whiteside, J. O., S. M.Garnsey, and L. W. Timmer, (1988). Compendium of Citrus Diseases. American Pathological Society Press, St. Paul, MN.

Whiteside, J. O. (1980). "Timing of Fungicide Spray Treatments for Citrus Melanose Control." Proc. Fla. State Hort. Soc. Vol. 93. 21-24.

Figure 1. Introduction Screen for the Citrus Pathogen Guide

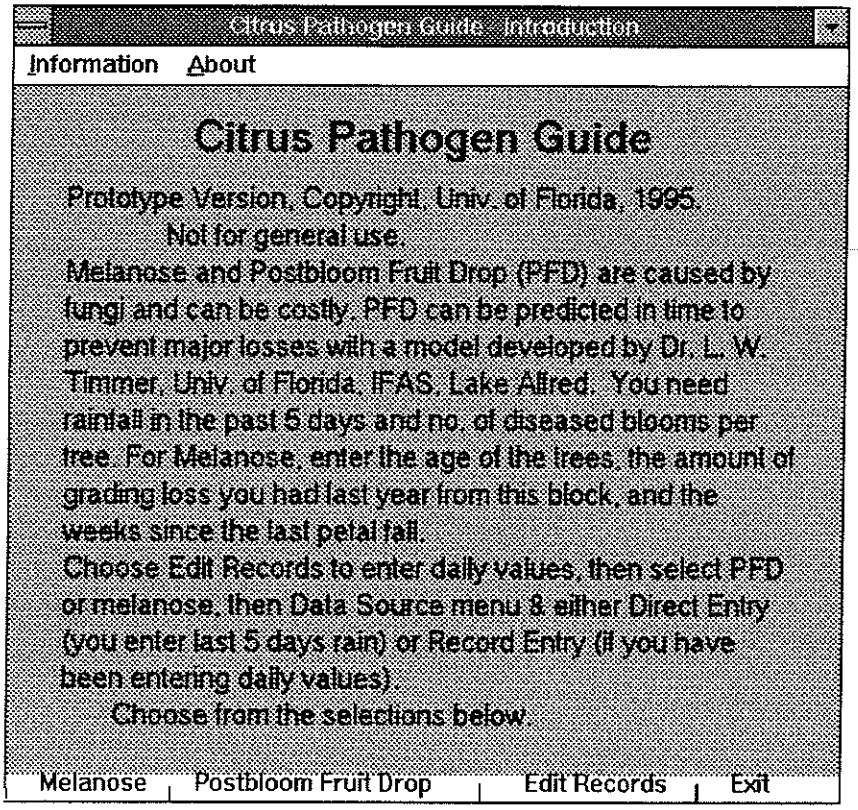


Figure 2. Record or Database Screen, Citrus Pathogen Guide

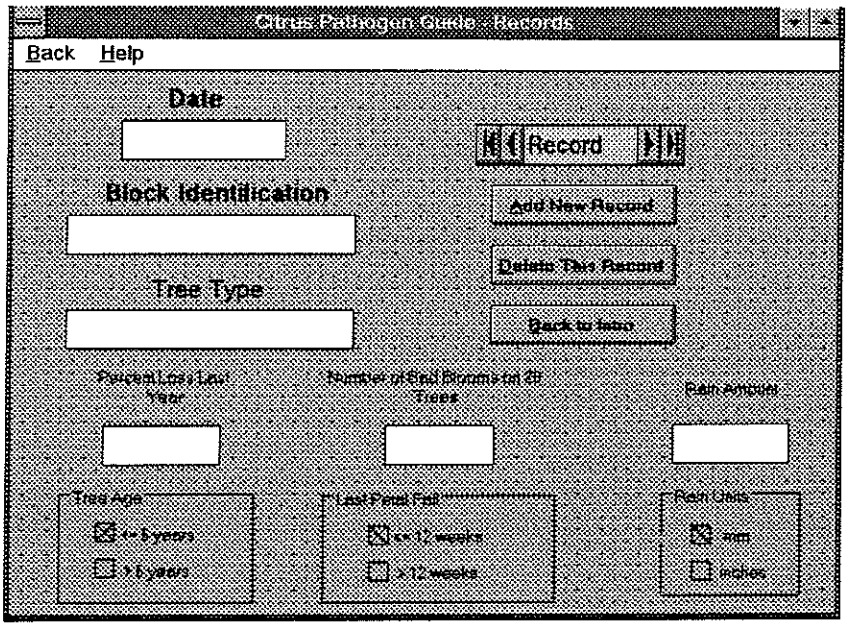


Figure 3. Screen for Postbloom Fruit Drop, Citrus Pathogen Guide

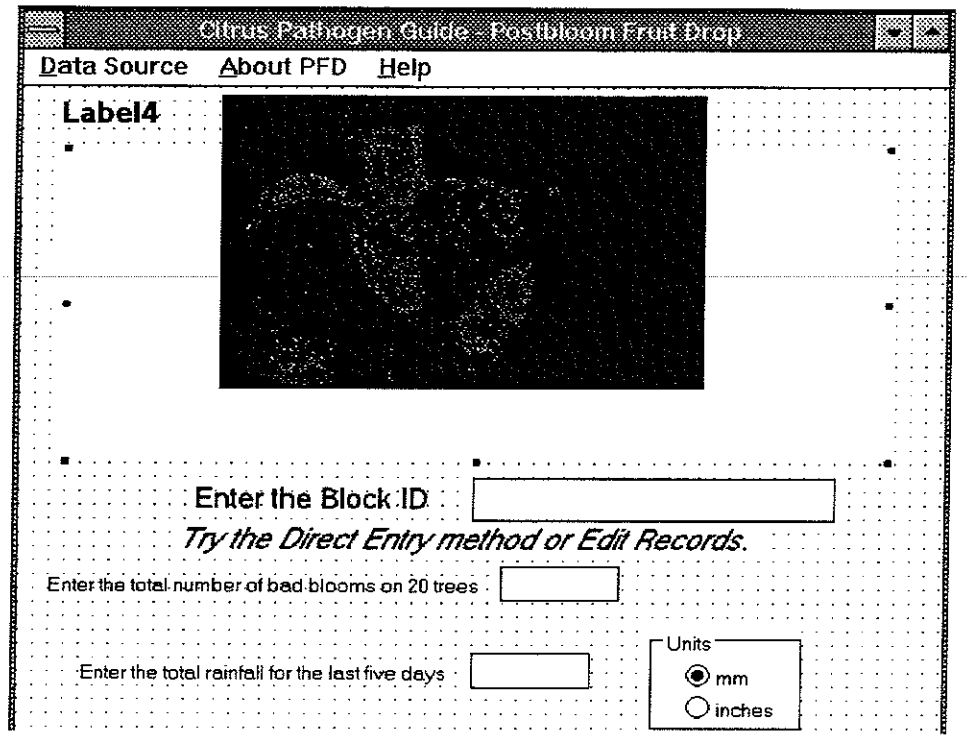
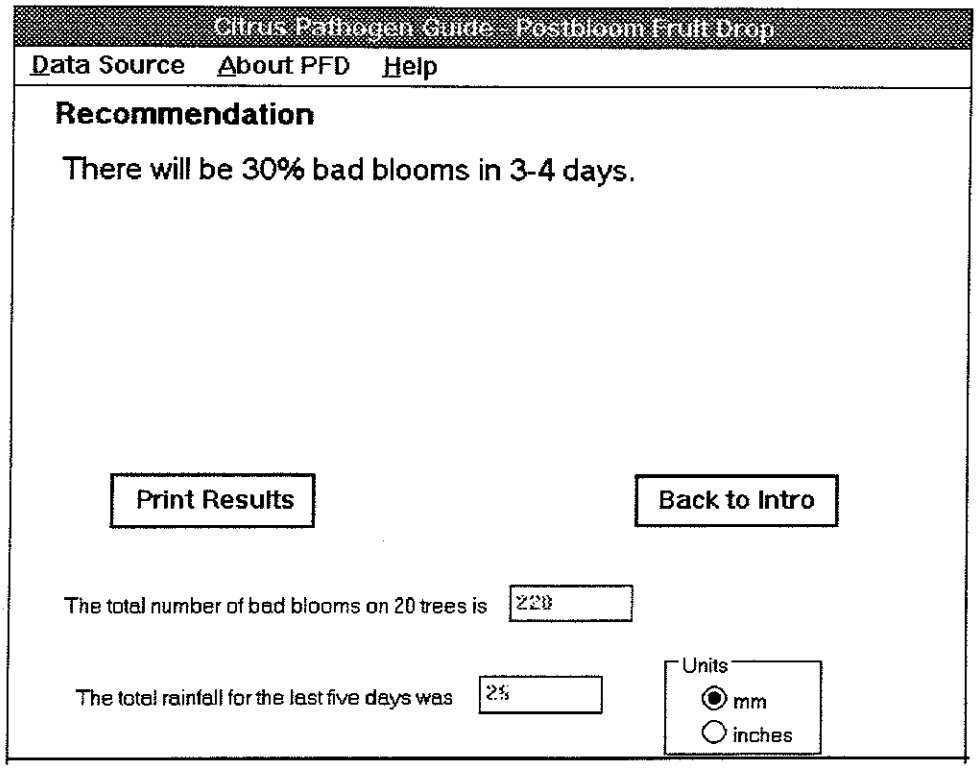


Figure 4. Recommendation Screen for Postbloom Fruit Drop



SGML versus Semantic Data Models in building Digital Agricultural Libraries

David B. Williams, University of Florida
dbw@gnv.ifas.ufl.edu
Howard W. Beck, University of Florida
hwb@agen.ufl.edu

ABSTRACT

A comparison was performed between Standard Generalized Markup Language (SGML) and a semantic data model (SDM) as vehicles for storage and knowledge representation of agricultural documents. A digital library is being built at the University of Florida using a SDM called Candide. SGML files are converted to objects and stored in the Candide database. The SDM is as expressive as SGML in storing extension information, provides robust data management functions and is capable of content based searching. As a knowledge representation language, SGML has strong syntactic expressiveness with a weak semantic interpretation while SDMs are strong in both areas.

INTRODUCTION

Documents, including bulletins, circulars, fact sheets, technical papers and handbooks, are a primary source of agricultural information. Extension documents embody a decades-old archive of agricultural knowledge. As these documents are converted to electronic form, the resulting electronic library provides not only more efficient document storage and retrieval techniques, but also potential for greater utilization of knowledge contained in the documents. The document analysis process is a form of knowledge acquisition that converts information contained within a document to a structural knowledge representation language. This enables greater utilization of the knowledge contained within the documents and improves the document search and retrieval process. Ultimately, knowledge gleaned from documents can be used by higher-level reasoning systems, such as qualitative and quantitative models or expert systems.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/405

Currently, electronic text handling is dominated by the use of Standard Generalized Markup Language (SGML), as evidenced by the Department of Defense's and several manufacturing companies' adoption of SGML and the rapid rise of commercial interest in the World Wide Web. SGML was adopted in 1988 as an international standard (ISO 8879) to represent and serve as an interchange for documents between computer systems. World Wide Web pages use an implementation of SGML called Hypertext Markup Language (HTML).

SGML/HTML use special character sequences within the document called markup tags to explicitly label the structure of the entire document. Each element is introduced by a *start-tag* and terminated with an *end-tag*. Markup tags can be nested within each other to identify all the characteristics of the element, as seen in Figure 1.

SGML uses an external Document Type Definition (DTD) that specifies the grammar of tagging rules for encoding and decoding a document's structure. The DTD contains a set of definitions giving the name of the element (generic identifier) and a content model that defines which subelements and character strings can occur in the content [2]. HTML is an implementation of SGML in that it is based on a particular DTD with a specific set of tags.

Most SGML-based languages, including HTML, identify the structural significance of the data,but do not provide any further

```
<body>
<title>Master Gardener Handbook</title>
<section1>
<sectitle>Entomology<sectitle>
<para> ..... </para>
</section1>
<section1>
<sectitle>Fruit Crops<sectitle>
<para> ...<emphasis>TEXT</emphasis>
</para>
</section1>
<section1>
<sectitle>Houseplants and Flowers<sectitle>
<para> ..... </para>
.
.
.
</body>
```

Figure 1: Sample of SGML tags

knowledge about the concepts in the documents. For example, the SGML tags used in Figure 1 identify the syntax of the document. There are no indicators identifying the *meaning* or *type* of information, which limits the categorizing ability. Candide can add numerous types to all of the instances. This allows Candide to infer the inheritance of type. For example, if data within the document had identifiers that indicated the meaning or type of data (i. e. *insect, parasite, predator*) then connections between objects could be done. A case in point would be using the taxonomical inheritance of an insect to select a method of treatment.

SEMANTIC DATA MODEL

SDMs not only store data, but also reveal relationships between data segments. A SDM is a graph or network defined as a structure of knowledge represented as a pattern of interconnected nodes (entities, attributes and events) and arcs (relationships). SDMs describe entities and their relationships within a domain and provide tools to build a conceptual model of a domain. It is from these relationships that more intrinsic information can be represented from documents by a SDM as opposed to the syntactic approach provided by SGML.

A semantic database management system (DBMS) also allows different views of the data, including derived concepts that are not explicitly stored in the data. These concepts can be found through the use of the relationship between entities in the SDM. Take the following statements for instance: "a ladybug is a beneficial insect," "a beneficial insect is a desirable insect" and "desirable insects should not be exterminated." Using generalization / specialization and association relationships, it can be determine that "a ladybug should not be exterminated." This concept is not directly stated in the original data set.

DOCUMENT DATABASE

The Institute of Food and Agricultural Sciences (IFAS) at the University of Florida has been developing an information retrieval system for delivery of extension information in Florida. At the heart of the project is an object-oriented database management system (OODBMS) called Candide [1]. Candide has been designed and implemented based on semantic data modeling principles. Design criteria include the ability to store a broad domain of media types and information in an consistent object format, expandability to handle new data types and capability of integrating large domains of agricultural knowledge. All the data, including text fragments, menus, tables, functioning equations and links to figures, illustrations, photos, fielded data, expert systems and simulations are stored as Candide objects. Interconnections among objects provide a high level of integration across board domains.

We have shown that SDM is at least as expressive as SGML by representing a tagged document as a Candide object. Essentially, each tag in SGML is modeled as an object in Candide. A portion of the Candide document data model is informally illustrated in Figure 2, with an example of a portion of a document stored in the database shown in Figure 3. Each document is separated into one or more objects, each of which has an internal structure consisting of a name, a list of parents and one or more attributes. Objects are arranged in a taxonomy of classes. The list of parents identifies the classes to which an object belongs. A single object represents the top level of the document including Title, Author and general publication information of

```
DocX
  ISA: Document
  Title: String
  Author: A1,A2...
  Subject: <subject of document >
  Body: Ordered Set( <BodyElem>
                     <BodyElem>,
                     Section1,...,Section2)
```

2a Top level of document.

```
Section(H1)
  ISA: Section
  PartOf: <Back references to higher level>
  Heading: String
  Subject: <subject of section>
  Body: OrderedSet(<BodyElem> ...
                   <BodyElem>,
                   Section1, . . . ,
Section2)
```

2b. A section of the document.

Figure 2: Model for document database

the document. The Body attribute of the object contains the body of the document and has values which are an "OrderedSet," meaning a set in which the elements occur in a particular order. This is used to reflect the order of the elements within the body of the document. Maintaining the object-oriented design, each section is modeled as an independent instance as shown in Figure 3b. More technical descriptions can be found in other papers [1].

Due to the object-oriented format design of Candide, new data elements can be added. Most of the common SGML/HTML elements have already been incorporated into the data structure and new ones can be easily added. Moreover, additional constructs have been added to handle new multimedia formats not covered by SGML. For example, SGML DTDs do not indicate how to process nontext objects. Candide stores the nontext media information as an object like the other data in the database. A sample of media types which have been represented using Candide objects include *real-time* equations (equations that can be evaluated after reading from database), fielded data and expert system rules.

DATABASE FUNCTION

SGML allows for easy transmission of documents between users and platforms, however, it does not provide a robust information retrieval system as would a semantic database. By storing documents in a SDM database, contextual information is more accessible. Tagged documents by themselves are not suitable for query

```
Master Gardener Handbook
  ISA: Extension Handbook,
       Gardening Reference
  Title: Master Gardener Handbook
  Editor: (Person (FI: K)(MI: C:)
                  (Last_Name: Ruppert))
  Subject: (Handbook: (ISA: Reference)
           (Subject (Gardening
                     (Gardener: Master:
                      (ISA: E xpert))))))
  Body: OrderedSet (MGH_Entomology,
                    MGH_Fruit Crops,
                    MGH_Houseplants and
                        Flowers
                    MGH_Landscape,
                    MGH_Nematology,
                    MGH_Plant Pathology,
                    MGH_Soils and
Fertilizers)
```

3a Top level of document.

```
MGH_Entomology
  Heading: Entomology
  Subject (Aspect Gardening))
  PartOf: Master Gardener Handbook
  Body:
     OrderedSet (MGH_1_Fruit Crops,
                 MGH_1_Household and
Structural.
                 MGH_ 1_Ornamentals,
                 MGH_1_Pests of Man and
                       Domestic Animals)
```

3b. First Level section.

Figure 3: Example of Candide object

applications since searching through the SGML tags is cumbersome and inefficient. The SDM database makes the tagged structure of the document explicit and provides a way to rapidly access elements of a document, or search and retrieve a large

collection of documents.

There are several benefits to storing documents in a DBMS as opposed to individual files. When a document's structure is translated and stored in a DBMS, it allows program-data independence. This provides a conceptual representation of data that hides storage details regarding how the data are stored.

Other advantages of storing documents in a DBMS include data security, integrity and quality control of the information contained in the database. These management topics are difficult to maintain with SGML since its information is stored in traditional ASCII files and can be manipulated from different sources.
Storing information in a DBMS makes it more difficult for unauthorized individuals to alter. SGML itself does not provide security for the information since unauthorized or false information can easily be exchanged for valid SGML files.

A DBMS provides better integrity checking to ensure that media links are maintained. When data is removed from the database, an integrity check can be performed to determine whether the information removed is linked to other documents. In SGML, all of the documents must be re-read to locate dependent links in order to maintain linking integrity or provide external ad hoc mechanisms to maintain.

Document management is easier when using a database rather than assemblage of ad hoc methods used with individual SGML ASCII files. A DBMS facilitates version and control of content by providing software tools to track and organize the maintenace of documents. Although a method can be arranged when managing SGML files, it is usually more labor intensive and requires an assortment of tools. Moreover, a similar problem to the security issue may arise since unauthorized changes to the unsecured SGML files could circumvent management procedures allowing erroneous material to enter the system.

SGML can be a useful tool in helping to build a digital library. Used to capture the structure of the document, SGML can serve as a stepping stone towards building a SDM database. Marked tags are analyzed and the associated data parsed into nodes representing parts of the document.

A natural outgrowth of a semantic database is content based searching, which relies on the defined or inferred meaning of the data. A semantic database uses association to infer information beyond what is directly stated in the document. SDMs can be seen as a way to capture many of the relationships which are not expressed in SGML/HTML systems as they support a variety of formalized links and relationships. Figure 4 highlights these relationships using a small network describing a ladybug. The links in the graph express generalization / specializ-ation relationships or "ISA" (beneficial insect ISA insect), part/whole (Abdomen is part of Insect), association (Ladybugs eat Aphids), and class/instance (Ladybug is an instance of Beneficial Insect).

Since concepts in SDMs are described by structured graphs expressing the relationships among symbols rather than connections between text files as in SGML, there exists the capability of manipulation of SDMs to produce a number of desirable functions. An example is a search, or query processing operation, expressed as a small semantic network or graph. A query graph is matched against the larger database graph to find connections. This gives a much more precise search capability than is possible with boolean keyword search over text files as used with SGML.

The ability to perform content based searching is a promising feature of SDMs. A thesaurus is being constructed for Candide to create a network of relationships that can be used to automatically infer meaning of information in extension documents.

SUMMARY

As people rush to construct HTML libraries, they are forfeiting the benefits of storing information in a database. SGML is good for browsing, marking structure and transport-

ation of information, but is not suited for searching and other data management functions. Few relationships are built into the existing SGML tagging schemes.

Data objects can be constructed to be as expressive as SGML for representing document structure. Computer applications can be linked to the SDM that access the information directly. A SDM extends knowledge representation by allowing content-based search exploiting domain semantics, parsing/text analysis used to convert text to objects, capturing the content and grammar within the database and performing database management on agricultural documents.

REFERENCE

[1] Beck, H. W., G. Sunit and Shamkant B. Navathe. "Classification as a query processing technique in the Candide semantic data model." *Proc. IEEE Fifth International Conference on Data Engineering*. Los Angeles, CA., 1989.

[2] Beck, Howard W., Dennis Watson. "Analysis of Agricultural Extension Documents." *AI Applications*, 6 (3), 1992. pp 17-28.

[3] Brown, Heather. "Standards for Structured Documents." *The Computer Journal*, 32 (6), 1989. pp. 505-514.

[4] Cronk, Randall D. "Unlocking Data's Content." *Byte*, Sept., 1993. pp. 111-120.

[5] Elmasri, Ramez, Shamkant B. Navathe. *Fundamentals of Database Systems.* Benjamin/Cummings Pub., 1989.

[6] Sowa, Fohn F. *Principles of Semantic Networks.* Morgan Kaufmann Pub., San Mateo, CA 1991.

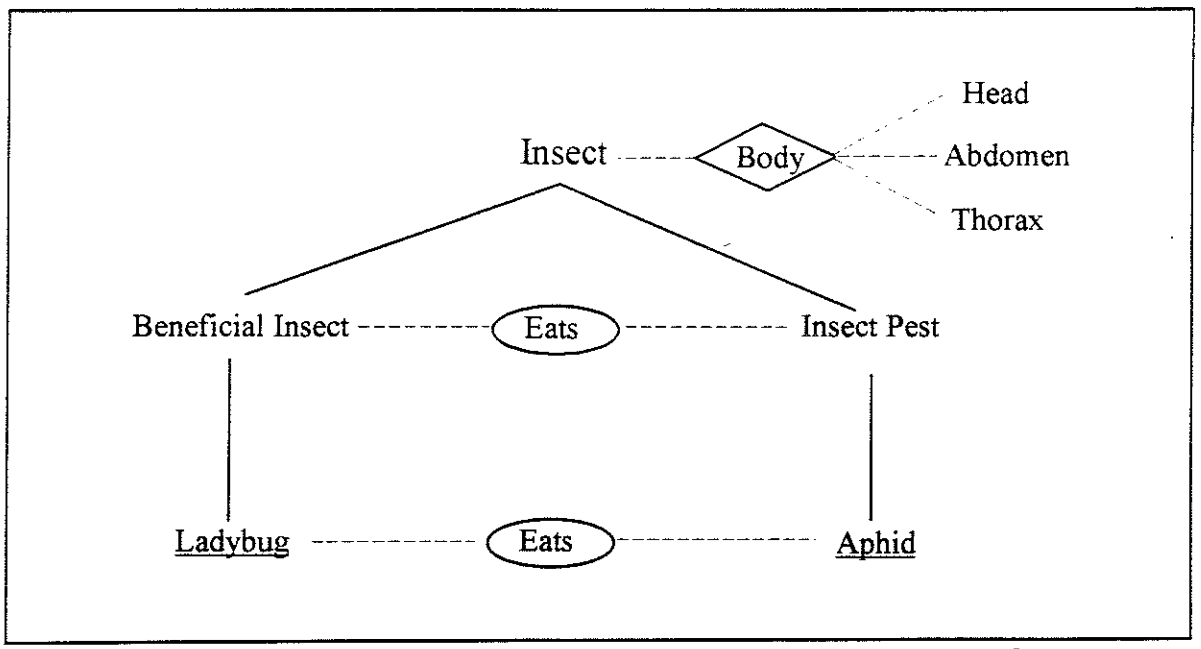


Figure 4: Semantic Data Model graph representation of insects. Solid lines are ISA relationships, diamonds are part/whole, circles are associations, instances are underlined

AN EXPERT SYSTEM FOR DIAGNOSIS AND TREATMENT OF BACTERIAL CLOGGING IN MICROIRRIGATION[1]

Fedro S. Zazueta, Professor
Agric. and Bio. Eng. Dept.
University of Florida
Gainesville, FL 32606, USA

Willem Huisman, Professor
Agric. Eng. and Physics Dept.
Wageningen Agric. University
6700HB Wageningen
The Netherlands

Allen G. Smajstrla, Professor
Agric. and Bio. Eng. Dept.
University of Florida
Gainesville, FL 32606, USA

Abstract

A knowledge-based tool for diagnosing and providing management recommendations for iron and sulfur clogging problems in microirrigation systems was developed, tested and distributed to agricultural irrigation system designers and managers. The knowledge base is the result of more than ten years experience by the authors in dealing with biological clogging problems in microirrigation systems in Florida and other parts of the world. The system has been used successfully since its first version in 1983 and has evolved continuously since its original version. The system diagnoses and recommends a treatment based on water quality, filtration system other equipment characteristics and management practices. Bacterial clogging problems have been decreased or totally eliminated in microirrigation systems where recommendations obtained from this tool have been used. The system was written using Prolog and the Txpert Prolog code generator.

Keywords: drip irrigation, irrigation management, irrigation water quality.

INTRODUCTION

Because of the small hydraulic passages and low velocities occurring in emitters, clogging prevention and control is an important issue in microirrigation system management. Microirrigation clogging can be traced to physical, chemical and biological factors (Bucks et al., 1977; Nakayama, 1986). Among these, biological clogging seems to be one of the most persistent and management intensive problems (Gilbert and Ford, 1986).

Bacterial activity and associated by-products produce slimy deposits on walls of containers and conduits. In addition, suspended particles tend to agglomerate with the slime, resulting in blocked water passages in the irrigation system. Instances of the effects of slimes on the hydraulics of pipes have been reported in the literature, particularly those related to decreases in hydraulic capacity of single pipe conduits (Characklis, 1973). Some of the early case studies (Minkus, 1954; Arnold, 1956) found that a large portion of the slime consisted of iron and manganese and that *Crenothrix* was present. Also, heavy growths of *Beggiatoa* were observed in waters containing hydrogen sulfide (Derby, 1947). Biological clogging problems in microirrigation systems were recognized as a serious problem and were reported by Sharp (1956), Ford and Tucker (1974a, 1974b), McElhoe and Hilton (1974), and Pelleg et al. (1974).

Four major forms of slime have been identified in irrigation systems (Abbott, 1985): 1) iron, 2) manganese, 3) sulphur, and 4) non-specific filamentous and non-filamentous slimes. Aerobic sulfur slime is formed by the oxidation of hydrogen sulfide by the filamentous bacteria *Thiothrix* and *Beggiatoa*. Iron related clogging problems can arise due to iron precipitation induced by bacteria such as the filamentous *Gallionella* and *Leptothrix*. In addition, bacterial clogging can occur due to an excessively large bacterial mass, as is the case with *Vitreoscilla*, *Enterobacter* and *Pseudomonas* (Ford, 1978).

Slime clogging problems may result in two management activities: 1) reclamation of clogged or partially clogged system components, and 2) maintenance activities that control the problem. Reclamation of pipes and irrigation systems with biological deposits requires injecting a chemical solution and flushing pipelines (Table 1).

Maintenance of pipes and irrigation systems with potential for a bacterial clogging problem requires injecting a biocide into the water. Several chemicals have been tested and used for clogging prevention of microirrigation systems. McElhoe and Hilton (1974) reported on the use of various oxidants, soil fumigants, metallic compounds, and other biocides with various degrees of success. It is widely recognized that an economical and practical technique for managing the iron and sulfur bacterial problem is to add chlorine to the irrigation water source or to inject it into the irrigation system. The purpose is twofold: 1) to precipitate iron from the irrigation water,

---

1 Florida Agricultural Experiment Station Journal Series No. R-02601.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/410 ©1996 FLAIRS

and 2) to kill the offending bacteria. Common forms of chlorine used are chlorine gas, or a chlorine compound such as calcium or sodium hypochlorite. Upon dilution the chlorine hydrolyses to hypochlorous acid (HOCl), which acts as a biocide. In addition, at high concentrations, chlorine acts as an oxidant.

Because of its biological nature, bacterial clogging problems are difficult to predict with accuracy. Bacterial rates of growth are dependent on reduction-oxidation potential, oxygen content of the water, pH, temperature energy sources in the water and other environmental factors. These may also affect the effectiveness of the biocide or oxidizing agent used. Because of this, most recommendations given for biological clogging control are broad and require trial and error in the field to obtain best results. Tools used by farmers to predict and manage biological clogging problems have included recommendations based on the initial and chlorine free residual concentrations in the irrigation water (Tyson and Harrison, 1985), effects of pH and time of contact (Goyal, 1990), heuristically based keys (Ford, 1979), software (Zazueta, 1983) and devices that indicate the potential for deposits before clogging occurs (Ford, 1978).

THE EXPERT-SYSTEM

A knowledge-based computer software tool was developed in this work. The objectives of the software are to:

1. Determine the potential for iron and sulfur bacteria clogging problems
2. Determine if an identified potential problem can be treated with the addition of free chlorine to the irrigation water.
3. If the problem can be treated with chlorine, outline the treatment, including:

    - Minimum level of free residual chlorine - required in the irrigation system.
    - Duration of contact.
    - Frequency of treatment.
    - Concentration of chlorine in the irrigation water at the point of injection.
    - Filtration requirements.

The treatment method was limited to chlorination because chlorine is readily available and commonly used for this purpose (Clark and Smajstrla, 1992).

Both water source and irrigation system factors affect biological clogging of irrigation systems. The user of this software must provide the following information about the irrigation water source: 1) iron concentration, 2) sulfide concentration, 3) pH, 4) type of source (surface or well), 5) presence of biological growths (algae), 6) presence of organic iron complexing agents, and 7) occurrence of suspended solids. In addition, the software user must provide the following irrigation system information: 1) type of filtration system, and 2) emitter orifice size.

Early Work

The most difficult step in developing a knowledge based tool is the process of knowledge acquisition. This consists of extracting, structuring and organizing the knowledge from the human expert in such a way that it can be implemented as computer software. This task was greatly simplified for the early version of this software because the domain expert was readily identified, and a basis for knowledge extraction had been presented by the domain expert in the form of a decision tree (Ford, 1979). The decision tree was originally programmed in Pascal to produce software that would meet the stated objectives. The software was then tested by three irrigation experts and the domain expert. A total of 16 different scenarios were tested, with the human experts agreeing in all cases with the expert system's diagnosis and recommendations. It was then published (Zazueta et al., 1983) and several thousand copies have been distributed through the Florida Cooperative Extension Service to extension agents, irrigation system designers and managers. Although the knowledge-base was developed based on experiences in Florida, it has been used as a teaching and diagnostic tool in many parts of the world, including Australia, Spain, Pakistan and several countries in Central and South America.

As the complexity of the software increased, maintaining the Pascal version of the software became increasingly difficult and time consuming At the time the first version of the software was written no useful expert system programming tools or languages were available for microcomputers. Thus, a decision was made to represent knowledge in the form of rules, and rewrite the software in a suitable language for expert system applications.

Knowledge Engineering

Knowledge was organized into sets of rules. The first set of rules was used to determine if the potential for iron or sulfur clogging problems exists for a given water source. When the potential for clogging exists, filtration is a necessity. Thus, a set of rules relating to the filtration system were developed. These rules include the effects of emitter type, filtration system type, location of chemical injection points, and other factors. Recommendations for modifications to the filtration system were included in this set of rules. Such modifications include changes of filtration system type , and location of chemical injection points.

Because environmental factors may compound biological clogging problems, a set of rules was developed to

accommodate some of these factors. These include conditions such as presence of organic iron complexing agents and visible biological growths in the irrigation water. These also include conditions that result in increased dissolved oxygen in the irrigation water, such as cracked well casings and valves on the intake side of the pump. When potential problems are identified, system modification recommendations and management precautions are provided to the user.

Depending on the combination of water quality, filtration system and compounding factors, a determination is made of the severity of the potential clogging problem. The severity of the problem one of 21 different levels. A chlorination treatment is recommended for each level.

Due to the author's involvement in extension programs related to biological clogging problems, actual field cases encountered over an eight year period provided additional experience that was also incorporated into the expert system, particularly in the rule sections pertaining to filtration recommendations and compounding factors. The software evolved through a series of versions. Changes and additions from version to version included changes in wording, rules and treatment recommendations.

Software Development

The language chosen for the application was Prolog. A skeleton of the Prolog code was generated using TXPERT (Zazueta, 1988). The generated code was then modified to extend some of the functions of the software. The current version contains 47 rules, 21 chlorination management recommendations and 32 miscellaneous recommendations. Figures 1, 2, and 3 show an example rule, a miscellaneous recommendation, and a chlorination recommendation, respectively. The rule in Fig. 1 is used after it has been determined that chlorination is needed because of the iron and sulfides concentration and pH of the water source. Also, within this range of values it has been found that systems that use pressure tanks which permit air-water contact tend to incubate bacteria in the tank because of the available oxygen in the pressure cavity. Figure 2 shows the recommended action with respect to the pressure tank. Figure 3 shows the recommended chlorination schedule.

Software Testing

To test the Prolog version of the software the set of scenarios originally used to test the first version were used. In addition a scenario testing each rule that had been added since the original version was devised by the authors. The software produced the correct diagnosis and treatment for each scenario. This testing was considered to be adequate since the underlying knowledge in the software had been applied to field problems in commercial production microirrigation systems since 1983 with successful results after the management recommendations were implemented.

CONCLUSIONS

A knowledge-based software tool is an effective aid for diagnosis and treatment of bacterial clogging in microirrigation systems. Different versions of this software have been used since 1983 to diagnose and provide management recommendations for iron and sulfur bacterial clogging problems. It is clear that the appropriate selection of the software development tools, as well as knowledge organization within the code used, greatly affect the ease of maintenance of the software.

REFERENCES

Abbott, J.S. (1985). Emitter Clogging-Causes and Prevention. ICID Bul. 34(2):11-20.

Arnold, G.E. (1963). Crenothrix chokes conduits. Eng. News Rec.

Bucks, D.A., Eire, L.J., & Nakayama F.S. (1974). Trickle irrigation management for grapes. 2nd Int. Drip Irrigation Congress Proc. pp. 503-507.

Bucks, D.A., Nakayama, F.S., & Gilbert, R.G. (1977). Clogging research on drip irrigation. Fourth Annu. Int. Drip Irrig. Assoc. Meeting Proc., Fresno, CA. pp. 25-31.

Characklis, W.G. (1973). Attached microbial growths-II, Frictional resistance due to microbial slimes. Water Research 7: 1249-1258.

Clark, G. and A.G. Smajstrla. (1992). Treating irrigation systems with chlorine. Florida Cooperative Extension Service Circular 1039,.4 p.

Cravens, B. (1966). Stabilized chloride dioxide for microorganism control. J. TAPPI 49, 53-55.

Derby, R.L. (1947). Control of slime growths in transmission lines. Journal of the Amer. Waterworks Assoc. 39,1107-1114.

Ford, H.W., & Tucker, D.P.H. (1974a). Water quality measurements for drip irrigation systems. Proc. Fla. State Hort. Soc. 87,58-60.

Ford, H.W., & Tucker, D.P.H. (1974b). Clogging of drip systems from metabolic products of iron and sulfur bacteria. 2nd Int. Drip Irrig. Congress Proc. pp. 215-220.

Ford, H.W. (1978). Bacterial clogging in low pressure irrigation systems. Technical Session, Irrigation Association Meeting. Cincinnati, Ohio. 10 p.

Ford, H.W. (1979). A key for determining the use of sodium hypochlorite (liquid chlorine) to inhibit iron and slime clogging of low pressure irrigation systems in Florida. Lake Alfred RR-CS79-3. Institute of Food and Agric. Sciences. Univ. of Florida. Florida Cooperative Extension Service. 5 p.

Gilbert, R.G. & Ford, W.H. (1986). Operation principles: Emitter clogging. Chapter 3.1, pp 142-163. In: Nakayama, F.S. & D.A. Bucks. (Eds.) Trickle irrigation for crop production. Elsevier. New York.

McElhoe, B.A. & Hilton, H.W. (1974). Chemical treatment of drip irrigation water. 2nd Int. Drip Irrigation Congress Proc.. Fresno, CA. pp. 215-220

Minkus, A.J. (1954). Determination of the hydraulic capacity of pipelines. J. New England Water Works Association 68,1-10.

Muller, W.S. & Listsky W. (1968). Effects of various chemical agents for the inhibition of Sphaerotilus natans in paper mill process water. Water Research 2, 289-296.

Nakayama, F.S., Bucks, D.A., & French, O.F. (1977). Reclaiming partially clogged emitters. Transactions of the ASAE 88,278-280.

Nakayama, F.S., (1986). Operational Principles: Water treatment. Chapter 3.2, pp. 164-187 In: Nakayama, F.S. & D.A. Bucks. (Eds.) Trickle irrigation for crop production. Elsevier. New York.

Sharp, R.B. (1956). The growth of mucus-forming bacteria in drip-feed irrigation lines. J. Ag. Eng. Research 1,83-88.

Pelleg, D., Lahav, N., & Goldberg, D., (1974). Formation of blockages in drip irrigation systems: Their prevention and removal. 2nd Int. Drip Irrigation Congress Proc.. pp. 203-208.

Piatek, A. (1969). Preventing fillamentous scale in well water. Wat. Wastes Eng. 4, 54-55.

Pollard A.L. & House, H.E. (1959). An unusual deposit on a hydraulic tunnel. J. Pwr. Div. Am. Soc. Civ. Eng. 85 PO6, 163-171.

Tyson, A.W., & Harrison, K.A., (1985). Chlorination of drip irrigation systems to prevent emitter clogging. Miscellaneous Publication 183. College of Agriculture. The Univ. of Georgia. 10 p.

Zazueta F.S., Smajstrla, A.G., Harrison, D.S., & Ford, H.W. (1983). Diagnosis & treatment of iron and slime clogging problems. Univ. of Florida. Florida Cooperative Extension Service. Circular 585. 2 p.

Zazueta, F.S. (1988). TXPERT: A Prolog code generator. Agric. Eng. Dept., University of Florida. (Software).

```
/* Rule 16b, contact with air in press tank */
    diagnosed(change_tank14c) :-
    ironconc(IronConcentration),
    pHvalue(PH),
    sulfides(SulfideConcentration),
    IronConcentration > 0.05,
    IronConcentration < 3.0,
    SulfideConcentration <= 0.05,
    PH < 6.5,
    check(welltype),
    check(air),
    recommendChangeOfPressureTank,
    recommendNaOClTreatment_c,!.
```

Figure 1: Example of a Prolog rule used in the software.

```
recommend_ChangeTank :-
  open_diagnostic_window,
  write("The pressure tank should be changed"),nl,
  write("to an airless type of tank. Otherwise, "),nl,
  write("the tank will act as an incubation"),nl,
  write("chamber for bacteria."),nl,
  write("When this is performed,"),nl,
  write("follow the next recommendation"),
```

Figure 2: Miscellaneous recommendation.

```
recommend_c :-
open_diagnostic_window,
  write("Use a minimum of 1.0 ppm (Mg/l) of free"),nl,
  write("residual chlorine measured at the last"),nl,
  write("emitter. To obtain this concentration at the"),nl,
  write("last emitter start with 3.0 ppm at the"),nl,
  write("injection point near the pump. Inject for a"),nl,
  write("minimum of 40 minutes for each cumulative 6"),nl,
  write("hours of irrigation time.    "),nl.
```

Figure 3: Chlorination management recommendation.

Table 1

Chemicals Reported in the Literature Used in the Reclamation of Pipes and Trickle Irrigation Systems with Bacterial Slimes

Chemical(s)	Concentration	System	Reference
Chlorine	9 mg/l - 12 mg/l	14 in diam. pipeline	Derby (1947)
Chlorine	50 mg/l	42 in. diam. pipeline	Minkus (1954)
Calcium Hydroxide	20 mg/l	11 ft. diam tunnel	Pollard (1959)
Chlorine-Ammonia	0.7 mg/l- 0,2 mg/l	36 in. diam. pipeline	Arnold (1963)
Arthium Dioxide, calcium hypochlorite, and Calgon mixture	58 l/m^3, 5.2 kg/m^3, and 47 kg/m^3 respectively	8 in. well	Piatek (1969)
2-Bromo-4-hydroxy-acetophene (Busan-90)	5 mg/l - 11 mg/l	Incubation chamber	Muller and Litsky (1968)
Bis-1, 4-bromoacetoxy-2-butene		Incubation chamber	Muller and Litsky (1968)
Sodium hypochlorite - Sulfuric Acid	100 mg/l - required for pH 2	Micro- irrigation system	Nakayama et al. (1977)
Chlorine	1000 mg/l	Micro- irrigation system	Abbott (1985)
Oxydants, soil fumigants, metallic compunds, oxides, and other biocides	Various		McElhoe and Hilton (1974)

AN EXPERT SYSTEM FOR MICROIRRIGATION CONTROL

J. N. Xin
Comp. Programmer/Analyst
Agricultural & Biological
Engineering Dept.
University of Florida
Gainesville, FL 32611

F. S. Zazueta
Professor
Agricultural & Biological
Engineering Dept.
University of Florida
Gainesville, FL 32611

T. A. Wheaton
Professor
Horticulture Science
CRC, Lake Alfred 700
Exp. Station Rd.
Lake Alfred, FL 33850

D. D. Dankel II
Assistant Professor
Dept. Computer and
Information Science
University of Florida
Gainesville, FL 32611

Abstract

An expert system (CIMS) was developed for citrus microirrigation control. Sensors were used to collect real-time field data as inputs to the system. The system was designed to operate continuously to respond to real-time data and execute necessary control actions. A knowledge base was constructed for citrus irrigation, fertigation, and cold protection management. A data uncertainty management approach was used to validate the sensor readings. Field and predictive tests were conducted. These demonstrated the expert system worked as an automated management tool for irrigation, fertigation, and cold protection. The system has the potential to improve microirrigation management, to achieve water and energy savings, and to prevent water pollution due to improper fertigation management.

INTRODUCTION

In Florida, microirrigation is the preferred method used for citrus irrigation. With microirrigation systems, water is applied at a low volume and a high frequency to increase water use efficiency. Irrigation may be applied at the rate equal to crop evapotranspiration requirements. Because irrigation events are high frequency and asynchronous, there is a need for real-time irrigation scheduling and control. Microirrigation systems are also used for chemical applications and cold protection purposes. Proper irrigation management is not only important to meet crop water requirements and to achieve water savings, it is also important to deliver crop nutrient and for cold protection. Expert knowledge is required to apply chemicals in such way that it meets crop nutrient requirements and to prevent potential water pollution. In addition, cold protection needs to be started at a critical time and last long enough to avoid cold damage.

Irrigation management involve complex daily decisions that are affected by water and nutrient requirement of crop, climate condition, rainfall probability, and economic factors. Sound irrigation management uses crop, weather, and soil-water content data in deciding how much water should be applied and when to apply it. Thus, soil moisture sensors and irrigation control devices are commonly used for automated irrigation systems. As personal computers become common in farm management, computer controlled irrigation systems have been developed to assist irrigation management (Cahoon et al., 1990; Zazueta et al., 1994). Computer controlled irrigation systems have proved to increase water savings than manually operated irrigation systems (Zazueta et al., 1984). In the past few years, a variety of expert systems have been developed for agricultural applications (Kline et al., 1987; Muttiah et al., 1988; McClendon et al., 1989; Srinivasan et al., 1991; Kumar et al., 1992). Most of these expert systems do not require real-time data as inputs. To realize expert irrigation control and management, field data and irrigation control devices are imperative. The expert system must be operated in a real-time domain to deal with dynamic data and time critical responses. The objective of this work was to develop an expert system for microirrigation control and management.

SYSTEM DESCRIPTION

The following hardware components were used in this work.

- An automated weather station to measure climate data (Campbell Scientific)
- Soil moisture sensors (tensiometers with pressure transducers) to measure soil water potential
- A personal computer with digital input/output control board
- Solenoid irrigation control valves used as actuators

Figure 1 shows the systems main module. The data acquisition module collects real-time field data (climate data and soil water potentials at the crop root zone). In

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/415 ©1996 FLAIRS

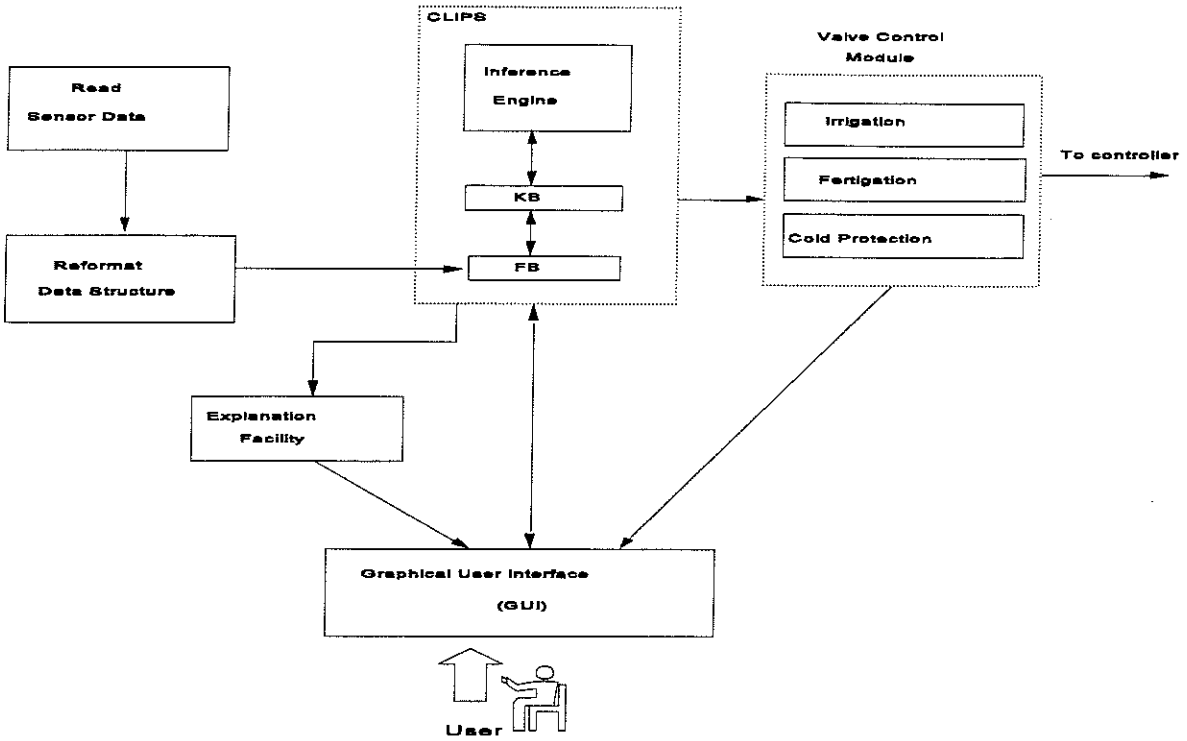


Figure 1. Block diagram of the expert system.

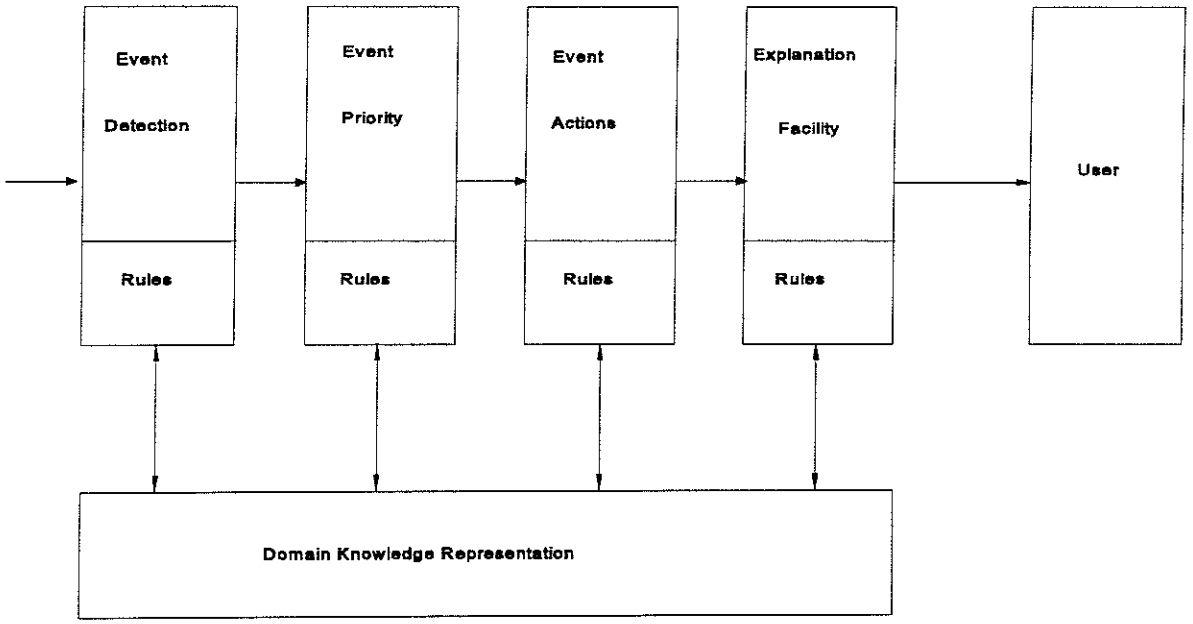


Figure 2. Knowledge base structure of the expert system

a real-time system, unreliable or missing data may prevent the system in carrying out a good decision or even jeopardize the decision process. To circumvent this a data validation procedure and a certainty factor (Shortliffer and Buchanan, 1975) approach were used to evaluate the sensor data. A forward-chaining expert system shell, CLIPS (NASA, 1991), was used as the inference engine. Expert rules on irrigation management were developed by interviewing the domain experts. Hardware control routines were developed to turn on or off the external devices. The control routines were integrated into the knowledge base so that the control action can take place when a proper rule is fired. The graphical user interface displays control actions and explains reasoning process.

Unlike common expert systems, CIMS deals with real-time data. Two basic technical issues were overcome in the system development: 1) how to deal with the real-time data inputs, and 2) how to reason about the behavior of the decision process. A special architecture for real-time data and temporal reasoning approach were used. Figure 2 shows the reasoning process steps 1) recognize input events and management of data uncertainty, 2) determine the priority of these events, 3) reasoning process to make management and control decisions, 4) take an action based upon the priority, and 5) explain the reasoning process. In order to operate the system continuously to respond to real-time data, CILPS source code was modified and recompiled to realize a continuous reasoning process.

KNOWLEDGE BASE DESCRIPTION

Domain experts were interviewed to acquired knowledge on citrus microirrigation management. Approximately 100 rules and associated certain factor were constructed. The knowledge base, in particular, addresses the issues of data uncertainty management, irrigation strategies for a given rainfall probability, fertigation, and cold protection management.

Irrigation management decisions are highly related to rainfall. In this work, forty years rainfall data were used to estimate rainfall probability based on a third order Markov chain (Feller, 1969). With the real-time weather data and the historical rainfall probability, an estimated rainfall probability was given to assist the irrigation decision-making process. Thus, full irrigation or deficit irrigation may be applied according to an estimated rainfall probability. Details of this approach were discussed by Xin (1995).

SUMMARY

CIMS is an expert system operated in the real-time domain. The system uses real-time data as inputs and is operated continuously to make decisions on microirrigation management. CIMS was evaluated by potential users and domain experts against their opinion. Positive comments were received from the reviewers. In addition, many input scenarios were created to test the reasoning process and data uncertainty management. Short term field tests were conducted in Orlando Conserv II site. The tests showed that the system worked as expected to turn on or off the system in respond to a variety of inputs. Further study is recommended to enhance the knowledge base and to integrate a simulation model to estimate crop water requirement.

References

Cahoon, J., J. Ferguson, D. Edwards, and P. Tacker. 1990. A microcomputer-based irrigation scheduler for the humid mid-south region. *Applied Engineering in Agriculture* 6(3):289-294.

Feller, W. 1969. An Introduction to Probability Theory and Its Applications. Vol. 1. 3rd ed. New York: John Wiley & Sons.

Kline, D. E., D. B. Bender, and B. A. McCarl. 1987. Farm-level machinery management using intelligent decision support systems. ASAE Paper No. 87-1046, St. Joseph, MI.

Kumar, D., C. D. Heatwole, B. B. Ross, T. A. Dillaha. 1992. A knowledge-based system for preliminary selection and economic evaluation of sprinkler irrigation systems. *Applied Engineering in Agriculture* 8(4):441-447.

McClendon, R. W. , W. D. Batchelor, and J. E. Hook. 1989. An expert simulation system for irrigation management. ASAE Paper No. 89-2460, St. Joseph, MI.

Muttiah, R. S., C. N. Thai, S. E. Prussia, R. L. Shewfelt, and J. L. Jordan. 1988. An expert system for lettuce handling at a retail store. *Transactions of the ASAE* 31(2):622-628.

NASA. 1991. CLIPS Reference Manual, Software Technology Branch, Lyndon B. Johnson Space Center.

Shortliffe, E. H. and B. G. Buchanan. 1975. A model of inexact reasoning in medicine. *Mathematical Biosciences* 23:351-379.

Srinivasan, R., B. A. Engel, and G. N. Paudyal. 1991. Expert system for irrigation management. *Agricultural Systems* 36:297-314.

Xin, J. N. 1995. A real-time expert system for citrus microirrigation management. Doctoral Dissertation, University of Florida, Gainesville, Florida.

Zazueta, F. S., S. Park-Brown, A. G. Smajstrla, and D. S. Harrison. 1984. Computer control of irrigation systems for nurseries. *Proc. Fla. State Hort. Soc.* 97:285-286.

Zazueta, F. S., J. N. Xin, T. A. Wheaton, J. L. Jackson. 1994. A facility to study the use of electronic system for irrigation in Florida citrus production. *Proc. of the 5th Int. Con. on Computers in Agriculture* pp. 670-675.

Functional diagnosis goes to the sea: applying FDef to the Heavy Fuel Oil Transfer System of a ship

Luca Chittaro, Roberto Fabbri
Dipartimento di Matematica ed Informatica,
Università di Udine, Via delle Scienze, 206
33100 Udine - ITALY
chittaro@dimi.uniud.it

Joaquín López Cortés
Department of Energy Systems and Marine Engineering,
Technical University of Hamburg-Harburg
Denickerstrasse 15
21073 Hamburg - GERMANY

Abstract

In this paper, we apply functional representation and reasoning techniques to a real case in the domain of marine engineering. More specifically, we consider the problem of diagnosing multiple faults in the heavy fuel oil transfer system (HFOTS) of a modern container ship. The paper presents the considered application, provides a short overview of the FDef approach [2], and extends it in order to diagnose the HFOTS, also including representation and diagnosis examples.

1 INTRODUCTION

Reasoning about function for diagnostic purposes has been recently investigated by several research groups [1, 2, 4, 5, 6, 7]. As a result, the exploitation of functional knowledge is gaining a growing attention in model-based diagnosis. Nevertheless, a lot of work has still to be done on the functional diagnosis of real complex systems.

In this paper, we adopt one of the flow-based approaches [3, 6, 7] to represent function and we apply it to a real case in the domain of marine engineering. More specifically, we consider the problem of diagnosing multiple faults in the heavy fuel oil transfer system (HFOTS) of a modern container ship. The HFOTS is a representative example of a wide class of marine technical systems [8] characterized by their (i) high structural complexity in terms of number of components and possible interactions, and (ii) exploitation of a few different types of basic components that appear frequently in various parts of the system.

The functional representation we adopt is the one proposed by the Multimodeling approach [3], and the associated diagnostic engine is FDef (Functional Diagnosis with efforts and flows [2]). The initial experimentation pointed out the need of extending the set of FDef diagnostic axioms in order to diagnose a larger class of faults and better suit the needs of hydraulic systems. The extensions presented in this paper are general, and are easily re-usable to diagnose other hydraulic systems. Section 2 presents the considered application and the relevant classes of faults to be identified, motivating why a functional approach is well suited to the domain. In Section 3, a brief overview of the basic FDef approach is presented, then the extension made is introduced and illustrated in detail, also providing some representation and diagnosis examples. Section 4 summarizes the achieved results and introduces future work.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/419 ©1996 FLAIRS

2 DIAGNOSIS OF THE HFOTS

Heavy fuel oil (HFO) is a low cost fuel used by main and auxiliary marine diesel engines. The low quality of HFO (high viscosity, low chemical stability, presence of undesired particles and salted water) makes it necessary to handle it properly. Three different subsystem are typically devoted to the handling of HFO on a ship: a transfer, a cleaning, and a supply system. In this paper, we concentrate on the transfer system (HFOTS).

The HFOTS (Fig.1) is devoted to the storage and transfer of HFO among tanks. The main components of the HFOTS are pumps, valves, tanks, and pipes (for clarity purposes, Fig.1 shows the pipelines, without detailing each single pipe). Four tanks are devoted to the storage of HFO during refuelling and two tanks are devoted to settling down possible impurities such as solid particles and water. The HFOTS can work in several different modes, in order to achieve three different types of purposes: (i) *refuelling*, by pumping HFO from bunker boat (through the external connections on backboard or starboard side) to one of the HFO storage tanks (BST, CST, OST, SST) or to the four tanks simultaneously, exploiting the pump on the bunker boat, (ii) *moving* fuel between two given tanks, exploiting the on-board pumps (Usgr006 or Usgr007), and (iii) *emptying* one ore more tanks exploiting the pump on the bunker boat or the on-board pumps.

The slow processes that take place in the HFOTS as well as economical considerations have led designers to reduce the number of sensors in this type of systems to a minimum. In particular, available observations are:

- Position of hydraulically driven valves (open/closed);
- Operating signals of pump driving electrical motors (powered/not powered);
- Pressure drop over transfer pumps;
- Levels of tanks (from 0% to 100%);
- Pressure at pipe C10 (between OST and valve V3).

The most likely classes of faults in the HFOTS are: (i) obstruction of components (e.g., HFO can react and produce more stable subproducts which deposit as solids, or it can become solid under a certain temperature threshold), (ii) leaks (in pumps, pipes, valves), and (iii) stuck

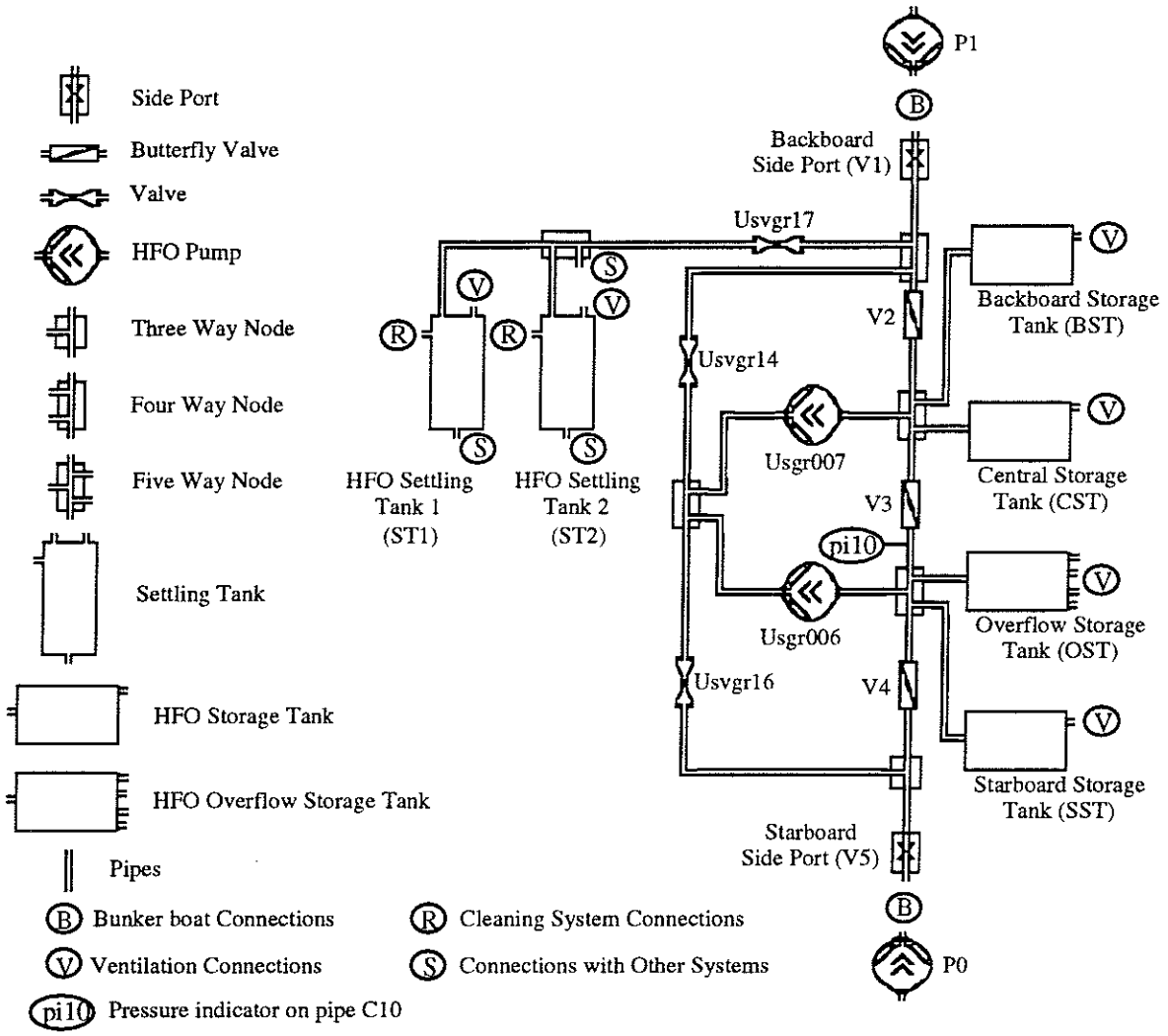


Fig. 1. Schematic of the HFOTS.

components (valves stuck at close or open, pumps that do not rotate).

A flow-based approach [3,6,7] to functional representation is well suited in the domain of marine technical systems. Such systems are indeed naturally represented as networks of abstract operators that act on the flow of substances. In particular, we adopt the functional representation proposed by the Multimodeling approach [3], because it also explicitly support the notion of effort (pressure, in the hydraulic domain) that is crucial in the considered application, and provides some criteria to abstract physical equations into functional roles. Moreover, interaction with the domain expert has shown that the explanations he provides very often refer to a limited number of abstract actions on substances (such as transporting, storing, transforming,...), and thus adopting a common functional modeling language did not present acceptance problems. From a reasoning point of view, the FDef [2] diagnostic engine is able to handle and generate not just single faults, but also multiple faults, as required by the considered application. With respect to the classes of considered faults, functional reasoning offers the potential of generating candidates without resorting to more complex and less efficient behavioral models, typically used in traditional model-based approaches.

3 FUNCTIONAL MODELING AND DIAGNOSIS OF THE HFOTS

This section first provides a brief overview of FDef, and describes how the HFOTS has been modeled. Then, the need of extending FDef is motivated by using a small example in the hydraulic domain, and the proposed extension is analyzed. Finally, a full HFOTS diagnosis example is illustrated.

3.1 The FDef Approach: an Overview

FDef [2] performs diagnostic reasoning, exploiting two different functional models: (i) a functional role model [3], made of conduit, generator, and barrier roles, and (ii) a process model [3], automatically derived from the functional role model, by recognizing patterns (called *cofunctions*) of functional roles that support a physical process (e.g. a circuit of conduits and a generator supports a transport process). In this way, the physical system is represented by

a network of functional roles that act on substances flowing through them, and a number of potential physical processes that can occur.

The observations on functional roles handled by FDef concern generalized flows and efforts and are of binary nature: a functional role is uncrossed (crossed) if the flow associated to it is (is not) zero, a functional role is unpushed (pushed) if the effort associated to it is (is not) zero. In the hydraulic domain, generalized flow and effort become hydraulic flow and pressure; a component is thus crossed (uncrossed) if it is (is not) traversed by flow, and it is unpushed (pushed) if the pressure drop across it is (is not) zero.

The diagnostic strategy of FDef is described in detail in [2]. It is mainly based on the identification of the so called *enabling sets* (an enabling set is a set of functional roles which are all allowing the passage of flow or effort), and *disabling sets* (a disabling set is a set of functional roles where there is at least one impediment to the passage of flow or effort). These sets are then used both for exoneration purposes (identify components which are performing their function), and to generate conflicts (i.e. sets of components, each one containing at least a faulty component). When the set of conflicts is available, a simple candidate generation algorithm [9] produces the minimal diagnoses.

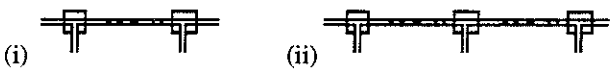


Fig.2. Decomposition of (i) a four-way node, and (ii) a five-way node.

3.2 Functional Representation of the HFOTS

In the hydraulic functional role model of the HFOTS, pipes are represented as conduits and denoted with C0,...,C42; valves (V1,...,V5) are conduits or barriers, according to their operating mode; pumps are generators or barriers, according to their operating mode (Usgr006 and Usgr007 are the on-board pumps, while P0 and P1 are pumps on the bunker boat); ST1 and ST2 are the two settling tanks, and SST, BST, CST and OST are the storage tanks. All the tanks could be in a closed or open state (each tank is provided with a built-in valve to switch between the two states) and the associated functional role is conduit (when open) or barrier (when closed). The four-way and five-way nodes in Fig.1 have been decomposed in their elementary components (pipes and three-way nodes, as shown in Fig.2). Since three-way nodes do not have faulty behaviors, they have not been represented as functional roles in the model, but just as connections between two different flows (graphically depicted as black filled circles). Fig.3 illustrates the hydraulic functional role model of the HFOTS in a mode devoted to the simultaneous refuelling of the four storage tanks. In the illustrated operating mode, the HFO is expected to flow from the pump P0 to the four storage tanks through C0, V5, C1, C3, V4, C4, C6, C8, C10, V3, C11, C13, C15.

3.3 The Need of Extending FDef

A crucial part of FDef diagnostic knowledge is represented by the axioms used for the generation of enabling and disabling sets. Although general and applicable to different physical domains, the set of axioms proposed in [2] cannot give satisfying results in the diagnosis of the HFOTS, because it does not consider the class of malfunctions where substance flows out from the intended structure of the system (e.g. shorts to ground in electrical systems, and leaks in hydraulic systems). Handling this class of faults requires to consider the direction of the flows and modify the axioms for the generation of enabling and disabling sets. For example, consider the simple circuit cofunction in Fig. 4, containing a generator (g1) and three conduits (c1, c2, c3), and suppose to observe c2 to be unpushed. In this case, FDef generates the disabling set [g1, c1, c3], i.e. at least one of these three roles is an impediment to the passage of flow and effort. This is correct if we exclude the possibility of leaks. On the contrary, if we consider leaks possible (for example, g1 could be a pump and c1, c2, and c3 could be pipes), the produced disabling set must be different and depend on the direction of the flow. More precisely, if the direction is clockwise, then the disabling set has to be [g1, c1], otherwise [g1, c3].

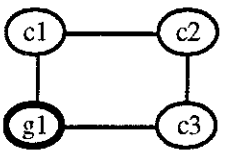


Fig.4. A simple circuit.

3.4 The Proposed Extension

Table 1 contains the new axioms we have introduced to generate enabling and disabling sets for the diagnosis of the HFOTS with FDef. These axioms apply to cofunctions where roles are ordered according to the direction of flow. They are not specific to the HFOTS and can be re-used for diagnosing other hydraulic systems.

Axioms E1 and E2 allow to derive enabling sets supporting crossed observations, dealing with the case where the observed crossed role in the cofunction is after or before a generator, respectively. In the first case (E1), the generated enabling set explains the observation hypothesizing that all the roles from the beginning of the cofunction to the observed role are allowing the passage of flow, and nothing is concluded about roles that are after the observed one in the cofunction (e.g. one of them could be leaking). In the second case (E2), the generated enabling set contains the roles in the cofunction from the beginning of it to the first generator after the observed role, and nothing is concluded about roles that are after that generator in the cofunction (e.g. one of them could be leaking).

Axioms E3 and E4 allow to derive enabling sets from observed pushed roles, dealing with the case where the observed pushed role in the cofunction is after or before a generator, respectively. In the first case (E3), the generated enabling set comprises the roles from the generator before the role (included) to the role itself (excluded), and nothing is concluded about the other roles in the cofunction. In the second case (E4), the generated enabling set contains the roles in the cofunction from the observed one (excluded) to the generator (included). Nothing is concluded about roles before the observed one, and the ones after the generator. Notice that axioms E3 and E4 can generate two enabling sets for a single pushed observation in a given cofunction, if there are generators both before and after the observation in the cofunction.

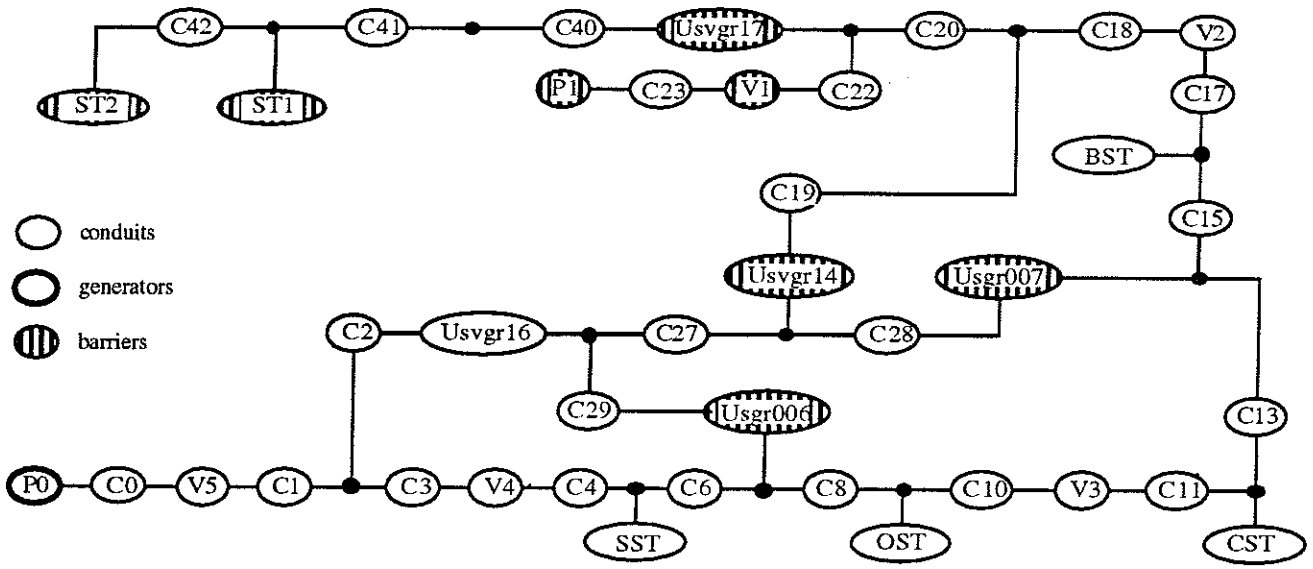


Fig. 3. Functional role model of the HFOTS during the refuelling of all the four storage tanks.

Axiom E5 deals with observed pushed generators: the generated enabling set contains only the pushed generator, covering the case where the observation is supported simply by the normality of that role.

Axioms D1 and D2 allow to derive disabling sets supporting uncrossed observations, dealing with the case where the observed uncrossed role in the cofunction is after of before a generator, respectively. In the first case (D1), the generated disabling set contains the roles from the beginning of the cofunction to the observed role (that is, at least one of this roles is an impediment to the passage of flow). In the second case (D2), the generated disabling set contains the roles from the beginning of the cofunction to the first generator after the observed role.

Axioms D3 and D4 allow to derive disabling sets supporting observed unpushed roles, dealing with the case where the observed unpushed role in the cofunction is after or before a generator, respectively. In the first case (D3), the generated disabling set comprises the roles from the generator (included) to the observed role (excluded). In the second case (D4), the generated disabling set contains the roles in the cofunction from the observed one (excluded) to the generator (included). Axiom D5 deals with observed unpushed generators: the generated disabling set contains only the unpushed generator, dealing with the case where the observation is supported by the abnormality of that role.

3.5 Reasoning Example

The observations (see Section 2) available on the HFOTS correspond to the following ones in the functional model:

- Expected functional roles associated to valves;
- Expected functional roles associated to pumps;
- Effort measurement (pushed or unpushed) for pumps (P0, P1, Usgr006, Usgr007);
- Flow measurement for tanks (ST1, ST2, OST, CST, SST, BST): the value is uncrossed if the level of the tank is steady, crossed otherwise;
- Effort measurement (pushed or unpushed) for C10.

Suppose that the HFOTS is in the operating mode of Fig.3 and the following values are obtained for the available flow and effort observations: P0 is pushed, P1 is unpushed, Usgr006 is pushed, Usgr007 is pushed, ST1 is uncrossed, ST2 is uncrossed, OST is crossed, CST is crossed, SST is uncrossed, BST is crossed, C10 is pushed.

The following are some of the enabling and disabling sets generated by FDef using the new axioms. The set {P0, C0, V5, C1, C3, V4, C4, C6, C8, C10, V3, C11, C13, C15, BST} is an enabling set generated by axiom E1 from observation "BST is crossed", that is a possible explanation of the observation is that all the functional roles in the set are allowing the passage of flow. Analogously, {P0, C0, V5, C1, C3, V4, C4, C6, C8} is an enabling set generated by axiom E3 to explain "C10 is pushed", while an alternative explanation is given, for example, by the enabling set {P0, C0, V5, C1, C2, Usvgr16, C27, C28, Usgr007, C13, C11, V3}, generated by axiom E3 too.

The set {P0, C0, V5, C1, C3, V4, C4, SST} is a disabling set generated by axiom D1 from observation "SST is uncrossed", that is at least one functional role in this set must be an impediment to the passage of the intended flow. Another example of disabling set generated by axiom D1 from "SST is uncrossed" is {P0, C0, V5, C1, C2, Usvgr16, C29, Usgr006, C6, SST}, with an analogous meaning.

The following is the set of minimal diagnoses generated by the FDef diagnostic engine using the enabling and disabling sets produced by the new axioms:

{ {SST}, {Usgr006,C6,V4}, {Usgr006,C6,C4},
{Usgr006,C6,C3}, {Usvgr14,C8,V4}, {Usvgr14,C6,V4},
{Usvgr14,C8,C4}, {Usvgr14,C6,C4}, {Usvgr14,C8,C3},
{Usvgr14,C6,C3}, {Usgr007,C8,V4}, {Usgr007,C6,V4},
{Usgr007,C8,C4}, {Usgr007,C6,C4}, {Usgr007,C8,C3},
{Usgr007,C6,C3} }

Axiom E1: If a role X in cofunction Cof has been observed to be crossed, and there is a generator G before X (or X is G) in Cof, and G has not been observed to be unpushed, then the roles of Cof from the first (included) to X (included) are an enabling set.	Axiom D1: If a role X in cofunction Cof has been observed to be uncrossed, and there is a generator G before X (or X is G) in Cof, and G has not been observed to be unpushed, then the roles of Cof from the first (included) to X (included) are a disabling set.
Axiom E2: If a role X in cofunction Cof has been observed to be crossed, and there is a generator G after X in Cof with no generators in between, and G has not been observed to be unpushed, and there is no generator before X in Cof, then the roles of Cof from the first (included) to G (included) are an enabling set.	Axiom D2: If a role X in cofunction Cof has been observed to be uncrossed, and there is a generator G after X in Cof with no generators in between, and G has not been observed to be unpushed, and there is no generator before X in Cof, then the roles of Cof from the first (included) to G (included) are a disabling set.
Axiom E3: If a role X in cofunction Cof has been observed to be pushed, and there is a generator G before X in Cof with no generators in between, and G has not been observed to be unpushed, then the roles of Cof from G (included) to X (excluded) are an enabling set.	Axiom D3: If a role X in cofunction Cof has been observed to be unpushed, and there is a generator G before X in Cof with no generators in between, and G has not been observed to be unpushed, then the roles of Cof from G (included) to X (excluded) are a disabling set.
Axiom E4: If a role X in cofunction Cof has been observed to be pushed, and there is a generator G after X in Cof with no generators in between, and G has not been observed to be unpushed, then the roles of Cof from X (excluded) to G (included) are an enabling set.	Axiom D4: If a role X in cofunction Cof has been observed to be unpushed, and there is a generator G after X in Cof with no generators in between, and G has not been observed to be unpushed, then the roles of Cof from X (excluded) to G (included) are a disabling set.
Axiom E5: If a generator G in cofunction Cof has been observed to be pushed, then {G} is an enabling set.	Axiom D5: If a generator G in cofunction Cof has been observed to be unpushed, then {G} is a disabling set.

Table 1. Axiomatic definition of enabling and disabling sets supporting observations about role X.

The first candidate is the most probable one: a faulty SST explains why pressure normally reaches C10, Usgr007 and Usgr006, and flow reaches OST, CST and BST, but not SST.

The other candidates are less likely triple faults. Consider, for example, the second candidate: {Usgr006, C6, V4}. In this case, V4 is faulty, and flow can thus not reach SST through it, a faulty Usgr006 allows the flow to reach all the other tanks (OST, CST, and BST), while the flow does not reach SST through Usgr006 because C6 is faulty. The explanation of the other triple faults is similar to this one.

The considered triple fault is also an example of a fault masking case, where the effect of the faulty V4 on the OST, CST, and BST tanks is masked by a simultaneous fault on Usgr006 which allows the flow to reach the three tanks.

FDef includes also an entropy-based [9] mechanism for test prescription, that can be used to suggest the most informative additional measurement to take in order to reduce the set of diagnostic candidates. In this case, measuring the pressure associated to SST is the suggested best measurement.

4 CONCLUSIONS

We have presented the first results of an on-going project aimed at applying the FDef approach to marine technical systems. The extension of FDef presented here has already been evaluated on a large set of examples concerning the HFOTS, in order to highlight directions for further investigation. In summary, the conclusions of the evaluation are that the approach is good at isolating faults when they result in a loss of functionality. Since some faults in the considered domain are preceded by a slow degradation in performance before turning into a loss of functionality, one of the subjects we are considering is the introduction and exploitation of "too low"/"too high" effort and flow values, besides the presence/absence values currently available. We are currently working at further scaling up the modeled marine technical system, by introducing other HFO subsystems belonging to non-hydraulic physical domains (such as thermal and electrical) and linking them to the HFOTS through influence links [3].

References

[1] Chandrasekaran B. 1994. Functional representation and causal processes, in Marshall Yovits (Ed.), *Advances in Computers*, vol. 38, Academic Press, 73-143.

[2] Chittaro L. 1995. Functional Diagnosis and Prescription of Measurements Using Effort and Flow Variables. *IEE Control Theory and Applications*, vol. 142, no. 5, 420-432.

[3] Chittaro L., Guida G., Tasso C., Toppano E. 1993. Functional and Teleological Knowledge in the Multimodeling Approach for Reasoning About Physical Systems: A Case Study in Diagnosis. *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no.6, 1718-1751.

[4] Hawkins R., Sticklen J., McDowell J.K., Hill T., Boyer R. 1994. Function-based modeling and troubleshooting. *Applied Artificial Intelligence*, vol. 8, no. 2, 285-302.

[5] Hunt J., Pugh D., Price C. 1995. Failure mode effects analysis: a practical application of functional modelling. *Applied Artificial Intelligence*, vol. 9, no. 1, 33-44.

[6] Kumar A. N., Upadhyaya S.J. 1995. Function based discrimination during model-based diagnosis. *Applied Artificial Intelligence*, vol. 9, no. 1, 65-80.

[7] Lind M. 1994. Modeling goals and functions of complex industrial plants. *Applied Artificial Intelligence*, vol. 8, no. 2, 259-284.

[8] Lòpez Cortés J., Michaelsen A. 1994. A Methodological Framework for Modelling Complex Technical Marine Systems for Diagnostic Purposes. *Proc. 2nd IEE Intelligent Systems Engineering Conference*, IEE Press, 298-303.

[9] de Kleer J., Williams B.C. 1987. Diagnosing Multiple Faults. *Artificial Intelligence* 32, 97-130.

PROVIDING AN AGENT THE ABILITY TO EXTRACT ITS OWN FUNCTIONAL REPRESENTATION

Luca Bogoni†
GRASP Laboratory, University of Pennsylvania
bogoni@grip.cis.upenn.edu

Abstract

The recovery and identification of features having functional significance in many developed systems are often provided a priori and these systems operate in simulated environments. This paper introduces a methodology for constructing functional representations of objects based on features identified and extracted by robotic agents. The shape of objects and the force monitored during interactions are integrated by means of Force-Shape maps. These maps capture an interaction in terms of functional features. The benefit of this approach is twofold. First, it allows a planner or a reasoning system to express functional tasks in terms of performatory and observational requirements and, at the same time, not to be constrained to the representation employed by the specific robotic system.

1 INTRODUCTION

Most of the research on functionality in the area of computer vision and artificial intelligence defines functionality by the occurrence of particular properties and relations fulfilling possibly more than one purpose [10, 5, 8, 13, 11, 7, 6, 17]. It is not clear, however, how these properties are initially acquired. How does a robotic agent actually extract from its sensors, features that allow it to identify an object as having a particular functionality? The representation of functionality is too often described independently of the agent and its perceptual capabilities. The representation should not only capture relations of parts of an object to the whole [15] and include the dynamics [16], and kinematics [4, 14, 12] of the use, but should also consider the interactive capabilities of the agent [1, 9] whose task it is to recognize the function of an object or to use the object in a functional manner. In many cases, the ability of the individual agent is assumed and this assumption sidesteps the problem of acquisition of the properties.

This paper introduces a methodology for the recovery and the construction of a representation for functionality of objects in the context of manipulation by robotic agents. This approach allows an agent to extract a representation based on its own perceptual and interactive capabilities. The system presented as part of this research is composed of three agents cooperating in investigating the functional properties of various tools. The observers fulfill a dual role: provide feedback control for the interactions and extract information for the representation of functionality. By analyzing the result of the interactions, it is possible to identify the extracted features of the object and of the interactions that contribute to the success of the operation. These features, integrated using Force-Shape maps, characterize the functionality of the objects while the experiments address the ability of a robotic system to extract functional features for piercing and chopping. The resulting representation (Force-Shape Maps) is expressed in terms of the agents' ability to observe the shape of the tool and the forces exerted over time.

While the extracted representation currently provides the ability to express only simple manipulatory functionalities, it promises to be a vehicle for agents to extract and represent objects' functionalities and, at the same time, provide means of communication between agents having different internal representations of objects having the same functionality.

Section 2 introduces the experimental setup and the experiments performed. Section 3 presents samples of the results. The identification of the functional features is discussed in section 4 and the extracted representation is presented in section 5. Discussions and conclusions are presented in section 6.

2 EXPERIMENTAL SETUP

The piercing and chopping tasks are described in terms of actions performed by the agent holding a tool as well as the monitoring and feedback provided by

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/424 ©1996 FLAIRS

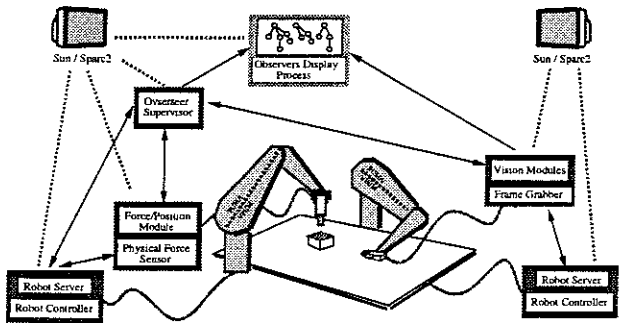


Figure 1: Experimental setup: two Puma 560 manipulators are used. One performs the interaction and monitors the force feedback while the other actively positions a camera.

the observers. The formalism [2] allows us to express tasks as finite automata in which the states represent actions and transitions are associated with observed events.

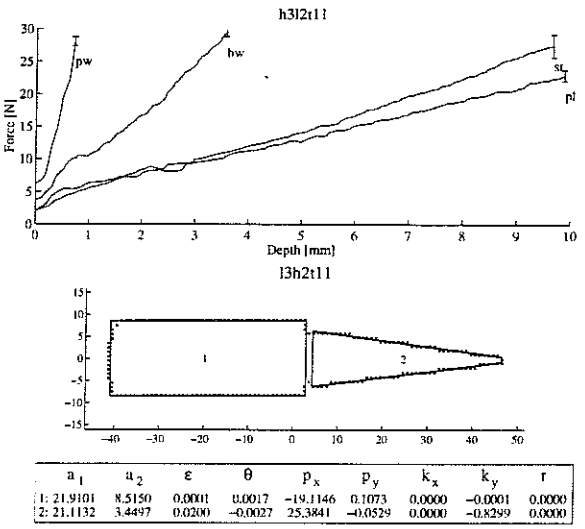
Both operations are described by the sequence of actions: *approach, contact, penetrate, extract* and *depart* and the functionalities are differentiated by the result of the operation. The result of a successful piercing operation is that of puncturing the target object to a given depth while chopping penetrates the object to a given depth possibly severing a portion of the material.

The architecture of the system is illustrated in Figure 1. The *overseer* and *supervisor* processes fulfill the dual role of controlling the feedback from the observers (overseer) and providing the appropriate commands to the robot carrying out the task (supervisor). The *vision observer* communicates with the overseer, using high-level information, commands and discrete events but also interacts locally with several processes. The *force observer* monitors changes in the force and provides continuous feedback to the robot moving the tool. The *position observer* gathers the information of the position of the end-effector and feeds it back to the overseer in the form of high-level events. The *robot processes* controls the motion of the tools and provides feedback for the force and position observer.

3 PIERCING AND CHOPPING

To identify functional features both in piercing and chopping, several materials and tools were employed. Each interaction was repeated several times to account for the variability of the material density and the noise in the force signal, using first order statistics. Over five hundred physical experiments were performed.

The results of the experiments for each tool are summarized in the form of extracted shape descrip-



	a_1	a_2	ε	θ	p_x	p_y	k_x	k_y	r
1:	21.9101	8.5150	0.0001	0.0017	-19.1146	0.1073	0.0000	-0.0001	0.0000
2:	21.1132	3.4497	0.0200	-0.0027	25.3841	-0.0529	0.0000	-0.8299	0.0000

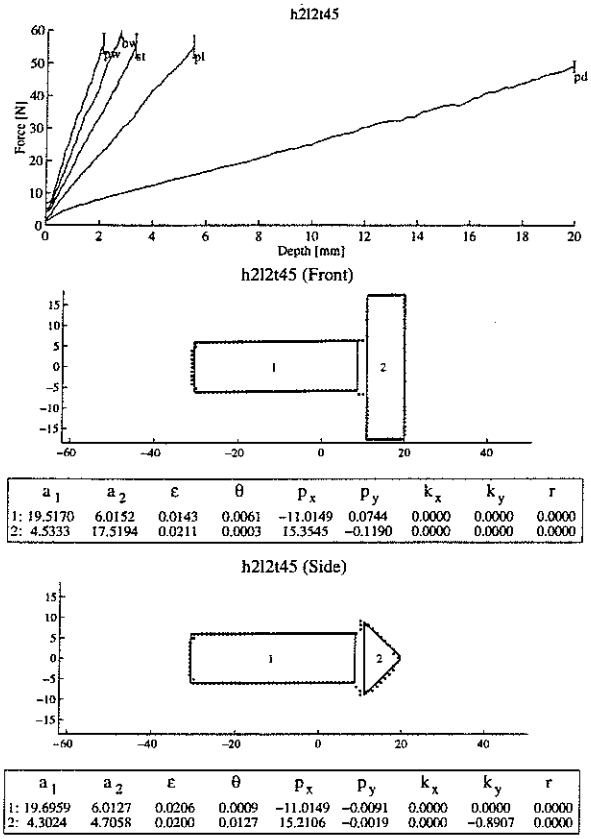
Figure 2: Graphs of the means for the force profiles and decomposition of the silhouette of the tool into superellipses.

tors and force profiles. Figures 2 and 3 show examples for tools employed in piercing and chopping.

To characterize the differences among the silhouettes of the various tools, the contour is segmented into polygonal subcomponents. These subcomponents are then expressed using superellipses. This choice of parametric primitives allows the individual subcomponents the ability to adapt to the dynamics of the interaction which may include relative displacement as well as deformations. The segmentation provides a means for comparing the subcomponents of the tools in an effective manner and, later, to monitor possible deformations or relative displacements of the components [1]

The force profiles provide a trace for the dynamic behavior of the interaction. This representation allows the penetration into a target object to be related to the material composing it. In addition, these profiles disambiguate observations which could lead to misinterpretation if they were entrusted solely to the vision system. In the case of piercing, this corroboration is needed in the case when, in an attempt to pierce, the tool is accidentally inserted into an existing hole of the target object. In the case of chopping, the position of the contact place could lead to different types of force profiles and hence it is important that the vision system be able to recognize the relative position of the target and tool. The complementary corroboration by the vision sensor is also necessary. In fact, if the han-

[1] A detailed description of the contour segmentation process and of the physics-based system developed to extract the decomposition of the tools into primitives was presented in [3].



	a_1	a_2	ε	θ	p_x	p_y	k_x	k_y	r
1:	19.5170	6.0152	0.0143	0.0061	-11.0149	0.0744	0.0000	0.0000	0.0000
2:	4.5333	17.5194	0.0211	0.0003	15.3545	-0.1190	0.0000	0.0000	0.0000

	a_1	a_2	ε	θ	p_x	p_y	k_x	k_y	r
1:	19.6959	6.0127	0.0206	0.0009	-11.0149	-0.0091	0.0000	0.0000	0.0000
2:	4.3024	4.7058	0.0200	0.0127	15.2106	-0.0019	0.0000	-0.8907	0.0000

Figure 3: Orthogonal views of a tool employed in chopping. The labels on the force profiles refer to interactions of the tool with target objects composed of pine wood (pw), balsa wood (bw), styrofoam (st), plasticine (pl) and play doe (pd).

dle of the tool is not rigid the force profile could falsely identify a penetration into the target object.

Force modeling and monitoring during the interaction, provide coarse and qualitative results about penetrability as related to the functionality of piercing and of chopping. Our goal is that of providing a robotic agent with the ability to perform interactions to test functionalities of tools having limited built-in knowledge, but capable of observing its actions and able to reason about mechanical interactions without the need to resort to a large set of complex theories. At the same time, the results of the experiments emphasize that, without the ability to verify hypotheses or acquire actual properties from the objects, the result of an interaction can not be fully predicted.

4 FUNCTIONAL FEATURES

The experiments reveal that not all the tools were able to pierce or chop the target object to a desired

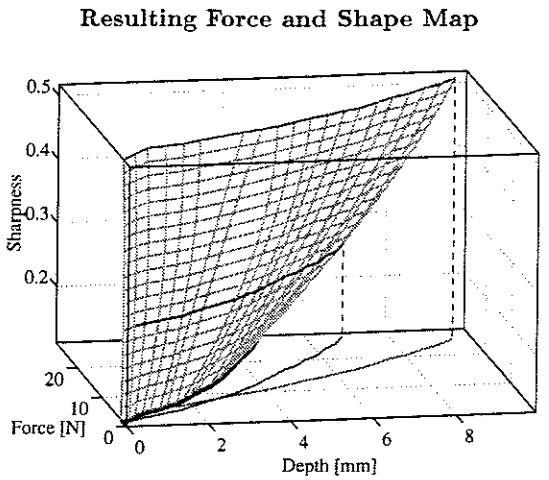


Figure 4: The force profiles extracted from the interaction of each tool with one material (level curves) are interpolated according to the sharpness value of the tools employed.

depth. To identify the properties of tool or target objects responsible for the success or failure several steps are taken. First, the extracted parameters of the shapes are classified according to their discriminating power. The classification defines an ordering of importance in the various parameters are expressed in terms of discriminating power. The shape parameters of the tool, material properties of the target object and the depth penetrated in the experiments are then considered.

The importance of a parameter is captured by its contribution to the success or failure of an experiment. To investigate this contribution, the ratio of successes for a particular value of a parameter to the total number of successes is computed. Thus, the length and width of the handle of the tool are identified as not important while sharpness and the material constituting the target objects are relevant to the success of the interaction.

5 CONSTRUCTING A REPRESENTATION

To integrate the observations gathered from the interactions extracted in the experiments previously described, this section introduces the concept of Force-Shape maps.

Figure 4 shows how the maps for a particular material are constructed. The interpolated surface is obtained by fitting a second order curve to the data points. Each interpolated surface derived from the means of the force signals has an associated lower and upper enveloping surface representing the variability

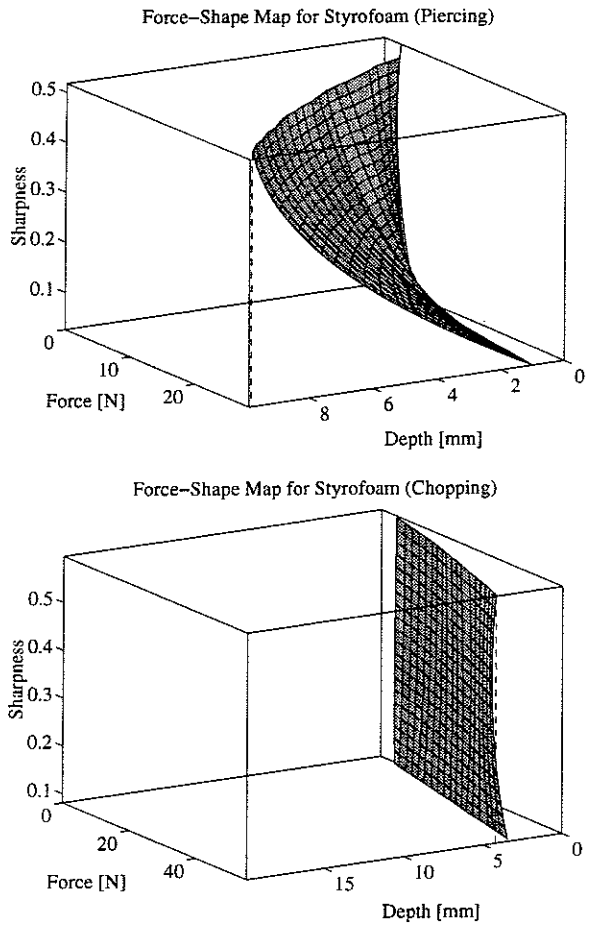


Figure 5: Force-Shape maps for the application of the tools to styrofoam for interactions of piercing (top) and chopping (bottom).

of the signal. The predictive power of the Force-Shape maps depends on the ability to model the actual behavior of the interactions and the ability of extracting shape parameters. Thus, the variability in the extraction of the shape parameters must also be included.

By considering the effect of the different tools on a single material, envelopes can be constructed. These **Force-Shape** maps provide a means for classifying the tools with respect to the shape, the force exerted during the interaction and the task that is currently being investigated.

Figure 5 shows the reconstructed maps for piercing and chopping styrofoam. The multi-dimensional shape description axes is identified in the graphs by sharpness values. The experiments performed provide contour levels which are interpolated to provide a continuous surface. The obtained surface can be used to relate intermediary shapes to their functionality with respect to a specific material.

By considering how the Force-Shape maps for different materials vary, we can construct maps which relate the operation of piercing and chopping carried out by various materials and by different tools. Figure 6 shows a rendering of the combined Force-Shape maps.

While an individual map provides predictive ability for one material, the individual maps may be also interpolated to span a volume. As an aid to visualize these maps and their relations, one can think of the individual maps as pages in a soft-cover book standing upright and partially open and bent. Hence, a particular functional interaction with a given tool may be identified as a line (force profile) whose position in a page would be associated with the shape of the tool and the position of the page in the book would reflect the material being probed.

6 CONCLUSIONS

The experiments discussed in this paper show how an agent may extract functional representations for various objects. The experiments performed here have assumed that the basic task description was provided. The high level description was instantiated in the robotic agent and the observers and the interaction was then performed. The functionalities discussed here are elementary but constitute a basis for constructing more complex ones. In fact, once an agent has extracted several basic functional representations it can then employ the acquired representations to perform more complex task.

The investigative methodology developed allows an agent to probe its environment and extract properties of objects (material, geometric, kinematic and dynamic) which are too often only assumed. This extraction may involve specific exploratory procedures or the actual performing of the functional task. Hence, an actual robotic system allows the flexibility of extracting a representation and to carry out simple functional investigations as well as the ability to verify the functionality of objects, hypothesized by reasoning, by investigating their properties. This type of approach facilitates the connection of the symbolic representation of tasks and functional representation of objects to particular environments in which the tasks are performed and objects are interacted with by real robotic systems.

The formalism of the approach and the distributed nature of the architecture allow for several agents to participate in exploratory tasks. In fact, a team of agents may contribute in different capacity providing expertise and observational power which can not be delivered by a single agent. More than one simultaneous view might be required when interacting with objects possibly having observers that automatically

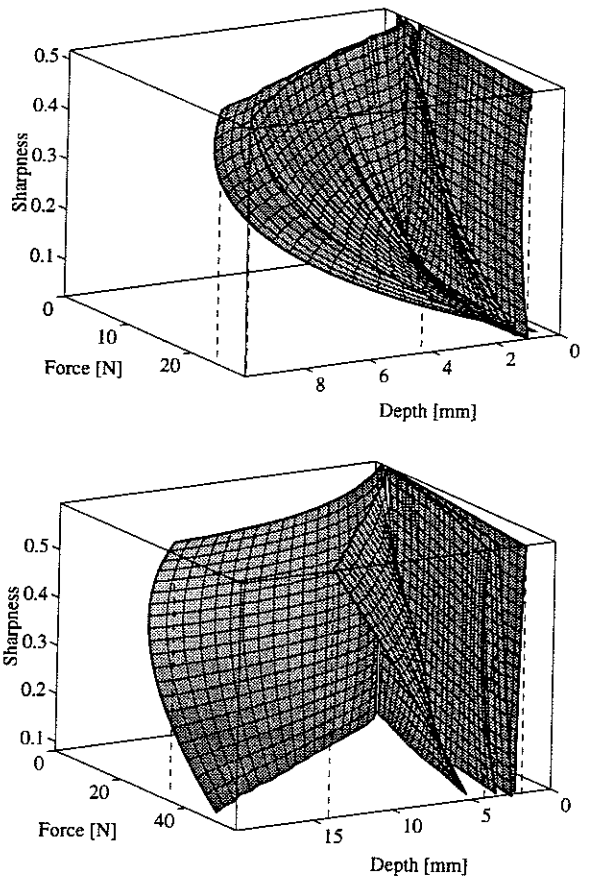


Figure 6: Force-Shape Maps for functionality of piercing (top) and chopping(bottom) for a set of tools and materials.

repositions themselves and select the best viewing loci to provide feedback or observe properties of the manipulated objects.

Acknowledgements ARPA Grants N00014-92-J-1647 DAAH04-93-G-0419; ARO Grants DAAL03-89-C-0031PRI, DAAL03-92-G0153; NASA Grants NGT-50729, NGT-70359.

† **Present Address** Siemens Corporate Research, Inc., Princeton NJ

References

[1] L. Bogoni and R. Bajcsy. An Active Approach to Characterization and Recognition of Functionality and Functional Properties. In *AAAI Workshop: Reasoning about Function*, pages 9–16, 1993.

[2] L. Bogoni and R. Bajcsy. Functionality Investigation using a Discrete Event System Approach. *Journal of Robotics and Autonomous Systems*, 13(3):173–196, 1994.

[3] L. Bogoni and R. Bajcsy. Interactive Recognition and Representation of Functionality. *Computer Vision and Image Understanding*, 62(2):194–214, 1995.

[4] K.W. Bowyer and L. Stark. Beyond Pure Static Shape in Function-based Object Recognition. In *SPIE 2055: Intelligent Robots and Computer Vision*, 1993.

[5] M. Brady, P.E. Agre, D.J. Braunegg, and J.H. Connell. The Mechanic's Mate. In *Advances in Artificial Intelligence: European Conference of Artificial Intellingece*, volume 1, pages 79–94, 1985.

[6] B. Chandrasekaran. Functional Representation and Causal Processes. *Advances in Computers*, 38, 1993.

[7] L. Chittaro, G. Guida, C. Tasso, and E. Toppano. Putting functional knowledge on firmer ground. In *AAAI: Workshop on Reasoning about Function*, volume 1, pages 23–30, 1993.

[8] J.H. Connell and M. Brady. Generating and Generalizing Model of Visual Objects. *Artificial Intelligence*, 31:159–183, 1987.

[9] P.R. Cooper, L.A. Birnbaum, and M.E. Brand. Causal Scene Understanding. *Computer Vision and Image Understanding*, 62(2):215–232, 1995.

[10] J. deKleer. Causal and Teleological Reasoning in Circuit Recognition. Technical Report AI-TR-529, MIT, 1979.

[11] M. DiManzo, E. Trucco, F. Giunchiglia, and F. Ricci. FUR: Understanding FUnctional Reasoning. *International Journal of Intelligent Systems*, 4:431–457, 1989.

[12] K. Green, D. Eggert, L. Stark, and K. Bowyer. Generic Recognition of Articulated Objects by Reasoning about Functionality. In *AAAI Workshop on Representing and Reasoning about Device Function*, 1994.

[13] J. Hodges. Functional and Physical Objects Characteristics and Object Recognition in Improvisation. *Computer Vision and Image Understanding*, 62(2):147–163, 1995.

[14] K. Kise, H. Hattori, F. Akiyama, T. Kitahashi, and K. Fukunaga. Object Recognition Based on Function, Mechanisms and Actions. In *Workshop on Functionality Recognition, CVPR94*, 1994.

[15] E. Rivlin, S. Dickinson, and A. Rosenfeld. Recognition by Functional Parts. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 267–274, 1994.

[16] L. Stark. Recognizing Object Function through Reasoning about 3-D Shape and Dynamic Physical Properties.

[17] M. Sutton, L. Stark, and K.W. Bowyer. Function-based generic recognition for multiple object categories. In *Three-Dimensional Object Recognition Systems*, pages 447–470. Elsevier Science Publishers, 1993.

A Framework for Document Functionality

D. Doermann
Document Processing Group
University of Marland
College Park, MD, 20742

E. Rivlin
Department of Computer Science
Technion Institute of Technology
Haifa, Israel 32000

Abstract

In this paper we introduce the concept of document functionality. Functionality refers to the role of a well designed document and its components in facilitating the transfer of information from an author to a reader. We claim that in general, the role of a document can be characterized by the functions of the documents components. The function of each component can in turn be derived from its physical attributes and from its relationship with other components. If a functional description of a document can be derived, it provides insight into the class of the document, and ultimately into strategies for automatic interpretation.

We show how functional representations can be viewed as an abstraction of the traditional physical and semantic organizations. To illustrate these ideas, we define a partial taxonomy of functional document components and show how they provide important contextual information to aid in interpretation.

1 INTRODUCTION

Written documents have long been the prefered medium for the transfer of information across both time and space. In this sense, the general role or "function" of a document is to store data produced by a sender in a symbolic form to facilitate transfer to a receiver. Traditionally, the data takes the form of a set of markings on a page, with the sender corresponding to the "author", and the receiver to the "reader".

A document can be constructed in a number of ways and one property of a well-constructed document is that it allows the transfer of information to occur efficiently. A set of conventions have developed which allow an author to 1) create a basic document where the primary source of information is contained in the content (raw documents) and 2) to use special constructs to enhance the interpretation of the message through variations in structure and presentation (structured documents).

The basic "block of text" document obeys certain fundamental convensions such as the use of a language and alphabet common to the author and reader, and the use of standard presentation rules such as the existing of word and line spacing, punctuation, etc. As the information which the author presents becomes more complex, the basic language may no longer be the ideal representation typically because the language can not efficiently express the intended message. Fortunately, authors have many tools, which allow them to structure documents in such a way as to maximize efficient transmission and reduce ambiguity. For example, an author may use spatially prominent features such as page or section headers to "summarize" content; a table or graph to present numeric data and the their interrelationships; or an ordered list of text items to explain a procedure. They may even augment the basic language with more expressive constructs such as is done when an author uses a map to present spatial data. Similarly, the author may use stylistic variations of a basic document in terms of class - e.g. business letter or journal article format; layout - e.g. margins or number of columns; and/or emphasis - e.g. italics, boldface and underlining; to enhance the transfer by helping to organize and prioritize the information. Each of these has its own role in the document.

Although a great deal of work has been done in the analysis of document structure, almost all work has involved component level models for specific classes of documents. We claim that in order to make progress in the automated analysis of general classes of documents, a more general framework for interpretation must evolve.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/429 ©1996 FLAIRS

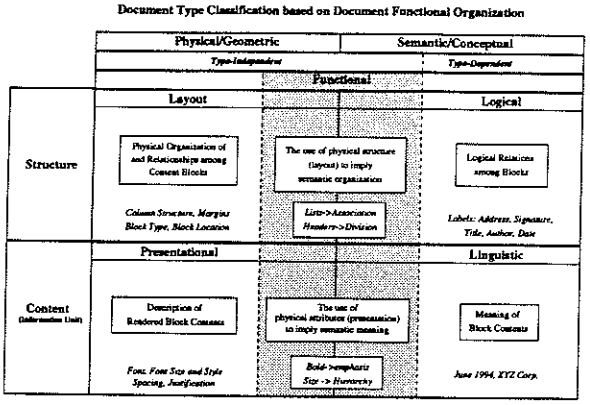


Figure 1: The relationship of Physical, Semantic and Functional descriptions.

2 GENERAL FRAMEWORK

To outline the general theory of document functionality, we first consider traditional views of document organization and how functional description can be viewed as an abstraction of its physical or semantic organizations (functional organization). Second, we must consider how the author and reader are able to use the design of a document to implicitly impose a higher level organization on the underlying language (functional design). Finally, we consider various conceptual levels at which this information transfer can occur and how the reader uses various strategies to achieve different goals (information extraction). Such a characterization provides us with a high-level abstraction in which we can consider more specific document understanding tasks.

2.1 Functional Organization

In document understanding, documents have traditionally been viewed according to either their physical and semantic organizations, as shown in Figure 1. Both organizations have a common *content* which represents a base level of data (typically text, a graphics or an image region) which can be referred to as a single entity dependent on the scope of the organization. Although the content is shared by both organizations, we can refer to the content's "physical" nature as how it is presented on the page (typeface, font size, etc.) and to its "semantic" nature as how its meaning is represented (linguistically, graphically, or visually).

Similarly, the organization of a document has both a physical and semantic component of structure. The *layout structure* corresponds to the geometric organization of a document into physically meaningful features such as lines, blocks, columns, etc. It provides relationships between these components and the relationships of individual components to the entire page. The *logical structure*, on the other hand, organizes the content according to the interpretation of the reader, and uses semantic constructs such as titles, paragraphs, figures and footnotes. It also provides the more global relationships such as reading order. The logical structure corresponds to the semantic or conceptual organization.

The functional organization is complementary to both the physical and the semantic organizations. A document object's *functionality* serves as a description of role of the object in the document. At a high level, the description can be independent of type and class and be based only on general physical features. Headers, footers, lists, tables, and graphics are examples of generic structures which may be common to any number of documents and may be identified from local structure, without knowledge of the structure of the entire page. Such functional descriptors will be referred to as class independent.[1].

If the class is known (for example, business letters, forms or technical articles), the functionality describes the role of a component with respect to the document type. For example, in a letter, functional descriptors may be more specific and may describe components such as the sender, receiver, date, or salutation. Descriptors which rely on models of the entire document and in particular the type, will be referred to as class dependent functional descriptors.

As should be obvious, functional descriptors can be used for components at many levels from single symbols up through much higher level constructs. It is therefore advantageous to define the equivalent of content and structure in the functional domain.

An *information unit* is an abstraction of the physical and semantic content which can be meaningfully referred and is the building block for higher level constructs. The information unit is the base level of representation necessary for the reader to perform a particular task. For example, if the task is to recognize individual characters, the base information unit is typically a single symbol. If the task involves indexing into a phone book, the information unit may be a single listing; or if the task is to describe the organization of a book, the information unit may be a block of text which corresponds to a paragraph or section.

A document component, consisting of one or more information units is called an *information structure*. For example, a list can be composed by a set list items which are in turn composed by a symbolic or numeric "bullet" and a block of text. The description is func-

[1]Some additional examples of class independent functional descriptors are given in Section 2.4.

tional in the sense that a list functions to either order a set of information units sequentially or present them as a set. The function of a table is to present relationships between text items along two axes. We claim that describing a document in terms of the functions of its components, greatly facilitates tasks associated with interpretation.

2.2 Functional Design

Conventions exist for organizing document functionally and it is up to the author to make use of these tools so that the reader can use the document effectively, whatever the task may be. Authors typically use a combination of *class, constructs, and emphasis* to make either their intended organization implicit, or assign priority to specific components.

The type or class of a document to be used is an authors first decision and its choice has several objectives depending on the intended reader and task[2]. A business letter functions to make certain reference information implicitly, such as the sender, receiver and date created. A form on the other hand, functions primarily to solicit information from the reader, and have is unambiguously transfered back to the author.

Within a document different physical constructs aid primarily in the organization of information. A list, for example, suggests a meaningful temporal or set relationship between its items. A figure and corresponding caption is interpreted as a illustration of some concept or fact in the text body. Higher level constructs such as section/subsection, indices, or running heads, aid in organizing documents at a global level.

Finally, techniques exist which can be used in conjunction with the class and constructs to draw (or conversely to suppress) readers attention. At a page level, the author may use headers and increase their point size, use all caps, and/or center them make them more prominent. At a word or phrase level, the author may use bold-face or italic fonts in a similar way to draw attention. Text which is seen as unimportant or which the author would rather not convey may be put in "fine print" with opposite results.

If the reader understands these conventions, they can use them for interpretation.

2.3 Information Extraction

The consumers of a document may have any number of different goals, and may thus abstract the contents at many different levels. The ability to interpret and use the structure of the document in a meaningful way is thus essential. We as humans are very good at resolving ambiguity and establishing a working context within the document space which is task dependent. For example, when looking for documents created on a specific date, we can rapidly locate the dates of documents such as business letters and forms without reading the entire document. If we wish to "read" a document, the context helps with correct interpretation and provides a framework in which to proceed in an orderly fashion through the text, figures, references, etc.

In an automated system, the goals are similar. The user may wish to have the system browse and search for information to answer specific questions, or simply convert the document to a machine manipulable form. When the goal is to browse or search a document, the structure of a component allows us to make some "educated" assumptions about the type of information that component contains and how it is implicitly organized. For example, lists typically provide a sequential ordering of information and figures serve to illustrate a concept in the text.

Let us assume for the moment that the goal of a reader is to extract and interpret the content of a document. The extraction of information can be viewed in three basic ways:

- READING - which usually involves examining the document from beginning to end; this mode is ordinarily used for letters, articles, and most types of books.
- BROWSING - which involves examining only selected parts of the document; this mode is ordinarily used for newspapers, magazines, and, at a high level, journals.
- SEARCHING - which involves looking for a specific piece of information in the document; this mode is ordinarily used for reference books such as dictionaries, encyclopedias, directories, manuals, handbooks, etc.

These ways of interacting typically apply not only with text-intensive documents, but may also apply to documents which are primarily representational such as maps and drawings. It should be noted that we have not restricted the nature of the content, nor the method of interpretation. For example, browsing a newspaper and browsing a map have the same basic goals of obtaining a high level summary of the contents, but the methods and internal which are used to accomplish this are very different. Similarly, searching a phone book and searching a map both require "navigating" and making decisions based on observations of key components. For phone books, one uses index

[2]Examples of types include correspondence (letters or memos), articles, forms, and advertisements, among others.

terms and alphabetical relationships; for maps, one uses symbols or landmarks and spatial relationships.

The three basic ways of interacting can be extended to account for particular goals of the author. For example, when the author needs to gather information via a form, we must consider the case where the reader will interpret and subsequently modify the document.

- MODIFYING - which involves changing a given document either physically or conceptually to produce a new or annotated document. A new document results when the original document is, for example, reformatted or translated into a new language, while an annotated document results from the intentional placement of additional markings to the original page, as with filling out a form.

Various other subclasses, such as editing, can be described for more specific user tasks. Having described how document can be designed and used to extract non-content information, it is useful to highlight some functional descriptors which can be used.

2.4 Functional Descriptors

At this point, we are concerned primarily with functional aspects of the layout (class independent), and how they give us insight into the document's content in the absence of optical character recognition and direct functional analysis of the semantic content[3].

The following tables show examples of attributes and structures which can be used to derive high level functional descriptions.

Structures:

Descriptor	Example	Description
header		
list	enumerated itemized	*conveys temporal precedence *suggests similar level of descriptiveness
separator	white space rule line	*physical and possible semantic Dis-association (same)
attachment	footnote boxed regs side-bar	*supplemental information
illustration	table figure	*supplemental information - preserves 2D associations *typically graphical

[3]Class dependent functional descriptors have been described previously by Taylor [10] who use basic physical construct along with working knowledge of business correspondences to do type classification.

Functional Attributes:

	Type	Examples
Info Structures	text	alignment, indentation, font
Info Units	text	point size, emphasis, justification

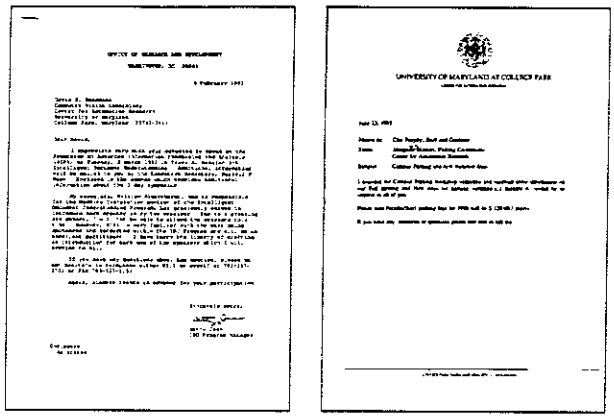
3 SAMPLE DOCUMENT TASKS

Let us assume that we have a goal that can be expressed in general terms about a large corpus of documents. As stated earlier, functional descriptors, derived from physical observations, provide insight into the document in a number of interesting ways. First, an automated system can organize the content prior to doing any type of recognition. This may allow the system to decide which portions of the document should be presented to the user and which to ignore or consider lower priority. Second, many of the relationships which are explicit in the structure can not be found at the content level such as the ordinal relationship between items in a list, or the spatial relationships between columns in table. Third, the use of functional constructs allows the system to avoid using an excessive amount of content level reasoning.

We have methods for extracting the primitives described in Section 2.4 as well as attributes such as bold face and italics from text and are using these features to address such problems Here are several examples of problems which are addressed by identifying functionally meaningful constructs of the document.

Type Classification: Consider the figure 2 which shows the differences between a memo and a letter. Simple functional features such as the head/body pairs explicitly describing the To:, From: and Re: fields, and the location of the handwritten portions are ideal for distinguishing between these two similar classes.

Functional Typing: We have identified four primary high level functional descriptors defined in Section 2.3, readable, browsable, searchable and modifiable. We can claim that the intended functionality can be grossly characterized by the size and organization of the information units and that the information units are identified by repetitive patterns in the document. For example, readable documents such as journal articles, tend to have single read-order with larger information units; browsable documents, such as magazine articles, tend to have multiple head body structures since the goal is to give the reader access to a number of "handles" to the content portions



(a) (b)

Figure 2: Example of the difference between a memo and letter

quickly, and searchable document tends to have very small, repetative information units as would be found in an index, or phone book. Similarly, a form is characterized by small, blank information units such as horizontal line segments, check boxes or boxes.

Document Navigation: Independent of document class, we are developing techniques to present document images to users who want to browse collections of documents. Providing the document or its functionally meaningful to the user in a way which is consistent with how document was intended to be used, or is consistent with the goals of the reader, is important if electronic representations of documents are to be accepted.

4 DISCUSSION

Although functionality can be viewed as simply structural analysis, we feel that a more general framework for analysis must prevail in order to develop truly ingelligent document systems. As one direction, we have provided a basis for understanding and representing the functional aspects of document design and usage. Clearly, the author uses classes, constructs and emphasis to make implicit information in the document. Unfortunately, traditionally document understanding and conversion techniques ignore the intended functionalities of the document, especially class independent functionalities. We see the primary advantages lying in the ability to organize semantic content at a higher level then can be done by interpreting raw text.

We will extend this work to provide a more complete taxonomy of functional primitives and implement a full system for functional typing and document classification.

References

[1] Green, K., Eggert, D., Stark, L. and Bowyer, K.W. Generic recognition of articulated objects through reasoning about potential function, accepted to appear in *Computer Vision and Image Understanding.*

[2] Rivlin, E., Dickinson, S.J., and Rosenfeld, A. Recognition by functional parts, accepted to appear in *Computer Vision and Image Understanding.*

[3] Rivlin, E. and Rosenfeld, A. Navigational functionalities, accepted to appear in *Computer Vision and Image Understanding.*

[4] Rosch, E. Principles of categorization, in E. Rosch and B. Lloyd (Eds.), *Cognition and Categorization*, Erlbaum, Hillsdale, NJ.

[5] Stark, L. and Bowyer, K.W. Function-based generic recognition for multiple object categories, *CVGIP: Image Understanding* **59** (1), 1–21, January 1994.

[6] Stark, L. and Bowyer, K. Indexing function-based categories for generic object recognition, in *IEEE Conference on Computer Vision and Pattern Recognition* (Champaign, IL), pp. 795–797, June 1992.

[7] Stark, L. and Bowyer, K. Achieving generalized object recognition through reasoning about association of function to structure, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13** (10), 1097–1104, October 1991.

[8] Stark, L. and Bowyer, K. Generic recognition through qualitative reasoning about shape and object function, in *IEEE Conference on Computer Vision and Pattern Recognition* (Maui, HI), pp. 251–256, June 1991.

[9] Sutton, M., Stark, L. and Bowyer, K. GRUFF-3: Generalizing the domain of a function-based recognition system, *Pattern Recognition* **27** (12), 1743–1766, December 1994.

[10] S.L. Taylor, M. Lipshutz, and R.W. Nilson. Classification and functional decomposition of business documents. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 563–566, 1995.

Multilayered Sequential Cascaded Networks

John F. Kolen
Department of Computer Science &
Institute for Human and Machine Cognition
The University of West Florida
11000 University Parkway
Pensacola, FL 32514
jkolen@ai.uwf.edu

Abstract

Since their introduction, single-layered higher-order recurrent networks have become popular for the induction of finite state machines (e.g. Giles, Miller, Chen, Sun, Chen, & Lee, 1990). These systems convert sequences of input symbols to sequences of accept/reject decisions. When scaled to larger problems with greater demand for computational resources, these single-layered systems cannot cope with the expected increase of complexity. The architecture described in this paper bypasses these difficulties by layering several higher-order recurrent networks. This approach parallels the layering of perceptron to form multilayered perceptrons capable of implementing a more robust set of input/output mappings. The networks described in this paper were trained using a combination of two error propagation techniques: Pollack's sequential cascaded network method (Pollack, 1991) and traditional back-propagation (Rumelhart, Hinton, & Williams, 1986). Several experiments with the Tomita languages (Tomita, 1982) demonstrated the feasibility of the system.

1 INTRODUCTION

Many varieties of recurrent neural networks exist in the literature. From time-delay neural networks to higher-order recurrent networks (Giles et al., 1990; Pollack, 1991; Watrous & Kuhn, 1992), many researchers have tried to exploit the internal state dynamics of these systems. One application of these mechanisms is the induction of finite state machines (FSM) via a set of positive and negative examples from a regular grammar. In this task, the network must learn to associate a sequence of input patterns (representing symbols from the target language) to a sequence of accept/reject decisions. The collective accept/reject decisions for all possible input symbol sequences defines a language.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/434 ©1996 FLAIRS

While it is possible to implement by hand any FSM in a higher-order recurrent network, there many difficulties conspire against training the network to induce the proper machine (Pollack, 1991). Some have explored alternative learning strategies for solving this problem. For instance, Giles and colleagues (Giles, Miller, Chen, Sun, Chen, & Lee, 1992) extended Williams and Zipser's (1989) idea of forward-propagating gradient information through the higher-order recurrent network. This paper, however, describes an architectural solution to this problem. Rather than have a single time-dependent mapping carry all computational the burden, I suggest that the proper approach to this problem involves the composition of multiple layers of sequential cascaded networks. The remainder of this paper describes multilayered sequential cascaded networks.

2 THE ARCHITECTURE

A sequential cascaded network (SCN) is a second-order recurrent network defined by two three-dimensional weight matrices (Pollack, 1991). The dynamic describing the network's behavior appears in Equation 1.

$$\begin{aligned} O(t) &= g(U \cdot \begin{bmatrix} S(t) \\ 1 \end{bmatrix} \cdot \begin{bmatrix} I(t) \\ 1 \end{bmatrix}) \\ S(t+1) &= g(V \cdot \begin{bmatrix} S(t) \\ 1 \end{bmatrix} \cdot \begin{bmatrix} I(t) \\ 1 \end{bmatrix}) \end{aligned} \tag{1}$$

The U matrix embodies the mapping between current state and current input to current output. The V matrix captures the mapping from current state and current input to the next state. These weight matrices are both three dimensional. Taking two inner products of the weight matrices has the same effect as performing pairwise multiplication between the state and input vector elements. Hence, the SCN performs a second order mapping. The function g is a nonlinear squashing func-

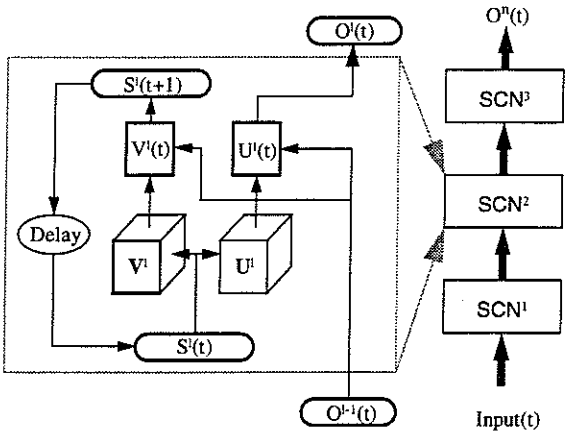


Figure 1: The Multilayered Sequential Cascaded Network.

tion, such as the sigmoid or hyperbolic tangent. The SCN is similar to the architectures described by Giles et al. (1990, 1992) and Watrous and Kuhn (1992).

In its traditional configuration, the SCN processes the raw input vector to calculate the network's output. This constraint limits the SCN to computing instantaneous input/output mappings (Kolen, 1994) no more complex than that of a single-layered perceptron[1] and prevents the SCN from representing time-dependent linearly inseparable mappings. For instance, the SCN could not represent a two-input one-output mapping that alternated between exclusive-or and exclusive-nor. The multilayer sequential cascaded network (MSCN), however, can represent this complex mapping. The MSCN consists of several SCNs layered as in Figure 1. Activation flows through the network from input layer to output layer, reminiscent of the multilayered perceptron. As activation flows through a layer, the internal state for that layer, l, is updated.

$$
\begin{aligned}
O^l(t) &= g(U^l \cdot \begin{bmatrix} S^l(t) \\ 1 \end{bmatrix} \cdot \begin{bmatrix} O^{l-1}(t) \\ 1 \end{bmatrix}) \\
S^l(t+1) &= g(V^l \cdot \begin{bmatrix} S^l(t) \\ 1 \end{bmatrix} \cdot \begin{bmatrix} O^{l-1}(t) \\ 1 \end{bmatrix}) \\
O^0(t) &= Input(t)
\end{aligned}
\qquad (2)
$$

1. The inner product of the weight matrix and state vector is a two-dimensional matrix identical to the two dimensional matrix controlling the behavior of a single-layered perceptron.

Unlike the multilayered perceptron, the internal state within each layer of the MSCN allows the layer to perform a time-dependent transformation of the current input. The internal state multiplicatively modulates the connections from input to output in each layer, producing different input/output mappings at different times. This modulations makes a wide range of mappings, from stationary to chaotic, available to the layer (Kolen, 1994).

3 TRAINING

While the MSCN performs time-dependent transformations, it is to our advantage to focus on the instantaneous behavior of the network when considering the problem of training the network. The current state of each layer reduces the three-dimensional weight matrix, U, into a two-dimensional matrix, U', used to calculate the net input to the output units. This matrix is identical to the matrices found in multilayered perceptrons. Hence, we can back-propagate error to each layer using U' and the standard equations found in (Rumelhart et al., 1986) (Equation 3).

$$
\begin{aligned}
\delta^n &= g'(O^n(t)) \cdot (O^*(t) - O^n(t)) \\
\delta^l &= g'(O^l(t)) \cdot (U'^{l+1} \cdot \delta^{l+1}) \\
U'^l &= \left(U^l \cdot \begin{bmatrix} S^l(t) \\ 1 \end{bmatrix} \cdot I_{dim(O^{l-1})+1,\, dim(S^l)} \right)^T
\end{aligned}
\qquad (3)
$$

The function g' is the two-dimensional derivative matrix[2] for the function g applied to a vector. The vector $O^*(t)$ is the desired vector at time t. The matrix I is zero everywhere except when $i = j$ ($I_{n,n}$ is the identity matrix). This matrix strips the input bias component from the weight matrix.

Once the error vectors computed for each layer, these deltas are used to calculate the gradients for both the output and next state weights. Calculating the output weight gradient involves an outer product of the current input, state, and error vectors. Calculating the gradient for the state weights, however, requires additional processing. While several approaches to this problem exist (e.g., Williams & Zipser, 1989), the current MSCN implementation relies upon Pollack's method of propa-

2. $g'_{i,j}(x) = g'(x_i)$ when $i = j$ and zero everywhere else.

gating error only one time step back (Pollack, 1991).

$$\frac{\partial E(t)}{\partial U^l} = \delta^l(t) \times \begin{bmatrix} O^{l-1}(t) \\ 1 \end{bmatrix} \times \begin{bmatrix} S^l(t) \\ 1 \end{bmatrix}$$

$$\frac{\partial E(t)}{\partial V^l} = \frac{\partial E(t)}{\partial S^l(t)} \cdot \frac{\partial S^l(t)}{\partial V^l}$$

$$\frac{\partial E(t)}{\partial S^l(t)} = I_{dim(S^l),\, dim(S^l)+1} \cdot \begin{bmatrix} O^{l-1}(t) \\ 1 \end{bmatrix} \cdot (\delta^l(t)^T \cdot U^l) \tag{4}$$

$$\frac{\partial S^l(t)}{\partial V^l} = g'(S^l(t)) \times \begin{bmatrix} O^{l-1}(t-1) \\ 1 \end{bmatrix} \times \begin{bmatrix} S^l(t-1) \\ 1 \end{bmatrix}$$

4 EMPIRICAL VALIDATION

Several experiments exploring the behavioral, representational, and training aspects of MSCNs are currently underway. The results reported here focus on the feasibility of the approach by comparing SCNs and MSCNs while learning the Tomita languages (Tomita, 1982). The Tomita data consist sets of positive and negative example strings from seven regular languages[1]. While Tomita originally used these simple languages to demonstrate the capabilities of an evolutionary algorithm for learning finite state automata, this data set has emerged as a benchmark for testing recurrent networks (Pollack, 1991; Giles, et al., 1992; Watrous & Kuhn, 1992).

The Tomita languages have binary alphabets. Hence, The networks trained upon languages each have one input unit and one output unit. The input symbols "0" and "1" are represented by input vectors <0> and <1>. A string (the sequence of input symbols processed by the network) is considered accepted by the network if the output is greater than 0.5 (when using the sigmoid), otherwise it is rejected. The target output vector for rejection was <0> and the target for acceptance was the vector <1>. Training was stopped when the network could correctly discriminate the positive and negative exemplars. Others have trained their networks to tighter bounds (e.g. below 0.2 for reject and above 0.8 for accept (Pollack, 1991)), however, their interests focused on the inductive capabilities of the networks--will the network find the underlying FSA that generated the small set of training examples. The goal of this experiment was to demonstrate the feasibility of the training mechanism, not its inductive prowess.

Each of the seven Tomita languages was presented to several different networks for each architecture (SCN and MSCN). Eleven unique initial weight assignments drawn from eight different initialization distributions (uniform distribution between +/- 0.1, 0.3, 0.5, 0.7, 1.0, 2.0, 3.0, 5.0) produce eight-eight learning situations for each language. The weight assignments are identical between I selected this set of initialization ranges since traditional back-propagation of error through multilayered perceptrons displays a wide variety of convergence patters over this range (Kolen & Pollack, 1991). The set of SCNs do not share the same initial weights with the MSCNs. The networks, however, do share identical weight assignments within the architectural sets. Again, the purpose is the ameliorate any effects of initial condition selection.

Each SCN has three state units (ala (Pollack, 1991)). The MSCN has three hidden units in addition to three state units in each layer. The training algorithm gathered gradients for each string and modification of weights did not occur until all training strings had been presented (epoch) and employed the momentum method of updating the weights (Rumelhart, et al, 1986). All trials used learning rates of 0.3 and momentum rates of 0.7 for all layers in the networks.

Tables 1 and 2 report the results of this experiment. Table 1 compares the number of networks that were able to discriminate the exemplars within 10,000 passes through the training set. Table 2 reports on the mean number of epochs for the successful networks. In compiling data for Table 2, only those networks that actually separated the outputs correctly were considered. Cells in the lower right-hand corner cells of the tables summarizes the overall behavior of the two architectures. First, the overall convergence percentages of both architectures indicates that the MSCNs used in this study converged more often than their SCN counterparts (88.3% vs. 77.8%). While this property holds across weight ranges, in some data sets the difference was minimal (sets 1 and 3) and others the difference was striking (set 6). Similar results were found with mean training time over converged networks. The overall training time was 511.8 epochs for the MSCN and 644.9 for the SCN. This relation ship generally held across weight ranges and data sets with the exception of data sets 1 and 3 and weight range 0.3. The latter case can be explained by skewing of the mean caused by the SCNs inability to learn Tomita 6 with a weight range of 2.0 or less. The

1. Each language could be described by a finite state automata with no more than nine states.

other two cases may indicate that Tomita 1 and 3 are more amenable to SCN encoding than the other training sets.

5 CONCLUSION

A combination of PDP-style feed-forward networks and second-order recurrent networks has been described. One purpose of this extension was to avoid the representational constraints of a single layered system. The original SCN, and its higher-order brethren, performed time-dependent convex mappings. MSCNs, on the other hand, perform the same arbitrary time-dependent mappings that multilayered perceptrons can. In the experiment described above, it is unclear how this particular advantage played any role in the significant difference in convergence percentage and convergence time. One possible explanation suggests that the dynamics of the output layer is different from the input layer. The input layer receives one of a finite set of input patterns (<0>,<1>). This constraint forces the input layer to behave like an iterated function system (Barnsley, 1988) by reducing the three-dimensional matrices to an input-symbol indexed set of two-dimensional matrices (Kolen, 1994). The set of output patterns generated by this layer may or may not be finite. If it is not finite, the output layer may receive unique patterns at each time step. This enumeration could allow the output layer to discover an appropriate mapping easier than the single case.

The MSCN offers many opportunities for future theoretical and applied work. From the theoretical side, the infinite set of patterns flowing into the second and subsequent layers of this network makes for a interesting analysis task. One might say that the MSCN has the capability of constructing and utilizing dynamic and distributed representations. Unlike the hidden unit representations of multilayered perceptrons, MSCN representations may not be stable. Yet the network is able to perform its information processing task by entraining the time-dependent behaviors of the individual layers. Exploration of such representations may provide artificial intelligence and cognitive science many new insights into the mechanisms of intelligence.

The MSCN is not without practical purposes. Currently, two projects are underway utilizing this architecture. The first involves analysis of helicopter pilot instrument-gaze patterns. In this task, the network must predict the next instrument pilot will look at given the current flight task and previous gaze history. Another project underway regards natural language acquisition. This project will test the networks ability to acquire long, but not unbounded, range dependencies from corpora. The MSCN may be able to perform this task since the it can form different attractors at each layer. These attractors may be able to encode word, sentence, and pragmatic structure. As we begin to understand the behavior and representational capabilities of this new architecture, additional applications will, no doubt, be found.

References

Barnsley, M. (1988) *Fractals Everywhere.* Academic Press.

Giles, C. L., Miller, C. B., Chen, D., Sun, G. Z., Chen, H. H. & Lee, Y. (1992). Extracting and Learning an Unknown Grammar with Recurrent Neural Networks. In John E. Moody, Steven J. Hanson & Richard P. Lippman, (Eds.), *Advances in Neural Information Processing Systems 4.* Morgan Kaufman.

Giles, C. L., Sun, G. Z., Chen, H. H., Lee, Y. C. & Chen, D. (1990) Higher order recurrent networks and grammatical inference. In D. Touretzky, (Ed.), *Advances in Neural Information Processing Systems* 2. Morgan Kaufman.

Kolen, J. F. (1991) Back Propagation is Sensitive to Initial Conditions. *Complex Systems, 3.*

Kolen, J. F. (1994) *Exploring the Computational Capabilities of Recurrent Neural Networks.* Ph.D. dissertation. The Ohio State University.

Pollack, J. B. (1991). The induction of dynamical recognizers. *Machine Learning, 7,* 227-252.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323,* 533-536.

Tomita, M. (1982). Dynamic construction of finite state automata from examples using hill-climbing. In *The Proceedings of the Fourth Annual Conference of the Cognitive Science Society.* Seattle.

Watrous, R. L. & Kuhn, G. M. (1992). Induction of Finite-State Automata Using Second-Order Recurrent Networks. In John E. Moody, Steven J. Hanson & Richard P. Lippman, (Eds.), *Advances in Neural Information Processing Systems 4.* Morgan Kaufman.

Williams, R. J. & Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation, 1,* 270-280.

Table 1: Number of initial conditions that converged before 10000 training epochs

Tomita Set	Arch	0.1	0.3	0.5	0.7	1.0	2.0	3.0	5.0	Total
1	mscn	11	11	11	11	11	11	11	11	100.0%
	scn	11	11	11	11	11	11	11	10	98.9%
2	mscn	10	10	10	11	11	10	9	7	88.6%
	scn	9	9	6	10	8	7	7	5	69.3%
3	mscn	11	10	11	11	11	11	11	10	97.7%
	scn	11	11	11	11	11	11	11	9	97.7%
4	mscn	11	11	11	11	11	11	11	10	98.9%
	scn	9	10	10	10	11	10	11	10	92.0%
5	mscn	10	11	11	10	10	10	10	8	90.9%
	scn	11	10	10	10	9	11	9	8	88.6%
6	mscn	1	3	3	3	6	8	8	6	43.2%
	scn	0	0	0	0	0	0	3	4	8.0%
7	mscn	11	11	11	11	11	11	11	10	98.9%
	scn	11	10	10	11	10	9	9	9	89.8%
Total	mscn	84.4%	87.0%	88.3%	88.3%	92.2%	93.5%	92.2%	80.5%	88.3%
	scn	80.5%	79.2%	75.3%	81.8%	77.9%	76.6%	79.2%	71.4%	77.8%

Table 2: Average number of training to convergence.[a]

Tomita Set	Arch	0.1	0.3	0.5	0.7	1.0	2.0	3.0	5.0	Total
1	mscn	102.4	88.5	83.6	78.9	77.5	64.3	164.8	1164.9	228.1
	scn	80.4	74.0	70.1	67.9	67.4	112.6	337.4	456.9	154.9
2	mscn	408.0	1043.6	974.9	654.9	332.1	432.7	687.1	1213.4	693.9
	scn	2429.6	2488.7	3242.0	2872.5	2463.0	3182.1	662.7	714.6	2338.2
3	mscn	481.7	474.2	263.0	177.5	135.5	121.2	184.7	474.9	284.8
	scn	123.8	120.8	103.9	106.9	116.5	103.2	190.6	976.8	213.0
4	mscn	269.0	258.5	165.9	116.4	95.6	74.3	144.8	293.8	176.0
	scn	215.3	219.9	240.3	154.0	407.2	230.5	219.5	249.1	244.1
5	mscn	472.2	730.0	343.4	336.3	298.0	252.2	606.0	2253.6	618.5
	scn	1120.8	388.8	374.2	478.3	401.1	423.7	466.3	2144.6	697.0
6	mscn	3925.0	4052.3	4115.0	2442.7	1460.7	2069.0	1324.6	2343.5	2256.0
	scn	***[b]	***	***	***	***	***	3093.3	3124.0	3110.9
7	mscn	164.7	494.7	177.3	426.6	171.1	168.4	156.9	983.7	335.6
	scn	501.1	301.8	245.9	740.9	441.1	631.3	333.3	706.3	488.5
Total	mscn	368.0	666.0	492.0	392.5	291.2	390.4	422.6	1143.9	511.8
	scn	707.9	551.6	516.7	716.2	570.4	632.2	481.0	1007.9	644.9

a. Networks that did not converge within the time limit are not counted.

b. No SCN converged within 10000 epochs for weight ranges between 0.1 and 2.0 on Tomita 6.

Speaker Verification Via Self-Configuring Neural Networks

Michael Sharkey and Lawrence O. Hall
Department of Computer Science and Engineering
University of South Florida
Tampa, FL 33620
hall@csee.usf.edu

Abstract

The goal of the work reported here is to verify a person's claimed identity based upon a spoken utterance. To achieve that objective, a single neural network is trained to perform the speaker verification task for a single target speaker. The experiments reported here use the self-configuring Cascade-Correlation neural network learning algorithm. Cepstrum coefficients, extracted from recorded speech, were used as the features for the neural network. Silence removal and a new technique to remove common speech sounds were applied to utterances. Results indicate that Cascade-Correlation is capable of effectively performing speaker verification under either scenario with a true positive rate greater than 90% and a low false positive rate (below 1.5%). The best results were obtained with common sounds removed. The small number of weights and configuration information could be stored on a credit card, for instance, rendering this approach practical.

1 Introduction

Speaker recognition is normally divided into the two categories of speaker identification and speaker verification. This paper will deal only with the area of speaker verification, where the objective is to verify a person's claimed identity based upon a spoken utterance. The experiments reported here use the self-configuring Cascade-Correlation neural network learning algorithm [3]. Cascade-Correlation proved itself capable of handling a larger training set, was faster, more reliable during training efforts, and provided better results during the actual verification tests than fixed architecture neural networks.

The feature extraction process for the utterances used in this paper mirrors that described in [1]. Silence removal though, is done after feature extraction on the neural network input via a trained fixed architecture Quickprop network [2]. Two cases are then considered: a) Scenario 1 uses the feature vectors from the spoken sentences after those deemed silence are removed; and, b) Scenario 2 uses the feature vectors from Scenario 1, but further reduces their number by removing additional vectors that are considered 'common' across multiple speakers.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/439 ©1996 FLAIRS

2 Feature Extraction

The spoken sentences used in this work were identical to sentences from the DARPA TIMIT database [6]. However the sentences were re-recorded by us, using new speakers [17]. One reason for this was we wished to have the same sentence spoken multiple times by the same speaker, at the same recording session, as well as across multiple days. In the TIMIT database each speaker says any given sentence only one time.

The spoken sentences were recorded using a Macintosh 520 laptop, using its internal microphone, in a small conference room adjacent to the main Computer Science department office. While the room was relatively quiet, it was **not** a sound-proof recording booth. Each recording was re-played to insure the absence of any clearly audible noise, it is certain that some noise exists as part of the data.

After each sentence was recorded the editing ability of the application SoundEffects [4] was used to remove most of the silence from the beginning and end of the recorded sentence. The sentences are initially recorded in Macintosh AIFF format at 22 KHz sampling frequency. The conversion program SOX [5] was used to convert the sentences to BIN format and to down-sample to 16,000 Hz. At this point a silence removing algorithm was **not** applied.

A pre-emphasis filter $H(z) = 1 - 0.95 * z^{-1}$ was applied. This was followed by a 30 ms Hamming Window applied to the speech every 10 ms. Each window / speech frame was subjected to 12th-order linear predictive (LP) analysis [1, 14]. Feature vectors of 12 values were obtained by the derivation of 12 cepstral coefficients from the LP polynomial.

3 Silence Removal and Removal of Common Sounds

Silence removal is done after feature extraction by a Quickprop neural network that was trained explicitly for

that purpose. The architecture of the Quickprop network was 12 input units, 16 hidden units, and 2 output units.

Recorded silence was given as the target and trained against vectors from normally spoken sentences that contained as near as possible, no silence, pauses, or hesitations. Tests showed that unambiguous silence is removed by the trained Quickprop network, and further, that the vectors representing actual speech remained. Only those vectors which are considered non-silence (classified non-silence with output strength ≥ 0.7 and non-target < 0.5) are then used for any future training or testing.

There are many possibilities for the identification of 'common' vectors amongst speakers. Our experiments showed that the methodology used could produce vastly different results for a given speaker's sentences. Certainly the results between speakers could be varied widely based upon the methodology and threshold used.

Our approach is to compare a target vector against a set of non-target vectors. If the target vector *matches* in order, feature value by feature value, any non-target vector it is placed in the common set. A *match* between values means that they agree within a defined threshold. For these experiments the threshold was defined to be plus/minus 0.15. The vector sets used as input to the common set creation process for both the target and non-target speakers were the feature vectors remaining after the vectors identified as silence were removed.

In our experiments a different common set was developed for each target speaker using 8 sentences from the target speaker and 8 sentences from each of the 4 non-target speakers. A point was made to include in these sets, all of the sentences that would later be used to train the neural network for the speaker verification task. This should help eliminate to some degree, the closeness observed to be present in the target and non-target vectors that initially caused the neural network to take a long time to train and occasionally fail to correctly identify speakers in testing.

On average for a typical target speaker, after silence removal, a total of 72 vectors were identified as common and removed from the sentences from which the common set for that speaker was created. Other sentences for that target speaker, from other recording sessions, had on average, a total of 38 vectors per sentence identified as common and removed. Using the common set for the above target speaker on the four non-target speakers, caused on average, the additional removal of 36, 28, 27, and 15 vectors from their respective sentences.

4 Speaker Verification

4.1 General

Initial experiments showed that Cascade-Correlation would out-perform (in terms of training time and testing accuracy) any fixed Quickprop architecture on both the DARPA TIMIT database [6] and on our originally recorded sentences. The end result was the choice of the Cascade-Correlation neural network for speaker verification, after silence removal via a trained Quickprop neural network, and in some cases, common removal via a C program written to perform that task.

For Experiments 1 and 2, sentences were recorded using a set of 5 speakers (s1, s2, s3, s4, and s5). All five were adult white males with no pronounced regional accents. Each time a speaker recorded a set of sentences, they spoke the same 8 sentences. These 8 sentences were taken from the DARPA TIMIT database with 3 of the 8 being the phonetically rich 'si' labeled sentences and the other 5 the more random (literary) 'sx' labeled sentences. Furthermore, speakers s1 and s2 spoke the same 8 sentences while the other 3 speakers all recorded unique sentences.

Four recording sessions were held across three days as follows:

- Day 1a: A morning recording session with Speaker 1 recording 3 sets of 8 sentences (labeled as target speaker 's1a') and Speaker 2 recording 2 sets of the same 8 sentences (label 's2a').
- Day 1b: Same day, afternoon recording session. Speaker 1 records 2 sets of the 8 sentences (label 's1b') and Speaker 2 records 1 set of the same 8 sentences (label 's2b').
- Day 2: One recording session held with Speaker 1 recording 2 sets of 8 sentences (label 's1c'), Speaker 2 recording 1 set of same 8 sentences (label 's2c'), Speaker 3 recording 2 sets of 8 sentences (label 's3'), and Speaker 4 recording 2 sets of 8 sentences (label 's4').
- Day 3: One recording session held with Speaker 1 recording 2 sets of 8 sentences (label 's1d'), and Speaker 5 recording 2 sets of 8 sentences (label 's5').

Therefore the speakers recorded various numbers of sentence sets, at 8 sentences per set, as follows: a) Speaker 1 recorded 9 sets across 4 recording sessions; b) Speaker 2 recorded 4 sets across 3 recording sessions; and c) Speakers 3, 4, and 5 each spoke at only one recording session where they each recorded 2 sets. Again, each time a speaker recorded a set of sentences, they recorded the *same* set of 8 sentences [17]. The recording strategy allowed for a speaker to be tested against sentences trained from the same recording session, a different session on the same day, and on sentences recorded on different days. A total of 152 (non-unique) spoken sentences were recorded.

4.2 Experiment One

Based upon the recording session/speaker pairs, a total of 10 target speakers exist (s1a, s1b, s1c, s1d, s2a, s2b, s2c, s3, s4, and s5) of which 5 are unique people and the other 5 target speakers consist of specific session recordings from one of the 2 people recorded in multiple sessions. Feature vectors were formed for all of the sentences spoken by these speakers and silence was removed from those sentences using the Quickprop neural network that had been trained for that purpose.

Ten Cascade-Correlation neural networks were trained, one for each of the target speakers. Input to a given neural

network consisted of the feature vectors representing one of the sets of 8 sentences spoken by that target speaker and as negative examples, 2 sentences from each of 4 different non-target speakers. When available, the two sentences chosen for a non-target speaker came from the same recording session as those spoken by the target speaker.

While vectors from entire sentences were used, minus silence, the sentences chosen for the non-target speakers were shorter than those of the target speaker so that the number of feature vectors provided to the neural network was approximately 3:2 in favor of the target speaker.

Finally, all of the recorded sentences for Experiment 1 and 2 were tested against each of the ten trained networks. Note that the 16 sentences represented in the training set are not reported as part of the test data since they were always correctly identified as target or non-target at 100%.

4.3 Experiment 2

Exactly the same sentences, target speakers, and procedures as in Experiment 1 were used with one difference. After a sentence under-went silence removal it also had the common removal process run on it, before it was used in either the training or testing procedures.

Therefore, ten different Cascade-Correlation neural networks were trained with the same 16 sentences used to train each of them as in Experiment 1, except that each of those 16 sentences also had 'common removed' based upon the common set developed for the relevant target speaker.

Similarly, all the test sentences also had common removed. The objective is to compare the results of the two experiments where the only major difference was the common removal process, as used here. Note that while the training data was still weighted towards the target speaker, some of the input sets approach a 1:1 ratio between the target and non-target speakers.

5 Results

5.1 Training and testing on target speaker utterances

The use of Cascade-Correlation allowed the creation of networks from training set sizes up to 5,500 input vectors. The number of nodes used in the networks (with only silence removed) ranged from 70 to 96 with 84 being the average. Training times averaged approximately 17 hours on Sun Sparc 10s that were unloaded about half of the time.

Similarly for the networks that had common vectors removed in addition to silence vectors, the number of nodes ranged from 51 to 84 with an average of 72. Training times for these networks averaged only 9 - 10 hours on SUN Sparc 10s that were unloaded about half the time. Training for networks larger than 5,500 input vectors was not attempted.

In Tables 1A and B, results are provided for each of the 10 trained networks when tested against the target speaker they were trained to verify/recognize. The values shown in the tables are the average acceptance percentages for the tested sentences. An acceptance percentage consists of (after the removal of any 'silence' or 'common' vectors), the number of feature vectors from a sentence correctly recognized as belonging to the target speaker divided by the total number of feature vectors from the sentence.

The test sentences consist of all the sentences spoken by a specific target speaker and not used in training. For example, the network for Speaker s1a is trained on sentences taken from the target speaker from Day 1a. It can be tested on all other sentences spoken by Target Speaker 1a including sentences from the same recording session that are not used in training.

The purpose of this experiment is to determine how consistent the trained neural network is in recognizing or verifying the target speaker. Note that identical sentences spoken on the same and different days/sessions are used. It can be seen that the recognition percentages are generally above 70% when only silence vectors are removed. A percentage of 70 means that 70% of the feature vectors for a sentence were recognized as being from the target speaker.

There are 26 tests[1] that were done and in only 4 of them was the acceptance percentage less than 70%. In only 1 case was the set of sentences recognized at less than 70% taken from the same day (Target Speaker 5), which indicates that the generalization capability of this network is not as good as the others. In all cases the recognition percentage is greater than 60%.

In Tables 1A&B it can be seen that with common vectors also removed, 21 of the test cases had an average acceptance percentage of greater than 68%. Again only Target Speaker 5 had a low acceptance ratio for sentences said in the same session as the test sentences. This network did not generalize as well as the others. In general, the networks trained to verify the presence of Speaker 5 are the worst performers.

5.2 General Results - Experiments 1 and 2

As can be seen from Tables 1A&B, the common removal process of Scenario 2 does depress the acceptance percentage of the test utterances/sentences spoken by a given target speaker almost across the board.

Tables 1A&B show that it is possible to train a Cascade-Correlation neural network to recognize a target speaker. Even further, it shows that the same speaker is capable of being recognized when they speak on different days than the session/conditions they were originally recorded in.

[1] From a possible total of 40 (sentences from 4 recording sessions applied to 10 trained networks). Not all speakers were recorded in all recording sessions. Each test consists of all sentences spoken by a speaker in a specific session that were not used in training, tested agains a trained network for the same speaker.

Target Speaker	Day 1a		Day 1b	
	Silence	Common	Silence	Common
s1a	79.0 t	77.3 t	63.6	60.1
s1b	74.0	72.1	72.1 t	69.8 t
s1c	78.2	75.9	71.3	65.8
s1d	78.6	76.4	64.2	61.6
s2a	77.1 t	76.5 t	71.5	68.9
s2b	82.0	79.5	t	t
s2c	75.0	74.8	75.3	71.8

A

Target Speaker	Day 2		Day 3	
	Silence	Common	Silence	Common
s1a	73.6	71.4	72.2	69.8
s1b	73.9	72.7	69.4	66.2
s1c	79.1 t	77.6 t	75.0	73.0
s1d	76.1	72.8	73.8 t	71.6 t
s2a	71.2	69.6	—	—
s2b	82.4	79.3	—	—
s2c	t	t	—	—
s3	71.6 t	69.8 t	—	—
s4	72.8 t	68.6 t	—	—
s5	—	—	65.1 t	57.9 t

B

Table 1: Average Acceptance Percentage Of Feature Vectors From Sentences Not In The Train Set. Note: A 't' indicates the session the target's training data sentences were taken from. In some instances, this left no sentences from that session to test against.

It is certainly true that noise present in the recordings must have depressed the values from some sessions and definitely for at least a few of the individual sentences. Still, given the open recording environment, the percentages achieved seem very reasonable.

Tables 2 and 3 describe the performance of the trained networks when tested with feature vectors representing target speakers as well as those representing non-target speakers. Table 2 provides for the indicated ranges, the acceptance percentages of all the target and non-target sentences across all twenty trials (10 for Scenario 1/silence, and 10 for Scenario 2/common removed also).

The numbers in the table reflect the fact that 152 sentences were recorded by the 5 speakers, of which 16 of the sentences were used for training each neural network, leaving 136 sentences available as test data for each of the trained networks. The total number of test sentences across the 10 trained networks (Silence or Common), was 352 sentences[2] for the target speakers versus 1008 sentences for the non-target speakers. The imbalance is due to the fact that Speaker 1 recorded 72 of the sentences and appears as the target speaker for 4 trained networks, Speaker 2 recorded 32 of the sentences and appears as the target speaker for 3 trained networks, and Speakers 3, 4, and 5 each recorded 16 of the sentences and appear as the target speaker for 1 trained network.

[2]These are not unique sentences. That is, sentences not used for training from a target speaker will be used in testing against all trained networks for that target speaker. The same sentences will also serve as non-target test sentences for speakers other than the speaker who said the sentences.

Acceptance Percentage Range	Number Of Sentences			
	Silence		Common	
	Target	Non Target	Target	Non Target
$70 \leq X < 99$	261	1	222	0
$65 \leq X < 70$	55	8	62	2
$60 \leq X < 65$	27	22	39	10
$55 \leq X < 60$	6	46	22	23
$50 \leq X < 55$	2	45	5	30
$00 \leq X < 50$	1	886	2	943

Table 2: Cumulative Results - All Tests

Accept percent	Number Of Sentences			
	True Positives		False Positives	
	Silence	Common	Silence	Common
70%	74.1	63.1	0.1	0.0
65%	89.8	80.7	0.9	0.2
60%	97.4	91.8	3.1	1.2
55%	99.1	98.0	7.6	3.5
50%	99.7	99.4	12.1	6.4

Table 3: Acceptance Percentages At Different Threshold Choices. Note: Values are the percentage of total sentences that fall in the given category.

Both scenarios, silence removal and common removal, show a fair separation between the target and non-target speakers - with some overlap. The existence of an overlap poses the normal question of where to place the acceptance/rejection threshold to balance the number of acceptable False Negatives versus False Positives.

Using Table 3, we can see that demanding a True Positive rate of approximately 90% would require a threshold of 65% for Scenario 1 and one of 60% for Scenario 2. In these instances the percentage of False Positives would be 0.9% for Scenario 1 and 1.2% for Scenario 2.

Avoiding complex hierarchies while still achieving a high true positive to false negative ratio was one of our major intents and we believe that the results indicate that our approach, using silence or common removal, is a viable one. The removal of common vectors in addition to silence removal allows for a choice of threshold which gives a higher true positive rate with little increase in the false positive rate. For example from Table 3, a 55% threshold, using common removal, allows a 98% true positive rate versus a relatively low 3.5% false positive rate.

6 Summary

Using Cepstrum-Coefficients to generate feature values, it was our intent to show that a single neural network was capable of performing the task of speaker verification. As indicated by the results of Experiments 1 and 2, it is possible when one uses Cascade-Correlation as the neural network architecture. It is true that total separation between target and non-target speaker sentences

was not achieved - though the number of false positives is small.

A new method of choosing feature vectors increases the separation between the target and non-target acceptance was the removal of a small amount of vectors that are considered common (by our definition) across multiple speakers. Using a fairly conservative methodology for common removal in Experiment 2, it can be seen that the separation did increase for our trials, by an average of almost 4 percent.

Future work will involve biasing our network towards false negatives to minimize the false positive rate. Recent work [15, 16] using more cepstral features and gaussian mixture models have demonstrated speaker identification rates of greater than 98% on highly idealized TIMIT data, telephone quality data results are much lower. The gaussian model is complex and would not lend itself to easy storage on an ID card, as our approach would. In [1] it was shown that a fixed architecture multilayer perceptron could be trained to recognize 5 speakers at greater than 95% accuracy when tested upon 5 sentences from each of the speakers. Compression [12] was used in this work and the data set was the high quality DARPA TIMIT data [6] with downsampling. The results are not directly comparable to ours as the recordings were different and multiple speakers were trained to be identified, where we tried to train nets to just identify one speaker. They do not have any non-target speakers included in either training or testing. In other experiments in [1] with other neural and non-neural models, they showed greater than 90% identification rates and low false positive rates when given non-speakers not in the train set. These results suggest that our results may generalize to larger test sets.

Acknowledgments

We would like to thank Kul Ohri who wrote the software to obtain cepstrum coefficients. We would like to also thank the numerous people who recorded sentences for the many trials we ran. This research was partially supported by Seaway Technologies.

References

[1] K. R. Farrell, R. J. Mammone, and K. T. Assaleh, "Speaker Recognition Using Neural Networks and Conventional Classifiers," *IEEE transactions on Speech and Audio Processing*, Vol. 2, no 1, part II, pp. 194-205, January 1994.

[2] S. E. Fahlman, "Faster-Learning Variations on Back-Propagation: An Empirical Study," *Proceedings of 1988 Connectionist Models Summer School*, Morgan Kaufmann

[3] S. E. Fahlman, and C. Lebriere, "The Cascade-Correlation Learning Architecture," *Advances in Neural Information Processing Systems 2*, D. S. Touretzky (ed.), Morgan Kaufmann, 1990

[4] Ricci, A. (1994), "SoundEffects Speech Software", Shareware, *Torino, Italy* , available at http://wuarchive.wustl.edu/systems/mac/umich.edu/sound/soundutil/.

[5] L. Norskog, and numerous other contributors, "SOX - Sound Exchange / Universal Sound Sample Translator" *Freely Redistributable Software w/ Copyright* , L. Norskog and others, July 1991.

[6] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallet, and N. L. Dahlgren, "DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CD-ROM" *NIST Speech Disk 1-1.1, October 1990*, U.S. Department Of Commerce and National Institute Of Standards and Technology (CR-ROM: October 1990, Documentation: February 1993)

[7] A. E. Rosenberg, "Automatic Speaker Verification: A Review," *Proceedings of the IEEE, Special Issue: Man-Machine Communication by Voice*, Vol. 64, no 4, pp. 475-487, April 1976

[8] A. J. Compton, "Effects of Filtering and Vocal Duration upon the Identification of Speakers, Aurally," *Speech Intelligibility and Speaker Recognition* , M. E. Hawley (ed.), Dowden, Hutchinson, and Ross, Inc., 1977

[9] L. K. Hansen, and M. W. Pedersen, "Controlled Growth of Cascade Correlation Nets," *Proc. of the ICANN 1994* , Vol. 1, Parts 1 and 2, pp. 797-800 E. Marinaro and P. G. Morasso (ed.), Springer-Verlag,

[10] G. P. Drago, and S. Ridella, "Cascade Correlation Convergence Theorem," *Proc. of the ICANN 1994* , Vol. 1, Parts 1 and 2, pp. 573-576 E. Marinaro and P. G. Morasso (ed.), Springer-Verlag,

[11] P.A. Lynn (1989), Introductory digital signal processing with computer applications, Wiley, N.Y.

[12] A. Gersho and R. M. Gray (1992), Vector quantization and signal compression, Boston : Kluwer Academic Publishers.

[13] S. Furui, M. Sondhi (Eds.) (1992), Advances in speech signal processing, New York : M. Dekker.

[14] R. P. Ramachandran, M. S. Zilovic, and R. J. Mammone, "A Comparative Study Of Robust Linear Predictive Analysis Methods With Applications To Speaker Identification," *IEEE Transactions On Speech And Audio Processing*, Vol. 3, no 2, pp. 117-125, March 1995

[15] D. A. Reynolds, and R. C. Rose, "Robust Test-Independent Speaker Identification Using Gaussian Mixture Speaker Models," *IEEE Transactions On Speech And Audio Processing*, Vol. 3, no 1, pp. 72-83, January 1995

[16] D. A. Reynolds, "Large Population Speaker Identification Using Clean And Telephone Speech," *IEEE Signal Processing Letters*, Vol. 2, no 3, pp. 46-48, March 1995

[17] M. Sharkey and L.O. Hall (1996), Speaker Verification by Neural Networks, Technical Report ISL-02-96, Dept. of Computer Science and Eng., Univ. of South Florida, Tampa, Fl.

A Model for Detecting Singular Points of a Fingerprint

D.C. Douglas Hung and Ching-Yu Huang

Computer Vision Lab
Department of Computer and Information Science
New Jersey Institute of Technology
Newark, NJ 07102
austin@optic.njit.edu

Abstract

The direction image of a fingerprint is first quantized into n levels. An inscribed circle is used to measure the local azimuthal distance between two adjacent *fault-lines*. Mathematically speaking, ridge flow of a fingerprint can be described as a set of concentric curves and, theoretically, the center of these curves is the (called a singular point point) vanishing point of the generated fault-lines. A singular point is the location where a minimum azimuthal distance is detected, and along the *major region*.

1. Introduction

Criminologically, a fingerprint is used as an identification due to its well-known unique, immutable, and permanent nature. Since a fingerprint is marked wherever a fingertip contacted, the authority can trace criminals from the latent prints left in the criminal scene.

Fingerprint can be described as a ridge flow of various type containing a set of local ridge disturbances called minutiae. Minutiae are essential geometric skin-mark of a fingerprint. However, they are useful only when known reference is represented. As the center of a concentric ridge flow, a singular point can be used as a salient reference in coordinating detected minutiae.

Rosin [3] used maximum gradient region to detect the singular paths. Rao and Balck [2] and Sherlock and Monro [4] developed a simple model to describe the topological behavior of ridge flow around singular points. Since their direction measurements were based on a squared grid spaced 16 pixels apart, their approach can only semi-part singular points in a resolution of 16×16. Srinivasan and Murthy, [5] on the other hand, use a six steps approach to identify singular points in a by using directional histograms in the neighborhood of probable singular points points. This approach also uses block direction, however, is less robust than the previous method. Our paper proposes a different approach than the aforementioned methods to get more precise position of sinuglar points.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/444 ©1996 FLAIRS

2. Directional Image

A fingerprint can be described as a set of concentric ridge flows. As Henry [1] had stated, ridge flows are direction-oriented characteristic. Let n be the number of directions used in representing directions of a ridge flow.

Let $D = \{ D_1, ..., D_n \}$ be a set of integer directions which are represented directional orientation ranging $[0, \pi]$ (shown as Fig. 1). To get any pixel P_{ij}'s direction is by computing all 8-connected pixels which are the same type (ridge or valley) as P_{ij} within $(2W+1)\times(2W+1)$ window size area. Then the maximum one in the direction histogram is assigned to the pixel P_{ij} as its direction. A discontinuity in digitized and quantized direction field is called a *region*. Then a *fault-line* is defined as the border line of two different direction regions.

Since the orientation angle along regional ridge flow of a fingerprint is gradually changed, the digitized n-values direction image can be divided into a number of cone-chopped regions, each of which is colored by the same direction number. Typically, several regions are existed in a fingerprint direction image. From the observation, there is a region crossing the whole image. This region is denoted as *major region* and its direction is called the *major direction*. Since the first derivative of the direction increases as the

position closer to the center of the concentric curves, each direction region is shaped as a cone with the center as the apex, and these fault-lines are converged toward a common center, called a *core*, and diverged from a common center, called a *delta*.

3. A Model of Inscribed Circle

Let $S_{ij}(a,b)$ be a smoothing function which the number of major direction is above b % within $(2a+1)\times(2a+1)$ area for pixel P_{ij}. Since there is possible local disturbance, it is necessary to smooth the original direction image.

Let ω_k denote the direction region colored by the value k, and L_k denote the fault line between ω_k and ω_{k+1}. For a pixel $P_{ij} \in \omega_k$ be an arbitrarily selected points in the direction image. Suppose that d_1 and d_2 be the shortest distances between P_{ij} and two nearest fault lines L_k and L_{k+1} with $d_2 > d_1$, and the intersection points are ${}_1E_{ij}$ and ${}_2E_{ij}$, respectively.

Let O_{ij} be the center of a virtual inscribed circle enclosing P_{ij} and ρ_{ij} is the radius of this virtual circle. As shown in Fig. 2, we have

$$\delta_1 = \rho_{ij} - d_1, \text{and } \delta_2 = d_2 - \rho_{ij}. \quad (1)$$

Let θ_{ij} be the smaller contained-angle between $\vec{d_1}$ and $\vec{d_2}$. By geometry, we have $\delta_1 = \delta_2\cos\theta$. This leads to the following

$$\delta_1 = \frac{d_2 - d_1}{1 + \cos\theta} \quad (2)$$

and

$$\rho_{ij} = \frac{d_2 + d_1\cos\theta}{1 + \cos\theta} \quad (3)$$

Since fault-lines are supposedly intersected at a singular point, it implies that if we draw a sequence of inscribed circles, their radius decrease as their centers close to a singular point. That is a local maximum area on $1/\rho_{ij}$ curve where is the position of a singular point.

4. Majority Core Points

Theoretically, a singular point, must be located at a position which is local maximum areas of $1/\rho_{ij}$. However, in a real fingerprint a local maximum area position (i,j) may not be a singular point because the inscribed circles may be converged before reaching major region by digitizing error. In Fig. 3, P_1, P_2, ..., P_5 are local maximum areas, but not all of them are singular points.

When the local maximum area is obtained, two rules are played for deciding it is a singular point or not:

- A fingerprint must have the major region and major direction.
- A local maximum area must have at least 3 different directions.

If the intersection of fault-lines satisfies these two rules, we define it is a singular point candidate.

Since the major region is combined by D_i and D_{i+n}, the maximum change is found between D_i and D_{i+n-1}. Recalling the definition of singular point, it must be located at the maximum direction change, so it is noted that the maximum direction change is happen at along the boundary of the major region. In addition, a fault-line is formed by two different direction regions, so a local maximum area is consisted of at least three different direction regions, in other words, there must be at least 3 different directions in a 2×2 window in local maximum area, shown as Fig. 3.

Even with the digitizing error, the singular point position only should be shifted few pixels called Φ tolerance. If the candidates are closed in Φ distance each other, we consider they are at the same group and get the averaging position of them as estimated singular point position. In Fig. 3, the estimated singular point points are computing as: $C_1 = P_2$ which P_2 is a group itself and C_2 is the average position of group P_3 and P_4.

5. Experiment

Since in real latent fingerprint images, there are lot of noise, such as ambiguous isolated points, bifurcations, ridge endings, etc. By our experiments, it is better using few directions. Because use more directions, some fault-lines will be converged before reach major region. So, here we use 3 directions.

Given a double loop fingerprint image, as shown in Fig. 4. (a) is the original image. And its direction image in (b) which is computed by applying the previous algorithm with using direction number $n = 3$, window size $W = 6$, and smoothing by $S(1,0)$ and $S(W,50)$. Finally, we apply the previous section rules to get the result in (c) with tolerance $\Phi = 6$, with the singular points marked as "X" on center. And the positions of singular points are show on (d).

We rotate the double loop fingerprint image Fig. 4(a) left 90 degree as Fig. 5(a). After re-computing its direciotn, the direction image is (b) with $n = 3$, $W = 6$, $S(1,0)$ and $S(W,50)$. The singular points and positions are shown on (c) and (d), respectively.

Table 1 explains the comparison of singular points position between Fig. 5 and Fig. 6, after the normalization. The normalization function is

to find the averaged position and rotate the longest distance to x-axis. We can find the two data set are close in one pixel.

Give more experiments shown as Fig. 6, Fig. 7 and Fig. 8, a single loop , a ellipse , a tent arch fingerprints, respectively. Fig. 6(a) is the detected singular points of the single loop image, and (b) is its direction image Fig. 7(a) is the detected singular points of the ellipse model and (b) is its direction image Fig. 8(a) is the detected singular points of the tent arch model and (b) is its direction image

6. Conclusion

In this paper, we represent fingerprint by direction image and fault lines. The mechanism of this paper can precisely and easily get the singular points position with simple geometry concept. The singular points position is helpful fo our further research, such as classifying and matching.

References

[1] E. R. Henry, "Classification and uses of fingerprint", Routledge, London, (1990)

[2] C. V. Kameshwara Rao, K. Balck, "Finding the core point in a Fingerprint" *IEEE trans. Comput.*, vo. C- 27, pp. 77 - 81, January, (1978)

[3] Paul L. Rosin, Alan C. F. Colchester and David J. Hawkes, "Early Image Representation using Regions Defined by Maximum Gradient Paths Between Singular Points" *Pattern Recognition*, vol. 25, no. 7, pp. 695 - 711, (1992)

[4] B.G. Sherlock and D.M. Monro, "A Model for Interpreting Fingerprint Topology", *Pattern Recognition*, vol. 26, no. 7, pp. 1047-1055 (1993)

[5] V.S. Spinvasan and N.N. Murthy, "Detection of singular points in fingerprint images", *Pattern Recognition*, vol. 25, no. 2, pp. 139-153 (1992)

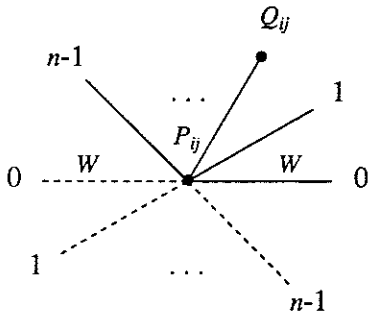


Fig. 1 Orientation quantification with n distinct levels and direction P_{ij} to Q_{ij}.

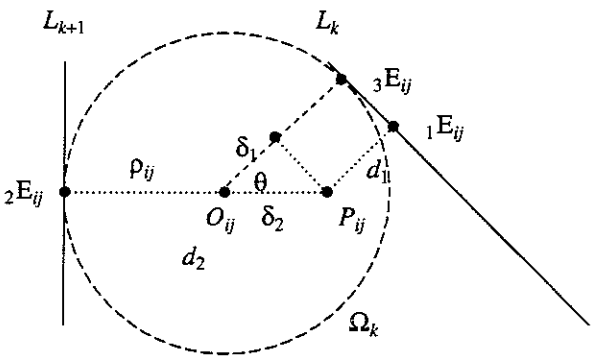


Fig. 2 Evaluation of the radius of an inscribed circle centered at an arbitrary point P_{ij}.

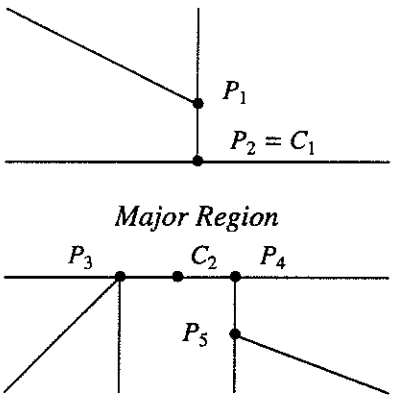
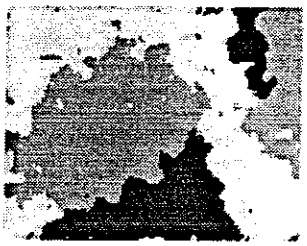
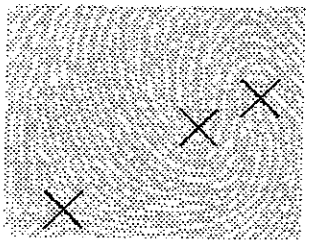


Fig. 3 The fault lines of double loop model.

(a)



(b)



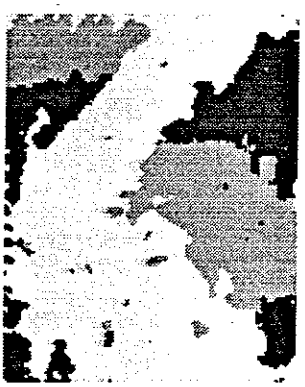
(c)

(141, 51)
(107, 68)
(33, 115)

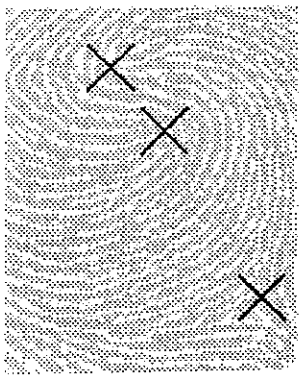
(d)

Fig. 4(a) the original double loop fingerprint
(b) The Direction image, with $n = 3$, $W = 6$
(c) The singular points with $\Phi = 6$
(d) The singular points position.

(a)



(b)



(c)

(47, 27)
(71, 56)
(115, 132)

(d)

Fig. 5(a) the rotated double loop fingerprint
(b) The Direction image, with $n = 3$, $W = 6$
(c) The singular points with $\Phi = 6$
(d) The singular points position.

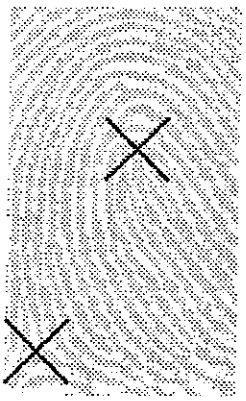
Original position	After normalized
(141,51)	(-54.47,-1.59)
(107,68)	(-16.59,1.59)
(33,115)	(71.06,0)

(a) Normalization of Fig. 4.

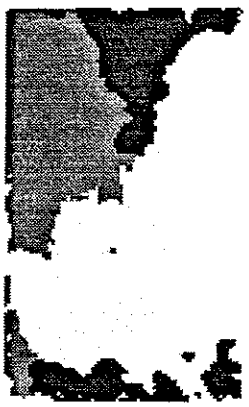
Original position	After normalized
(47,27)	(-54.12,-2.57)
(71,56)	(-16.83,2.57)
(115,132)	(70.95,0)

(b) Normalization of Fig. 5.

Table 1 The comparison of singular points position between (a) Fig. 4 and (b) Fig. 5,

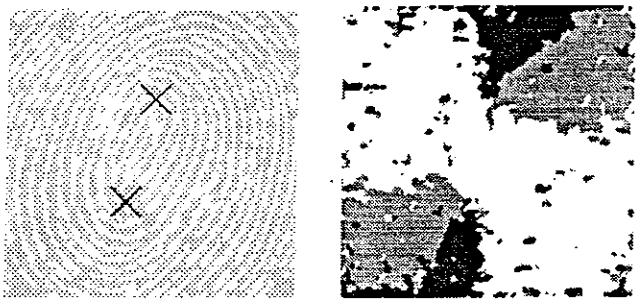


(a)



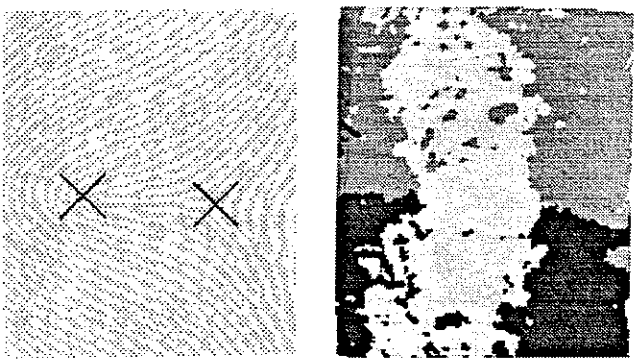
(b)

Fig. 6(a) The single loop fingerprint with detected singular points.
(b) The Direction image, with $n = 3$, $W = 6$



(a) (b)

Fig. 7(a) The ellipse fingerprint with
detected singular points.
(b) The Direction image, with $n = 3$, $W = 6$.



(a) (b)

Fig. 8(a) The tent arch fingerprint with
detected singular points.
(b) The Direction image, with $n = 3$, $W = 6$.

POWERFUL SEARCH HEURISTICS BASED ON WEIGHTED SYMBOLS, LEVEL AND FEATURES

Matthias Fuchs
Fachbereich Informatik, Universität Kaiserslautern
Postfach 3049, 67653 Kaiserslautern
Germany
E-mail: fuchs@informatik.uni-kl.de

Abstract

Problems stemming from the study of logic calculi in connection with an inference rule called "condensed detachment" are widely acknowledged as prominent test sets for automated deduction systems and their search-guiding heuristics. We present here a heuristic based on a weighted sum of function symbols and variables which takes into account the level of deduced facts. We show that this heuristic is already very powerful on the basis of numerous experiments conducted with an experimental program called 'CoDE'. By adding the ability to consider features of deduced facts further improvements can be achieved. We compare our results with the results the creators of OTTER obtained with this renowned theorem prover and this way substantiate our achievements.

1 INTRODUCTION

Problems in the area of (automated) deduction usually confront automated deduction systems with infinite search spaces. In order to cope with these tremendous difficulties it is imperative that an automated deduction system is equipped with search-guiding heuristics that help to prune the search spaces so as to reduce the deduction of irrelevant facts not contributing to the proof eventually found.

In this paper we shall present a heuristic which is based on a weighted sum of function symbols and variables and also takes into account the level of deduced facts. It is further upgraded by considering features which describe (syntactic) properties of facts with the help of integer numbers. For our experimentation we have chosen problems that originate from the study of logic calculi with an inference rule called *condensed detachment* (CD for short; see [9] and [3] for original work). The study of logic calculi with CD is widely acknowledged as a challenging field for automated deduction (cf. [6], [10], [5], [7], [11]), providing numerous problems ranging from almost trivial to extremely difficult. Despite the simplicity of CD (in terms of an unproblematic application of this inference rule) the problems can have at least the same degree of difficulty known from other fields of automated deduction. Therefore, the heuristic proposed in this paper should be of general interest.

Although the heuristic to be presented is rather simple in its design, it is nevertheless remarkably powerful. We substantiate this claim by comparing the results obtained with an experimental program called 'CoDE' (employing this heuristic) with the results the creators of OTTER ([4]) attained with this renowned theorem prover (cp. [5]).

The paper is organized as follows. First, section 2 presents the basics of CD and of our experimental program CoDE. Section 3 introduces the heuristic ϖ based on a weighted sum of function symbols and variables which makes use of the level of deduced facts, while section 4 reports on experimental results obtained with ϖ. Section 5 deals with upgrading ϖ by employing features. Experimental results in this context are given in section 6, before a final discussion in section 7 concludes this paper.

2 CONDENSED DETACHMENT

In this section we present the study of logic calculi as a research area that can be tackled with automated deduction systems. (See [9] and [3], here in particular pp. 250–277, for motivation and a detailed theoretical background.) Furthermore, we also introduce such an automated deduction system named 'CoDE'.

The inference rule 'condensed detachment' (CD) is the central part of the different logic calculi we are going to investigate. This inference rule manipulates first-order terms which we shall also call *facts*. The set of terms (facts) $Term(\mathcal{F}, \mathcal{V})$ is defined as usual, involving a finite set $\mathcal{F}$ of function symbols and an enumerable set $\mathcal{V}$ of variables.

CD (in its basic form) is defined for a distinguished binary function symbol $f \in \mathcal{F}$, allowing to deduce the fact $\sigma(t)$ from two given facts $f(s,t)$ and s', where σ is the most general unifier of s and s'. (CD can consequently be seen as a generalized version of the well-known modus ponens.) $f(s,t)$ and s' are the *immediate ancestors* of $\sigma(t)$. The (proof) problems consist of deducing a certain given fact λ_G (the *goal*) from an also given set Ax of facts (the *axioms*) by applying CD.

A very common principle used to solve such proof problems algorithmically is employed by most deduction systems based on, e.g., resolution. It also constitutes the core

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/449

of CoDE. Essentially, CoDE maintains a set F^P of so-called *potential facts* from which it selects and removes one fact λ at a time. λ is put into the set F^A of *activated facts* or discarded it if it is subsumed (denoted by '$\lhd$') by an already existing activated fact (*forward subsumption*). Activated facts $\lambda \in F^A$ are, unlike potential facts, allowed to produce new facts via CD, which then are put into F^P. Initially, $F^A = \emptyset$ and $F^P = Ax$. The indeterministic selection or *activation* step is realized by heuristic means. To this end, a selection heuristic $\mathcal{H}$ associates a natural number $\mathcal{H}(\lambda) \in \mathbb{N}$ with each $\lambda \in F^P$, which is referred to as "weighting λ with $\mathcal{H}(\lambda)$". Subsequently, that $\lambda \in F^P$ with the smallest weight $\mathcal{H}(\lambda)$ is selected. Ties are broken according to the FIFO-strategy (*"first in–first out"*).

The search-guiding heuristic $\mathcal{H}$ is crucial for the efficiency of the proof procedure just described. In the sequel, we shall present such a heuristic and its improvements, demonstrating its usefulness in the light of experimental results.

3 THE BASIC HEURISTIC ϖ

We present here the foundations of our search-guiding heuristic used by CoDE. Recall that a search-guiding heuristic associates a weight with each fact and selects the one with the smallest weight. Since the majority of proof tasks consists in proving facts that have a comparatively small number of function symbols, our basic heuristic is designed to focus on "small" facts. (Heuristics centered on this observation are very common. See, for instance, [2] or [5].) Instead of simply counting the number of function symbols and variables, we compute a weighted sum.

Definition 3.1 (Weighted Sum) *Let λ be a fact (a term). The* weighted sum $w(\lambda)$ *is defined by*

$$w(\lambda) = \begin{cases} 1, & \text{if } \lambda \in \mathcal{V} \\ 2 + \sum_{i=1}^{n} w(t_i), & \text{if } \lambda \equiv f(t_1, \ldots, t_n) \end{cases}$$

By giving a smaller weight to variables we cause a bias towards the selection of facts with more variables rather than function symbols. Such facts tend to be more general and hence more useful during the deduction process. The downside, however, is that more general facts allow to deduce more facts than less general ones, and hence may complicate the search by bloating up the search space. But, in connection with CD, the advantages appear to outweigh possible disadvantages (cp. section 4).

This simple heuristic w has quite understandably apparent limitations. But using it we discovered that the *level* of the proofs found was rather low. The level of a fact is the maximum of the levels of its immediate ancestors plus 1, or 0 if it is an axiom. The level of a proof is the maximum of the levels of the facts constituting this proof. Let $\delta(\lambda)$ denote the level of a fact λ. Having observed the prevalence of "low level proofs", the next obvious step to improve the basic heuristic was to incorporate the level of the fact λ to be weighted, so that "deeper" facts would receive a (moderate) penalty. We chose a weighted sum of the level and the result of w yielding the *basic heuristic* ϖ. In order to speed up the selection of a fact λ which subsumes the current goal λ_G and to conclude the proof faster this way, the minimal weight 0 is associated with such a λ.

Definition 3.2 (Basic Heuristic) *Let $c_\delta, c_w \in \mathbb{N}$, λ the fact to be weighted, and λ_G the goal. The* basic heuristic ϖ *weighting λ with $\varpi(\lambda)$ is defined by*

$$\varpi(\lambda) = \begin{cases} 0, & \text{if } \lambda \lhd \lambda_G \\ c_\delta \cdot \delta(\lambda) + c_w \cdot w(\lambda), & \text{otherwise} \end{cases}$$

The design of ϖ allows us to examine borderline cases, namely w alone ($c_\delta = 0$, $c_w \geq 1$), breadth-first search ($c_\delta \geq 1$, $c_w = 0$) or the FIFO-strategy ($c_\delta = c_w = 0$). The additional effort on account of the subsumption test clearly pays off as experiments have shown.

The following section 4 presents some of the results we obtained by applying ϖ with various configurations of c_δ and c_w. It also discusses a "by-product" we encountered when running tests with ϖ, namely finding *shorter* proofs *faster*.

4 EXPERIMENTS WITH ϖ

First, we give an excerpt of our experimental results in table 1. (See [1] for more information.) The first column lists the names of the proof problems. All problems considered in this paper are taken from [5]. The name of a problem is composed of the abbreviation of the calculus it belongs to and of the continuous numbering used in [5]. (These problems also appear in the TPTP library ([8]) as 'LCL problems', and the names used here correspond to the names in the slot 'Names' of TPTP problem files.) The last column displays the *best* results of OTTER (also taken from [5]). Note that OTTER uses a variety of heuristics, some of which rigorously eliminate facts that have certain syntactic properties, and hence are unfair. The head of each remaining column shows the ratio '$c_\delta : c_w$' employed by CoDE's basic heuristic ϖ. The entries of the tables list the (approximate) run times in seconds (CPU time) obtained on a SPARCstation ELC. The run times of OTTER were obtained on a SPARCstation 1+ which is a comparable machine. An entry '—' (CoDE's columns only) signifies that no proof could be found when restricting the memory to 45 MB. Naturally, OTTER has to control its memory usage in order to be able to cope with a limited memory (12 MB in OTTER's case). CoDE also disposes of ways to control its memory usage which, just like OTTER's, can cause incompleteness. (Basically, memory usage is controlled by discarding certain facts respectively clauses.) But we did not have CoDE use them here, since the results attained without them are satisfactory.

It must be emphasized that CoDE is an experimental C program whose core was developed in a couple of weeks as opposed to the well-renowned OTTER which has been improved over years. CoDE does not use sophisticated indexing techniques. These are crucial for efficient (forward) subsumption which is exhaustively needed in connection with CD. Consequently, faster run times of CoDE can only stem from heuristics that allow for a more efficient search.

Table 1: Results of the Basic Heuristic ϖ

Name	0 : 1	1 : 3	1 : 2	2 : 3	1 : 1	4 : 3	3 : 2	2 : 1	3 : 1	4 : 1	OTTER
CN-06	—	—	—	—	—	—	—	90s	24s	11s	1467s
CN-19	84s	—	—	—	—	—	—	—	—	—	423s
CN-21	91s	—	—	—	—	—	—	—	—	—	447s
CN-25	—	—	—	—	—	—	33s	18s	15s	11s	89s
CN-28	—	—	—	—	—	73s	50s	15s	11s	36s	89s
CN-29	197s	47s	40s	35s	82s	—	—	—	—	—	257s
CN-31	—	—	—	—	94s	25s	16s	10s	17s	—	480s
CN-32	—	89s	45s	46s	86s	—	—	—	—	—	511s
CN-33	208s	58s	15s	46s	3.4s	1.4s	2.8s	2.9s	1s	1.1s	224s
MV-59	—	—	—	130s	40s	51s	82s	12s	75s	68s	1468s

Problems CN-06 and CN-25 are perfect examples for the benefits of taking into account the level of a fact. There, a proof can only be found when $c_\delta : c_w$ is at least $2 : 1$ respectively $3 : 2$. Moreover, the higher the ratio, the faster a proof is found. For other problems, the increase of the ratio has to be more moderate (e.g., problems CN-29 and CN-32). Nonetheless table 1 shows that considering the level is profitable (see [1] for more experimental evidence). The only two remarkable exceptions are problems CN-19 and CN-21. But they reveal that computing a *weighted* sum of function symbols and variables is favorable here. (OTTER's heuristic that simply counts symbols needs more than 1000 seconds to find a proof of either problem.)

Apart from finding proofs faster, raising the ratio $c_\delta : c_w$ can also allow *shorter* proofs to be found, as table 2 reveals. The length of a proof is measured as in [10] in terms of the number of facts that have to be deduced in order to accomplish the proof. (Axioms hence do not contribute to proof length.) [10] already addresses the problem of finding shorter proofs than the ones found so far. This issue is important if the proofs obtained are to be presented to a human reader who quite naturally prefers short, less complex proofs. The search-guiding heuristics, however, (in general) do not support this desire. So, in [10] several methods are examined which force OTTER to seek (shorter) alternative proofs. But such a forced quest for shorter proofs often results in much longer run times.

Examining table 2, we discover that except for the problems EC-79 and RG-102 all other problems exhibit a "regular" behavior. By increasing the ratio $c_\delta : c_w$ both run time and proof length continuously decrease until a certain "lower bound" is reached. As problems EC-79 and RG-102 show, proof length and run time may show some irregular behavior. Moreover, fastest and shortest proofs are not as perfectly correlated as they are for the other problems. But, as a common result, table 2 demonstrates significant improvements both with respect to run time and proof length compared to the case where the level is not taken into account at all (ratio $0 : 1$). Please note that the ratio $9 : 1$ is not a "magic" boundary. Depending on the problem at hand, increasing the ratio even further can still produce shorter proofs.

We would like to emphasize the results obtained in connection with problem EC-69 which corresponds to *Theorem 5* in [10] on pages 228 and 229. The length of the first proof presented there is 31. A second shorter proof has length 18, but required approximately seven times as much CPU time. By raising the ratio $c_\delta : c_w$ to $9 : 1$, our heuristic produced a proof of length 6. It succeeded in finding it roughly 37 times *faster* than it found the "longest" proof (ratio $0 : 1$), which is, by the way, identical to the shorter proof documented in [10]. The last column of table 2 lists OTTER's best results[1] as a point of reference to underline the quality of ϖ's achievements.

The simplicity and experimentally documented performance argue in favor of ϖ. But despite its success we want to point out that the use of ϖ with increasing ratio $c_\delta : c_w$ does not guarantee to find shorter proofs. Nonetheless, our heuristic use of the level *is* profitable whenever shorter proofs depend on "larger" facts occurring at a low level, so that a lesser penalty due to a lower level can compensate for a higher weight ensuing from w.

5 USING FEATURES

[5] reports on OTTER using various heuristics, some of which eliminate facts that have certain syntactic properties. 'Having a syntactic property $\mathcal{X}$' can be expressed with the help of *features*. A feature f maps (in our case) a fact λ into the set of integers $\mathbb{Z}$, i.e., $f(\lambda) \in \mathbb{Z}$ is the corresponding *feature value*. By choosing an appropriate feature f and a subset $\emptyset \neq V_f \subseteq \mathbb{Z}$, '$f(\lambda) \in V_f$' can be made equivalent to 'λ has the syntactic property $\mathcal{X}$'. V_f may be viewed as the set of *permissible* feature values (w.r.t. feature f). In contrast to rigorously eliminating a fact λ if $f(\lambda) \notin V_f$, we prefer the more moderate alternative to impose a weight penalty $w_F(\lambda)$ on λ. (Naturally, this may in practice amount to elimination if the weight penalty is big enough.) Several features can be handled in the following way: Let $f_1, \ldots, f_k$ be a set of features and $V_1, \ldots, V_k$ the associated sets of permissible feature values. Using the *minimal feature value difference*

$$\Delta_i(\lambda) = \min(\{|f_i(\lambda) - v| \mid v \in V_i\}), \quad 1 \leq i \leq k$$

[1] [5] provides information on run time only.

Table 2: Run Time and Proof Length in Dependence of Ratio $c_\delta : c_w$

Name	0 : 1	1 : 1	2 : 1	3 : 1	4 : 1	5 : 1	6 : 1	7 : 1	8 : 1	9 : 1	OTTER
EC-69	111s	74s	66s	16s	11s	11s	6s	3s	3s	3s	244s
	18	18	6	6	6	6	6	6	6	6	
EC-79	100s	41s	92s	51s	6s	10s	31s	73s	30s	20s	188s
	39	34	45	42	25	28	25	25	34	24	
R-86	3s	1s	0.7s	0.4s	0.3s	0.3s	0.2s	0.2s	0.2s	0.1s	5s
	30	12	7	7	7	7	7	7	7	7	
R-88	6s	2s	1.5s	0.6s	0.3s	0.3s	0.3s	0.3s	0.3s	0.3s	21s
	12	12	6	6	6	6	6	6	6	6	
LG-89	31s	15s	3s	5s	2s	1.2s	1s	0.4s	0.5s	0.4s	115s
	22	18	17	4	4	4	4	4	4	4	
LG-90	107s	79s	4s	1s	1s	1s	0.4s	0.3s	0.3s	0.3s	2s
	8	6	6	6	6	6	6	6	6	6	
LG-91	25s	25s	9s	3s	2s	2s	0.6s	0.6s	0.6s	0.6s	9s
	12	12	4	4	4	4	4	4	4	4	
RG-102	44s	13s	8s	6s	7s	8s	7s	8s	8s	13s	130s
	31	22	24	21	21	23	19	19	21	21	

straight forwardly leads to the following definition of the weight penalty w_F:

$$w_F(\lambda) = \sum_{i=1}^{k} c_i \cdot \Delta_i(\lambda)\,, \quad c_i \in \mathbb{N}\,.$$

The use of $\Delta_i(\lambda)$ instead of the Boolean value resulting from $f_i(\lambda) \in V_i$ or $f_i(\lambda) \notin V_i$ is more general, containing the Boolean special case (choose c_i big enough). By combining the basic heuristic ϖ and w_F in the "obvious" way we obtain the *feature heuristic* ϖ_F (which also forces the selection of a fact subsuming the goal λ_G):

$$\varpi_F(\lambda) = \begin{cases} 0, & \text{if } \lambda \lhd \lambda_G \\ \varpi(\lambda) + w_F(\lambda), & \text{otherwise} \end{cases}$$

Please note that both the coefficients c_i and the sets V_i may be determined based on previous proof experience. Thus, this approach to using features also constitutes a framework for learning from past experience (see [1]).

The subsequent section deals with our experiments regarding ϖ_F.

6 EXPERIMENTS WITH ϖ_F

We conducted our experimental studies in the light of problems EC-69,...,EC-84 and RG-102,...,RG-112, omitting those problems that both OTTER and CODE can solve in less than 60 seconds regardless of parameter settings or strategies employed. CODE currently has 13 features $f_1, \ldots, f_{13}$ at its disposal, including w, δ, number of distinct variables, nesting of function symbols etc. (cf. [1]). In order to facilitate a transparent and sensible experimental evaluation we confine ourselves to feature f_{11} which counts the number of occurrences of the subterm $e(\xi, \xi)$, where ξ is some variable. Using $V_{11} = \{0\}$ the weight penalty $w_F(\lambda)$ is c_{11} times the number of occurrences of a subterm $e(\xi, \xi)$ in λ. The usefulness of this sort of syntactic property has already been recognized in [5], where it is employed in form of a rigorous deletion strategy that *eliminates* all facts λ that have at least one occurrence of *an instance of* $e(\xi, \xi)$.

The experimental results reveal that our approach appears to be more adequate. We ran experiments with ϖ_F on the problems just mentioned, varying the ratio $c_\delta : c_w$ between 0 : 1 and 4 : 1 (as in table 1), while c_{11} took on values from $\{0, 5, 10, 15, 20\}$. (Note that in case $c_{11} = 0$ ϖ_F degenerates into ϖ.) Quite naturally there is no configuration c_δ, c_w, c_{11}—strictly speaking each representing an individual heuristic—that is "optimal" for all problems (under consideration). The existence of such a "universal heuristic" is anyway more than doubtful. (This is not solely our opinion, but was already pointed out in [5].)

In order to solve problems RG-105 and RG-106 we had to lift the rigid memory restriction and enable CODE's memory control mechanism which works as follows[2]: There is a memory *quota* of 30 MB and a memory *limit* of 45 MB. If the limit is exceeded, potential facts are deleted until the quota is reached, starting with those potential facts that have the highest weight.

Tables 3 and 4 show the results with respect to the problems from the EC-calculus and the RG-calculus, respectively. The first row (ϖ [*]) of table 3 gives the *best* results produced by ϖ—in other words ϖ_F with $c_{11} = 0$, hence not taking into account features—using those various ratios $c_\delta : c_w$. The second row of table 3 lists the run times obtained when employing ϖ_F with a fixed parameter setting, namely (ϖ_F [1]) $c_\delta : c_w = 0 : 1$ and $c_{11} = 20$. The third row (ϖ_F [*]) gives the best results produced by ϖ_F when running the experiments as described above. The last row shows the best results of OTTER. Table 4 is arranged like table 3, the fixed parameter setting here being $c_\delta : c_w = 1 : 3$ and $c_{11} = 20$.

[2]The entry '—' in the columns related to RG-105 and RG-106 therefore means that no proof could be found within a given period of time (one hour).

Table 3: Results of ϖ_F w.r.t. EC-calculus

	EC-69	EC-75	EC-77	EC-78	EC-79	EC-80	EC-81	EC-82	EC-83	EC-84
ϖ [*]	11s	26s	19s	24s	5.2s	—	16s	—	—	—
ϖ_F [1]	39s	13s	13s	69s	19s	42s	36s	41s	115s	40s
ϖ_F [*]	9.4s	4.8s	10s	24s	4.6s	18s	8.8s	41s	85s	40s
OTTER	244s	159s	85s	326s	151s	281s	245s	499s	886s	484s

Table 4: Results of ϖ_F w.r.t. RG-calculus

	RG-102	RG-103	RG-104	RG-105	RG-106	RG-108	RG-109	RG-110	RG-112
ϖ [*]	5s	—	—	—	—	—	1.5s	—	6.9s
ϖ_F [1]	5.4s	28s	13s	318s	2772s	27s	<1s	4.3s	<1s
ϖ_F [*]	3.4s	13s	4.4s	159s	1760s	18s	<1s	2.9s	<1s
OTTER	130s	104s	62s	809s	5634s	136s	40s	5837s	19s

Our "softer" use of features yields significantly better run times right down the line. In this context we want to point out the results obtained in connection with problems RG-102 and RG-110 (table 4). The goal of RG-102 contains an occurrence of $e(\xi,\xi)$. Therefore, there is no point in OTTER's using the deletion strategy, because the goal itself would be discarded. Since ϖ_F is not that strict, facts with occurrences of $e(\xi,\xi)$ may occur and actually do occur in a lot of the proofs eventually found. (In particular the fact $e(x,x)$ itself is a frequent member of proofs.) This property of ϖ_F is even more advantageous in connection with problem RG-110 where OTTER's deletion strategy also fails, because it eliminates all interesting paths according to the authors of [5]. Moreover, as also stated in [5], OTTER's deletion strategy is realized via demodulation which is a very expensive operation accounting for between one third and one half of the CPU time. Computing feature f_{11} also causes some overhead, but it is substantially smaller and clearly pays off.

7 DISCUSSION

We have presented a simple, powerful search-guiding heuristic that is based on a weighted sum of function symbols and variables. It also incorporates the level of derived facts. It is further upgraded by considering features (syntactic properties of facts). We have demonstrated its capabilities with the help of numerous experiments taken from the area of condensed detachment (CD), using an experimental program called CODE. The quality of CODE's achievements are underlined by a comparison with the renowned theorem prover OTTER. Although CODE is specialized in CD, OTTER is clearly superior in terms of inference rate due to sophisticated indexing techniques. Nonetheless CODE achieves (significantly) better run times (on even slightly slower machines). Therefore, CODE's search must be more efficient, i.e., far less irrelevant facts not contributing to the proof eventually found enter the search.

The simplicity of our heuristic both in terms of implementation aspects and computational effort during execution time should make it interesting for problems other than those from the area of CD. But we want to emphasize that CD does provide a batch of problems considered as a serious test environment for new ideas (cf. [10], [5]).

References

[1] **Fuchs, M.:** *Experiments in the Heuristic Use of Past Proof Experience*, SEKI-Report SR-95-10, University of Kaiserslautern, 1995 [http://www.uni-kl.de/AG-AvenhausMadlener/fuchs.html]

[2] **Huet, G.:** *Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems*, Journal of the ACM 27, No. 4, 1980, pp. 798–821

[3] **Łukasiewicz, J.:** *Selected Works*, L. Borkowski (ed.), North-Holland, 1970

[4] **McCune, W.:** *OTTER 3.0 Reference Manual and Guide*, Techn. Report ANL-94/6, Argonne Natl. Laboratory, 1994

[5] **McCune, W.; Wos, L.:** *Experiments in Automated Deduction with Condensed Detachment*, Proc. CADE-11, Saratoga Springs, NY, USA, 1992, LNAI 607, pp. 209–223

[6] **Peterson, G.J.:** *An automatic theorem prover for substitution and detachment systems*, Notre Dame Journal of Formal Logic, Vol. 19, Number 1, January 1976, pp. 119–122

[7] **Slaney, J.:** *SCOTT: A Model-Guided Theorem Prover*, Proc. IJCAI '93, Chambery, FRA, 1993, pp. 109–114

[8] **Sutcliffe, G.; Suttner, C.; Yemenis, T.:** *The TPTP Problem Library*, Proc. CADE-12, Nancy, FRA, 1994, LNAI 814, pp. 252–266

[9] **Tarski, A.:** *Logic, Semantics, Metamathematics*, Oxford University Press, 1956

[10] **Wos, L.:** *Meeting the Challenge of Fifty Years of Logic*, JAR 6, 1990, pp. 213-232

[11] **Wos, L.:** *Searching for Circles of Pure Proofs*, JAR 15, 1995, pp. 279–315

Referees for Teamwork

Jörg Denzinger, Dirk Fuchs
Department of Computer Science
University of Kaiserslautern
Postfach 3049, 67653 Kaiserslautern
E-mail: {denzinge,dfuchs}@informatik.uni-kl.de

Abstract

Teamwork is a method to distribute automated theorem proving and is mainly based on the competition and cooperation of different search control heuristics. Together with the usual control heuristics that try to anticipate whether an inference step and the resulting fact is good, teamwork also employs assessment heuristics that judge the impact a fact has had on the search after a while. For this assessment teamwork uses so-called referees that allow it to throw away facts that did not meet the expectations. In this paper we discuss several referee concepts and compare them by experiments.

1 Introduction

The main goal when developing search strategies for automated theorem provers is to do as few unnecessary inference steps as possible. Unfortunately, experience has shown us that automated theorem provers do many more unnecessary steps than steps that are needed for a proof. Since unnecessary steps tend to produce more unnecessary steps, a research goal with high priority is to eliminate or forget unnecessary inference steps.

Research concerning this problem is aimed in two directions. The first direction is to reduce the applicability of inference rules that generate new facts while developing other rules that simplify or remove existing facts. A good example of this direction are the works of Bachmair and Ganzinger on ordered superposition and on general redundancy criteria (see [BG90]). The second direction is to develop criteria to throw away facts without any justification by inference rules, what we call *forgetting* of facts. An example of results of this direction is the limitation of the size of generated facts that can be found, for example, in the Otter prover (see [Mc94]). Unfortunatly almost every usable criterion results in losing the completeness of the prover. Furthermore, most of the criteria developed so far are very crude, which means, for example, that they do not take the problem one wants to prove into account.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/454 ©1996 FLAIRS

With our teamwork method (see [AD93], [De95]) we developed a knowledge-based distribution method for automated theorem provers that concentrated on both search strategies to generate new facts and assessment heuristics to forget facts in order to avoid an explosion of the search space. The general idea of teamwork is to let several different search heuristics (so-called experts) work on the given problem in parallel and independently. Periodically referees judge the experts and select outstanding new facts from them. These facts are added to the set of facts of the best rated expert by a supervisor. Since the resulting search state is the new start state of all experts, all the experts' non-selected facts (without those of the best expert) are forgotten. The supervisor also uses the judgement by the referees to exchange experts for better suited ones, thus allowing for an adaptation of the system to the given problem.

After reports concerning special search heuristics ([DF94]) and the planning task of the supervisor when selecting the team for the next round ([DK94]) we will concentrate in this report on the referees. Obviously, the referees allow a synthesis of different search heuristics and strategies for forgetting facts. By selecting facts that have proven to be "good" (which has to be defined by the referees) and thus forgetting useless facts, a very sophisticated concept for deleting facts –that takes the problem to prove into account– is realized. This is enhanced by the competition of different search heuristics which is a result of the teamwork concept. In the following we will demonstrate several ways to design referees and compare the different referees by giving some examples.

2 Automated theorem proving and completion

Due to lack of space we can only give a very brief introduction to automated theorem proving, equational theorem proving and the completion method. For more details we refer to [CL73], [HR87] and [AD93].

Theorem proving means solving the following problem :
Given: A set A of axioms and a theorem T to prove.
Question: Is T a logical consequence of A ?
In equational proving A = $\{s_i = t_i \mid i=1,...,n\}$ is a set of (all-quantified) equations and T is an equation u = v, too.

All successful methods for automated theorem proving, if equality is involved, are based on two kinds of inference rules: generation rules and contraction rules. The generation inference rules add new facts to the data base. These facts are derived either from the axioms alone (as in the case of equational theorem proving by completion) or from both the axioms and the theorem T (as in the case of resolution and paramodulation). The contraction inference rules change or delete facts from the data base.

For equational reasoning, the area we applied teamwork to, the generation inference rule is the critical pair generation and the contraction inference rule is reduction (also known as demodulation). According to a so-called reduction ordering the equations are oriented to rules (if possible). A critical pair to two rules $l_1 \rightarrow r_1$ and $l_2 \rightarrow r_2$ is the equation $\sigma(l_1)[p \leftarrow \sigma(r_2)] = \sigma(r_1)$, where σ is the mgu of l_2 and the subterm of l_1 at position p (which mustn't be a variable) and where $\sigma(l_1)[p \leftarrow \sigma(r_2)]$ denotes the exchange of the subterm at position p in $\sigma(l_1)$ by $\sigma(r_2)$. Rules are also used to reduce terms. If there is a match μ from the left hand side of a rule $l \rightarrow r$ to a subterm of term t (at position p), then this subterm is replaced by $\mu(r)$ (written $t \Rightarrow t[p \leftarrow \mu(r)]$. An equation or rule is in normal form, if no reduction with any rule of the data base is possible.

From a theoretical point of view we need for an automated theorem prover, besides the inference rules, fairness criteria for the use of the inference rules. These criteria guarantee that each application of an inference rule that is enabled infinitely often will finally be executed. But, for challenging examples there are many possible inferences and a systematic application results both in enormous run times and the need of much (memory) space. It is not uncommon for a prover to require an agenda of 500,000 to 1,000,000 inference rule applications.

Therefore, implementations of automated theorem provers use strategies and heuristics to select the next inference rule to apply and the facts these rules should work on. A strategy guarantees theoretical completeness, whereas for heuristics it is possible that the prover will not find a proof even if there is one and enough time and space are provided. Heuristics are very important, because very often only the use of a heuristic allows finding a proof. The same can be said about simply forgetting facts that do not seem to be necessary. This allows a reduction of the search space but there is the danger that necessary information for finding a proof is deleted. Many theorem provers allow the user to choose between various strategies and heuristics. But two problems still remain. First, for a given problem there may be no appropriate strategy or heuristic implemented in the system. Second, even if there is a good one implemented, how does the user know which one is good? We tackled both problems with our teamwork method as we will demonstrate in the next section.

3 The teamwork method

The main intention of the teamwork method is to achieve cooperation between different strategies for automated theorem provers while using competition to direct the whole system in the most promissing direction. The main instruments for achieving these goals are techniques from the areas of distributed systems, multi-agent systems and knowledge-based systems together with control strategies, heuristics and techniques from automated deduction.

A system based on teamwork consists of four types of components: experts, specialists, referees and a supervisor. Experts and some specialists are those components working on solving a given problem. Some other specialists, the referees and the supervisor control the team and are responsible for achieving cooperation and competition.

Experts are automated theorem provers. They differ either in the inference rules they use or in the control strategies or heuristics they employ (mainly the control of the generation rules). All known general control strategies and heuristics can be used, but –since they work in a team– also fragments of strategies or even specialized ideas. All experts use a common representation of their search state. So, each of the experts can continue its work using the state of another expert. During the so-called working period all experts work independently on the given problem.

In the case of equational theorem proving by completion the experts use different selection methods for critical pairs. Besides standard criteria, like the number of symbols in the pair or using functional interpretations of the symbols over $\mathbb{N}$, we also have developed criteria that define similarities between a critical pair and the goal (see [DF94]). These criteria are examples of specialized heuristics that can only solve very few examples when working alone but are very useful in cooperation with other experts.

Specialists can either be used to assist in finding the solution to the given problem or to help the supervisor in its controlling task. Some specialists can even contribute to both tasks. One of these specialists, the so-called domain detection specialist, will be described together with the supervisor. In contrast to experts, specialists needn't use the common search state representation. Specialists also work independently of each other and the experts during the working periods.

Referees are responsible for comparing experts, specialists and their results and therefore represent the competition aspect of teamwork. After a working period each expert and each specialist is judged by a referee. A referee computes for its expert a *measure of progress* and it selects the best facts generated by the expert during the last working period. A referee for a specialist only selects good facts of the specialist, provided that the specialist has generated any facts. Other information computed by a specialist, as for example the domain of the problem to prove, is passed on to the supervisor unjudged. In section 4 we will describe in more detail possible criteria that can be used by referees to fulfill their tasks. The measure of progress of each expert is transmitted to the supervisor. Then, after the best expert is determined, the experts' and specialists' selected facts are reported to the supervisor.

The *supervisor* is responsible for strategic decisions in the team and therefore for the adaptation of the team to the given problem. In addition, the supervisor also achieves the cooperation of the team members by generating a new

start state for the next working period. In detail, the supervisor performs the following actions: First, it receives the measure of progress of each expert and the information from the specialists. Then the supervisor determines the best expert, i.e. the expert with the highest measure. After that it receives the selected facts of all other experts and the specialists and adds these facts to the search state of the best expert resulting in a new start state. Then the supervisor selects the members of the next team based on a long-term memory (consisting of information about domains of interest, good teams and good and bad expert combinations) and on a short-term memory (consisting of the information gathered by the specialists and the measure of progress of each expert for each working period so far). After transmitting the new start state to all members of the next team the supervisor also determines the length of the next working period and sends this information to the team members. Then the work of the supervisor is done and a new working period begins.

An important help for determining the new team members are the results of the *domain detection specialist*. A domain is described by a set of facts. With a domain are associated known consequences of the facts (which are the delivered to the referee of the specialist in case the domain is detected), a plan skeleton, hierarchical information about the domain and known good single experts and referees for this domain. The supervisor uses the plan skeleton as basis for its decision about the new team, but modifies this plan in reaction to the measures of the experts. For more about the planning task of the supervisor we refer to [DK94].

The teamwork method allows an efficient implementation with a very effective communication between processors. By representing experts/specialists, referees and supervisor as one process with different modi (supervisor modus always given to the processor running the best expert), the only interprocessor communication that is necessary are the reports of the referees to the supervisor and the transmission of the new start state. Since this transmission uses broadcast and allows for filtering the data on the side of the sender (eliminating unnecessary data) and ordering the data on the side of the receivers (typically using different orderings due to the different selection criteria), there is no big overhead due to communication.

Our experiments have shown that teamwork was indeed able to achieve cooperation between competiting strategies resulting in synergetic speed-ups and even in the ability to solve much more examples than all experts working alone (see [DF94]). Teamwork also provides a higher level of automatization than most other provers, which is due to the planning and adaptation ability of the supervisor (see [DK94]). Finally, we also have shown that teamwork can successfully be applied to other search problems, for example optimization problems. For the traveling salesman problem the same synergetic effects as for theorem proving can be achieved (see [De95]).

4 Referee concepts

Referees have two tasks, namely judging the whole work of an expert of a working period and selecting outstanding results of an expert. Therefore a referee consists of two functions, a *judge-expert* function and a *select-good-results* function. Select-good-results functions compute for each new fact a measure. Then the best n facts will be selected by the referee, provided their measure exceeds a given threshold. Since these measures can also be used to judge an expert (by, for example, using the mean value of all new facts as measure), in the following we will mainly concentrate on select-good-results functions and we only provide a short section about statistical criteria for judge-expert functions that have proven to be quite useful.

4.1 Judging an expert

In order to judge the complete work of an expert in a working period statistical criteria are the first obvious choice. And our experiments showed that these criteria are sufficient to allow successful proof runs with teamwork based systems. There is much statistical data that can be used in computing a measure of progress for an expert. In our experiments that will be described in section 5 we used only the judge-expert function *relational* that has two parameters, *red_factor* and *cp_factor*, and combines the number of critical pairs |CP|, the number of reductions made |Red|, the number of rules |R| and the number of equations |E|. The value of *relational* is computed as

$$red\_factor * |Red| - \frac{|CP| * cp\_factor}{|R|+|E|}.$$

The idea of *relational* is that many reductions obviously are a good sign, while many critical pairs mean more difficulties in selecting the right ones. But, if there are many rules and equations, then many critical pairs have to be expected while many critical pairs out of only a few rules and equations clearly indicate that experts with a better (i.e. lower) ratio should be preferred. We found it useful to start proof runs with referees using *relational* with a parameter combination, where *red_factor* >> *cp_factor* and then employing in later periods referees with growing *cp_factor*.

4.2 Select-good-results functions

Measuring facts is a task of both, experts and referees. On the one hand this makes finding criteria that can be used by select-good-results functions of referees easier, because the criteria of experts can be used as a starting point. On the other hand the question arises whether referees are different from experts at all and therefore possibly not necessary.

Experts try to measure the impact of a fact (or an inference) on the problem at hand. Since experts do not want to measure this impact by actually doing the inference leading to the fact with all consequences that are involved –this would be much too inefficient– they have to do a kind of guessing, which can be partially or even totally wrong.

On the other hand, referees measure facts after their impact on a search process has become known. They can observe what inferences resulted from the addition of a

fact. The impact of a new fact is also measured by the number of facts that could be thrown away, because they have become redundant. Since facts selected by a referee are added to the search state of the best expert, a further criterion that should be taken into account by a referee is the probability that a fact enhances this search state. This means that selected facts should produce inferences that will be given a high priority by the winning expert, without treating them in a special way. So, referees offer a new dimension and are indeed necessary.

In the following we will present three concepts for developing select-good-results functions that have different priorities concerning the goals given above.

Last(number_of_facts)

The select-good-results function *Last* is quite simple, it selects the last *number_of_facts* facts produced by the expert that were not redundant. The idea behind Last is that several experts that were not the best one may be members of the next team again. These experts should be able to continue their work without repeating most of the inferences they have done in the last working period. By including their last results into the new start state there is a high probability that they will concentrate on these results in future.

As we will see in section 5, this simple function can be used for several examples with some success. But for other examples the idea is just too simple.

Statistic(number_of_facts, crit_pair_penalty, reduction_bonus, threshold)

The aim of the select-good-results function *Statistic* is to measure the impact a fact has had on the search of an expert. The impact is observed by the number of critical pairs generated by a fact (weighted by *crit_pair_penalty*) and the number of reductions that have been made using the fact (weighted by *reduction_bonus*). Different reductions are counted differently when computing the number of reductions: Reductions of a goal count more (typically 100 times) than the reduction of a critical pair. Also the reduction of a rule or an equation counts more (typically 5 times). Statistic computes for each fact a weighted sum of the number of critical pairs and the number of reductions and selects the best *number_of_facts* facts (i.e. the facts with the highest sums), provided they are better rated as *threshold*.

As the names of the weights suggest, we intended to see many reductions as a bonus while the generation of many critical pairs should be given a penalty (i.e. *crit_pair_penalty* is negative). But it may be necessary to vary penalty and bonus during a proof run. At the beginning of a proof run the generation of many critical pairs should not be penalized, because a broad base of facts is needed. Therefore the number of reductions alone (i.e. *crit_pair_penalty*=0) or even a positive penalty for critical pairs can be used. Later on, using a positive value for weighting the number of critical pairs often becomes disastrous.

Nevertheless, Statistic has proven to be quite successful. For some examples an adjustment of the parameters allows for quite some improvements. Therefore there are some possibilities for the use of further knowledge for parameter adjustment. But there are standard parameters that result in a satisfactory performance.

Winner_Driven(number_of_facts, threshold)

The aim of *Winner_Driven* is to select results that have a high probability to help the best expert of the working period, the winner. The idea behind Winner_Driven is to use the selection function for critical pairs of the best expert to measure the facts of some of the other experts. Since the selection of the best facts can be delayed until the winner is determined, this idea can easily be implemented.

Provided that the expert whose best facts have to be selected differs very much from the best expert the following effect occurs: A heuristic for traversing a search space can be seen as following a kind of valley through a mountainous region. The mountains are those paths through the search space that have higher measures than the path that is actually taken. But behind the mountains may be a much deeper valley than the actual followed one that is not reached because of the mountain barrier (in our case facts that do not have a good measure but which produce other facts that would be measured good). Paths that are mountains for one expert may be valleys for another one. Therefore an expert may, after some time, produce facts that are very good according to the heuristic used by another expert but are also impossible to reach using this heuristic. Such facts are selected by Winner_Driven.

As we will see in the next section, Winner_Driven is a very generally usable select-good-results function. It is very stable under variation of the parameters (although an astronomically high *threshold* can greatly influence a proof run). A rip-off of the idea of Winner_Driven is to use not the function of the winner but always goal similarity functions (see [DF94]). This rip-off can also be very successful.

5 Experiments

In this section we will compare the select-good-results functions presented in section 4.2. For this purpose we have chosen several examples for which teamwork has been quite successful. These examples are taken from [ADF95] and [DK94]. We omitted those examples for which not the exchange of results caused the improvements but the change of the selection heuristic for critical pairs that is a result of choosing the best rated expert as basis for the new start state of a cycle.

Because Statistic has some parameters that can be adjusted to an example, we devote Statistic two columns in Table 1, one stating the run times with standard parameter values (St.st.) and another one stating the times for the best values found (St.b.). In each row of the table the same lengths of working periods and (of course) the same expert combinations were used, so that only the select-good-results functions of the referees had changed. In or-

der to give a hint at what the gain provided by teamwork is, we give in the last column the run times of the best known experts for the examples, provided we have been able to find an expert that is capable of solving them.

Ex.	St.st.	St.b.	Wi._Dr.	Last	b.exp.
bool5b	75.2	70.0	57.9	—	—
sa2	28.2	16.8	10.7	15.1	—
ra2	41.4	41.4	42.3	49.7	230.0
lusk6	460.6	312.2	341.0	495.8	3019.0
cal3	87.5	79.8	81.9	81.7	297.2
cal4	275.8	171.0	111.7	96.5	—
p2.a	6.2	6.2	5.7	6.9	79.5
p8.b	40.6	39.4	40.6	97.4	—
p10	26.6	26.4	23.0	—	—

Table 1: Comparison of different select-good-results functions on 2 SUN-ELC, run times in sec.

The examples cover a wide range of domains: bool5b boolean algebra, sa2 groups, ra2 robbins algebra, lusk6 rings, cal3 and cal4 propositional calculus and p2.a p8.b and p10 lattice ordered groups.

The first and most important observation that can be made in Table 1 is that the two more "intelligent" select-good-results functions Statistic and Winner_Driven enable our system to solve all examples even those that couldn't be proved in a sequential run. The importance of cooperation as result of these functions is emphasized by examples bool5b and p10 that could not be solved by using Last, thus indicating that good cooperation is necessary to solve them. But Last has its merrits, as can be seen in example cal4.

If we take a closer look at the results of Winner_Driven and Statistic, we can observe that for 7 examples Winner_Driven is better (i.e. faster) than the standard parameter setting of Statistic. And for the remaining 2 examples it is (nearly) as good as the standard version. With respect to the reduction of support by the user this is a very good sign, since Winner_Driven doesn't have many parameters that have to be set. On the other hand there are 4 examples for which a little fiddling with the parameters of Statistic produced better run times than those using Winner_Driven. So there is definitely room for some improvements. These improvements can either be a combination of Winner_Driven and Statistic or a planned use of specialized Statistic referees using the concept developed for experts in [DK94].

As a consequence of our experiments so far we use the standard version of Statistic for all experts in teams that do not include goal-oriented experts. Teams that use goal-oriented experts use Winner_Driven.

6 Conclusion

A very crucial part of teamwork based systems are the referees. Only good referees allow for synergetic effects and the forgetting of useless results, features that are among the great advantages of teamwork. After presenting the main ideas for experts and the supervisor in other papers we concentrated in this paper on the referees. Since a good result is not determined by a good inference that generated this result but by the consequences a result has for a proof attempt, concepts for select-good-results functions of referees are not only important for teamwork, but for each theorem prover.

In this paper we presented several simple concepts for such selection functions that have proven to be successful. The use of statistical data offers both a good starting point and adjustment to particular examples, while expert oriented selection functions offer robustness and stability. Even selecting the newest facts can be useful. So statistical criteria are a basis that should be strengthened by including other, knowledge-based criteria. One possibility that should be investigated in the future is to combine Statistic with Winner_Driven to achieve this strengthened basis.

The development of more concepts for referees is also important because of the growing number of distribution and cooperation approaches for automated theorem proving. In both areas the central problem is the huge amount of communication that real systems have to deal with. Since communication is quite expensive a major concern should be to avoid communicating unnecessary results, which can be achieved by using referee-like concepts. Even if one is purely interested in search control heuristics, forgetting of facts remain a necessity.

References

[AD93] **Avenhaus, J. ; Denzinger, J.**: *Distributing equational theorem proving*, Proc. 5th RTA, Montreal, LNCS 690, 1993, pp. 62-76.

[ADF95] **Avenhaus, J. ; Denzinger, J. ; Fuchs, M.**: *DISCOUNT: A system for distributed equational deduction*, Proc. 6th RTA, Kaiserslautern, LNCS 914, 1995, pp. 397-402.

[BG90] **Bachmair, L.; Ganzinger, H.**: *On restrictions of ordered paramodulation with simplification*, Proc. 10th CADE, Kaiserslautern, Springer, LNCS 449, pp. 427-441.

[CL73] **Chang, C.L.; Lee, R.C.T.**: *Symbolic Logik and Mechanical Theorem Proving*, Academic Press, 1973.

[DF94] **Denzinger, J. ; Fuchs, M.**: *Goal oriented equational theorem proving using teamwork*, Proc. 18th KI-94, Saarbrücken, LNAI 861, 1994, pp. 343-354.

[DK94] **Denzinger, J. ; Kronenburg, M.**: *Planning for distributed theorem proving: The team work approach*, SEKI-Report SR-94-09, University of Kaiserslautern, 1994.

[De95] **Denzinger, J.**: *Knowledge-Based Distributed Search Using Teamwork*, Proc. ICMAS-95, San Francisco, AAAI Press, 1995, pp. 81-88.

[HR87] **Hsiang, J.; Rusinowitch, M.**: *On word problems in equational theories*, Proc. 14th ICALP, Karlsruhe, LNCS 267, 1987, pp. 54-71.

[Mc94] **McCune, W.W.**: *OTTER 3.0 Reference manual and Guide*, Tech. Rep. ANL-94/6, Argonne National Laboratory, 1994.

Backwards Basic Factoring: A Pessimistic Analog to Basic Factoring

David Sharpe
University of New Brunswick
http://www.cs.unb.ca

Abstract

Basic factoring in the Selected Literal procedure is an inference rule that optimistically delays the proof of factorable goals until the last factorable goal is selected. This study proposes backwards basic factoring which is a *pessimistic* analog of basic factoring that reorders the proof of factorable goals to occur immediately when the first factorable goal is selected.

J.D.Horton and B.Spencer's clause tree methodology aids the discussion and implementation of backwards basic factoring. Backwards basic factoring is contrasted by inference counts with basic factoring and c-reduction on a small set of TPTP problems. It is found that backwards basic factoring generally takes fewer inferences than basic factoring (up to one half as many); however, it entails more computation per inference. Backwards basic factoring is better than c-reduction inference counts on one half of the test cases and both entail similar computation.

1 Introduction

Basic factoring in the Selected Literal (SL) procedure unifies the current unproved goal with another unproved goal in the search space [KK71]. Stickel raises the following concern with this strategy:

> "[I]f a pair of factorable goals do not happen to have a common provable instance, factoring them will ultimately result in failure." [Sti94 pp192]

This is a Bad Thing for a search procedure due to the implicit assumption that a failure at the current goal is naively delayed until failure occurs at the other factored goal, as is the case in SL. Stickel's concern, taken in concert with the assumption of delayed failure, motivates the following modification to SL. The proposed modification -- called backwards basic factoring -- reorders the proof of two factorable goals to occur immediately when a basic factor is detected as opposed to optimistically delaying the proof until the other factorable goal is selected. Backwards basic factoring alleviates Stickel's concern and at the same time maintains the inferential power of the SL procedure.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/459 ©1996 FLAIRS

2 Literature Review

This paper assumes the reader is familiar with concepts in clausal automated reasoning. Basic factoring is an inference rule for the SL procedure introduced by Kowalski and Kuehner [KK71]. SL is a top down automated reasoning procedure using the resolution principle [Rob65]; it is a refinement of Loveland's Model Elimination (ME) procedure [Lov69] since it is ME plus a stricter selection function and the basic factoring inference rule. Another pessimistic analog of basic factoring is c-reduction which was developed by Shostak in the graph construction procedure (GC) [Sho76]. C-reduction is integrated into analytic tableaux as folding-up in [LMG94] and into clause trees as left paths in the ALP procedure [HS95a].

Backwards basic factoring will be contrasted with basic factoring and c-reduction after clause trees and then the procedure is defined.

2.1 Clause Tree Definition

The clause tree methodology supports a description of the SL proof search space that makes backwards basic factoring immediate. The reader is referred to [HS95a/b] for a more complete development of clause trees.

Definition 1 ***Clause tree*** **[HS95b],** ***clause node, atom node, associated, merge path, precedes relation*** **($\prec$) and** ***chosen merge paths.***

$T=\langle N,E,L,M\rangle$ is a *clause tree* on a set S of input clauses over the atoms A such that:

1. $\langle N,E\rangle$ is a(n unrooted) tree.
2. L is a labelling of T given by $L: N\cup E\rightarrow S\cup A\cup\{+,-\}$. A node in N is either labeled by a clause in S and called a *clause node*, or labelled by an atom in A and called an *atom node.* Edges are labelled +or –.
3. No atom node is incident with two edges that are labeled the same.
4. Every edge $e=\{a,c\}$ in E joins an atom node a and a clause node c; e is *associated* with the literal $L(e)L(a)$.
5. For each clause node c, $L(c)=\{L(\{a,c\}),L(a)/\{a,c\}\in E\}$. A path $\langle v_0,e_1,v_1,\ldots,e_n,v_n\rangle$, where $v_i\in N$ and $e_i\in N$, is a *merge path* if $L(e_1)L(v_0)=L(e_n)L(v_n)$. Path $\langle v_0,\ldots,v_n\rangle$ *precedes* ($\prec$) path $\langle w_0,\ldots,w_m\rangle$ if for some $i=1,\ldots,m-1$, then $v_n=w_i$.

6. M is the set of merge paths called *chosen merge paths* such that: (i) the tail of each merge path is a leaf (called a *closed leaf*), (ii) the tails are all distinct and different from the heads, and (iii) the precedes relation $\prec$ on M can be extended to a partial order. ■

Definition 2 *Elementary clause tree, open leaf,* and *resolving clause trees.*
A clause tree with a single clause node is said to be an *elementary clause tree* and corresponds to a disjunction of literals (a clause). An *open leaf* is an atom node leaf that is not the tail of any chosen merge path. A clause tree corresponds to a disjunction of the literals at its open leaves. Two clause trees T_1 and T_2 can be resolved to form a new clause tree by identifying two identically labelled open leaf atom nodes, one from each of T_1 and T_2, but with incident edges signed differently. ■

Definition 3 *Tautology path.*
A *tautology path* joins two complementary literals. ■

Definition 4 *Legal* paths, *visible* nodes.
A set P of paths in a clause tree is *legal* if the precedes relation $\prec$ on P can be extended to a partial order. A merge path P is *legal* in $T=\langle N,E,L,M\rangle$ if $M\cup\{P\}$ is legal in T. If the path joining t to h is legal in T, then h is said to be *visible* from t. ■

Definition 5 *Unifiable (merge) paths,* and *choosing a (unifiable) merge path.*
Let $T=\langle N,E,L,M\rangle$ be a clause tree that contains two distinct atom nodes v_0 and v_n such that the atoms that label v_0 and v_n are unifiable by a substitution θ, and v_n is visible from v_0. A unifiable path P in T is given by $P=\langle v_0,e_1,v_1,\ldots,e_n,v_n\rangle$. P is a *unifiable merge path* if $L(e_1)=L(e_n)$ and θ is non-identy. *Choosing* a (*unifiable*) *merge path* P to T results in the clause tree $T_1=\langle N,E,L\theta,M\cup\{P\}\rangle$. ■

Definition 6 *Clause tree derivation.*
Given a set S of clauses, a derivation of T_n from S is a sequence $T_1,\ldots,T_n$ of clause trees such that for $i=1,\ldots,n$:

1. T_i is the elementary clause tree of a clause in S,
2. T_i is the result of resolving T_j and T_k, $j<i$ and $k<i$, or
3. T_i is the result of choosing a leaf to leaf merge path in T_j, $j<i$. ■

Definition 7 *Clause tree refutation proof.*
If a clause tree T has no open leaves, then T represents a *clause tree refutation proof.* ■

3 Backwards Basic Factoring in ModSL

SL augmented with backwards basic factoring and early tautology detection is defined as *ModSL* in Procedure 1. In contrast to SL, ModSL detects backwards basic factors before ancestor merge paths, and it detects all backwards basic factor paths at selection time (i.e., repeat step 2 until no backwards basic factors exist).

Procedure 1 *ModSL*
Given is a set S of satisfiable clauses, a refutational goal clause C, and a depth-first left to right selection function ϕ. Initially T is the elementary clause tree of the top clause C.

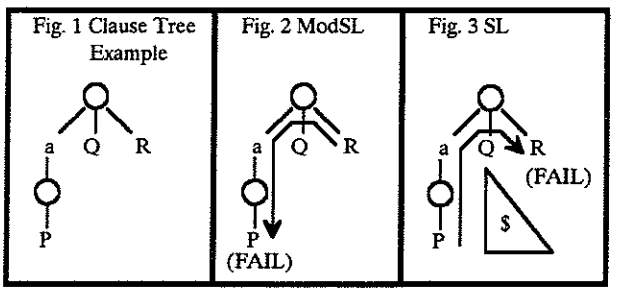
1. Select the current goal node $g=\phi(T)$.
2. [Backwards basic factor] If a (unifiable) merge path P from an open leaf to the current node g exists, (nondeterministically) choose P and repeat 2.
3. [Ancestor merge] If a (unifiable) merge path P to an ancestor of g exists, (nondeterministically) choose P and go to 6.
4. [Extension] If an elementary clause tree E from S exists such that E has no shared variables with T and E can be resolved with T at g under a substitution θ, then nondeterministically pick an E, apply θ to E and T, resolve E and T, and continue to 5. Else fail to previous nondeterministic choice.
5. [Tautology detection] If a tautology path exists from any member of $E\setminus\{\neg g\}$ to an ancestor of g or any open leaf, fail to the previous nondeterministic choice. Else continue to 6.
6. [Continue] Now g is closed node. If there are no open nodes, exit with success. Otherwise go to 1.

4 Backwards Basic Factoring Versus Basic Factoring

The motivation for backwards basic factoring is illustrated by the following example. Consider a clause subtree with three open leaves to be selected in order: ***P***, ***Q***, ***R***. Suppose that ***P*** and ***R*** are factorable and that ***Q*** requires a computationally expensive proof with a certain amount of nondeterminism (denoted "$" in Fig. 3 below). Further, *pessimistically* suppose that ***P*** and ***R*** do not share a proof. (Fig. 1, where "a" is an arbitrarily labelled atom node, edges are not labelled).

The ModSL procedure first selects ***P*** as the goal, then chooses the backwards basic factor merge path from ***R*** to ***P***, and fails to the previous nondeterministic choice (Fig. 2).

In contrast, SL first selects ***P*** as the goal, then optimistically chooses the basic factor merge path from ***P*** to ***R***, then proves ***Q***, and selects ***R*** as the goal and fails (Fig 3). The failure will cause backtracking through the proof of ***Q*** until ***P*** is again the current goal and the attempt to prove ***P*** fails to the previous nondeterministic choice.



However, backwards basic factoring does not guarantee fewer inferences -- as might be concluded from the example above. The reason for this is that if the current factorable goal is provable, but an open leaf sibling is not, then SL will fail earlier whereas ModSL will delay failure by proving the factorable goals. This corresponds to reordering the selection of literals within a single extension clause. Neither SL nor ModSL is strictly superior with a depth first left to right selection function; however, if an effective pessimistic heuristic is used that selects the open literal most likely to fail, then ModSL would outperform SL.

The real benefit of backwards basic factoring is how it behaves in concert with a chronological backtracking search engine (the common search engine for SL). When ModSL fails due to factorable goals not having a common proof, it backtracks through the basic factoring inference in one step to try another extension clause. Contrast this with SL failing due to factorable goals not having a common proof, then SL backtracks through the entire search space generated between the factorable goals before it backtracks through the basic factoring inference to try another extension clause. This backtracking commonly costs SL a large number of inferences since it effectively *thrashes* through the search space by exhausting all the nondeterministic choices. In short: backwards basic factoring combines with backtracking search in a more intelligent manner than basic factoring.

5 Backwards Basic Factoring Versus C-reduction

As mentioned, c-reduction is another pessimistic analog of basic factoring. To contrast c-reduction with (backwards) basic factoring, consider a clause tree with two factorable open leaves, P and Q, where P is the current goal. The procedure with c-reduction completes a proof at P, and when Q is selected as the goal, the *c-reduction* inference is choosing a merge path from Q to the *c-literal* P. Note that the merge path must still be legal (i.e., P is visible from Q) and P and Q must still be unifiable. In general, every visible literal with a complete clause tree proof is a c-literal. Hence, c-reduction detects (backwards) basic factors but at a different point in the procedure.

This means that c-reduction will avoid the over instantiation of variables in factorable literals.[1] Consider a clause tree with two open leaves: $P(a,Y)$ and $P(X,b)$. Suppose that $P(a,Y)$ and $P(X,b)$ both have proofs but $P(a,b)$ does not. When either leaf is selected, (backwards) basic factoring yields the unprovable goal $P(a,b)$; but, backwards basic factoring fails immediately while basic factoring optimistically delays failure. Whereas a procedure with c-reduction will prove one of the leaves and the other leaf will have a c-reduction only if the two leaves are still unifiable.

On the other hand, (backwards) basic factoring detects tautology paths earlier than c-reduction, and tautology detection allows the procedure to prune the search space. Consider a clause tree with two identical open leaf atoms with opposite signed edges: $\neg P$ and P. Obviously, such a tautology cannot aid in a refutation proof and the procedure should not explore such a search space. (Backwards) basic factoring will detect the tautology ($\neg P$ and P) immediately; whereas a procedure with c-literal tautologies (e.g. Horton and Spencer's ALP [HS95a]) will attempt to prove one of the leaves and if successful will detect the tautology only when the other leaf is selected. The tautology detection causes the procedure to backtrack through the entire search space generated between $\neg P$ and P. This backtracking commonly costs a large number of inferences since it effectively *thrashes* through the search space by exhausting all the nondeterministic choices.

6 Illegal Sets of Paths

A major disadvantage of ModSL is that merge paths will not necessarily form a legal set; whereas merge paths in SL will always form a legal set [HS95a]. GC with c-reduction shares this disadvantage with ModSL [HS95a]. The reason ModSL sacrifices this feature is that it can build irregular proofs (proofs with duplicate atoms in the set of ancestor nodes) which is illustrated with the following argument. Consider a ModSL clause tree with a backwards basic factor merge path whose head is h. Suppose that a descendant of h is the tail of a potential ancestor merge or tautology path to an interior node of the path. However, such a potential ancestor path is illegal due to a conflict with the path: no interior nodes of the path are visible to descendants of h. This argument implies that the proof of a backwards basic factor head will require a test for a backwards basic factor and an update of the visible nodes before and after a backwards basic factor inference is completed. It turns out that backwards basic factoring and c-reduction involve much of the same computation overhead. That suggests that backwards basic factoring might be added to c-reduction at little extra computation cost. That is the topic of current research [Sha96].

7 Empirical Results

Table 1 summarizes the results of running GC, ModSL, and SL on a small set of TPTP problems [SS94]. This paper is a preliminary study, hence short problems are tackled; in general, any difference in performance may become arbitrarily large on harder problems. ModSL is implemented in Prolog, the SL results are from restricting the ModSL implementation to SL with early

[1] This characteristic of c-reduction and the following example is due to the anonymous reviews.

tautology detection. The GC results are reported in [SHF95].

ModSL, SL, and GC are compared by inference counts (a merge path or a resolution extension). From the problems tested, ModSL took more inferences than SL in only one case: GRA001-1, the reason is due to the selection order of open leaves (Section 4). For the other problems tested, ModSL consistently took fewer inferences: from 0.5 to 1, the average is 0.70. Backwards basic factoring is equal or better than c-reduction on half of the test cases. However, when c-reduction is better, it takes an average of 0.50 the number of inferences; whereas, when backwards basic factoring is better, it takes an average of 0.85 the number of inferences.

TABLE 1: Inference Counts

TPTP Problem *	GC	ModSL /GC	ModSL	ModSL /SL	SL
GRA001-1	40	1.58	63	1.11	57
GRP003-1 *	24	3.75	90	0.87	103
GRP034-4 *	113	0.63	71	0.89	80
MSC002-1 *	2059	0.84	1729	1	1729
MSC003-1 *	94	0.99	93	1	93
MSC007-1.004	99	1.03	102	0.63	162
MSC007-1.005	674	1.09	734	1	734
PUZ004-1	11	1	11	0.52	21
PUZ009-1	18	1.83	33	0.77	43
PUZ013-1	26	0.96	25	0.63	40
PUZ014-1	45	2.47	111	0.62	180
PUZ033-1	19	1.16	22	0.63	35
SYN001-1.003	27	0.89	24	0.77	31
SYN001-1.004	113	0.74	84	0.85	99
SYN010-2.002	19	3.32	63	0.5	126
SYN044-1	11	0.91	10	0.67	15

* Starred problems are first-order logic.

8 Conclusion

The advantage of backwards basic factoring relates to maintaining the inferential power of SL yet alleviating Stickel's concern that basic factoring optimistically delays failure. If a backwards basic factor is detected, then Stickel's quote in Section 1 is still strictly correct, except now the assumption that failure is delayed is no longer valid: the proof of a backwards basic factor path will now "ultimately fail" immediately at the current goal.

This study shows that backwards basic factoring is sometimes better and sometimes worse than either basic factoring or c-reduction. Understanding if and when backwards basic factoring is an appropriate inference rule is still an open question.

References

[HS95a]Horton, J. and Spencer, B. *Clause Trees: a Tool for Doing and Understanding Automated Theorem Proving*, UNB TR95-095, 1995.

[HS95b]Horton, J. and Spencer, B. *A Top Down Algorithm to Find Only Minimal Clause Trees.* in preparation, 1995.

[KK71]Kowalski, R. and Kuehner, D. "Linear Resolution with Selection Function" in *Artificial Intelligence* , 2, 227-260, 1971.

[LMG94]Letz, R, Mayr, K, and Goller, C. "Controlled Integration of the Cut Rule into Connection Tableau Calculi" in *Journal of Automated Reasoning 13*, 297--337, 1994.

[Lov69]Loveland, D. "Theorem-Provers Combining Model-Elimination and Resolution" in **Meltzer, B. and Michie, D. (eds.)** *Machine Intelligence 4*, 73--86, University Press, Edinburgh, 1969.

[Rob65]Robinson, J. A. "A Machine-Oriented Logic Based on the Resolution Principle" in *Journal of the ACM*, 12(1):23 -- 41, 1965.

[Sha96]Sharpe, D. *The AllPaths Automated Reasoning Procedure with Clause Trees,* Faculty of Computer Science, University of New Brunswick, in preparation, 1996.

[Sho76]Shostak, Robert E. "Refutation Graphs" in *Artificial Intelligence* 7(1), 51 -- 64 1976.

[SHF95]Spencer, B., Horton, J. and Francis, K. . *Experiments with the ALPOC Theorem Prover,* UNB TR95-094, 1995.

[Sti94]Stickel, Mark E. "Upside-Down Meta-Interpretation of the Model Elimination Theorem-Proving Procedure for Deduction and Abduction" in *Journal of Automated Reasoning 13*, 189 -- 210, 1994.

[SS94]Suttner, Christian. and Sutcliffe, Geoff. *The TPTP Problem Library: TPTPv.1.1.1*, Technical Report AR-94-03, 1994.

TEACHING BAYESIAN NETWORKS THROUGH INTERACTIVE EXPLORATION

Varina Hammond
hammond@cs.rpi.edu

Ellen L. Walker
walkere@cs.rpi.edu

Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180-3590

Abstract

Bayesian Networks are an extremely powerful and useful method for reasoning. However, surprisingly few people outside of a small community of Artificial Intelligence researchers actually have a working knowledge of Bayesian Networks. We describe in this paper our goal of enabling students to more readily use Bayesian Networks, the software tool we have developed to achieve this goal, and our motivation for the various design aspects of the tool.

1 INTRODUCTION

When faced with something new, people commonly ask two questions, who and why. Here, those questions would be: **who** uses Bayesian Networks and **why** are they used?

Bayesian Networks are typically used by Artificial Intelligence researchers who have an excellent background in and understanding of reasoning and probability theory. Applications for Bayesian Networks include medical diagnosis, traffic scene analysis [7], map learning, and computer vision/perceptual grouping problems [8]. Charniak [2] says that "It is probably fair to say that Bayesian networks are to a large segment of the AI-uncertainty community what resolution theorem proving is to the AI-logic community."

This comment leads us to another why: **why** aren't more people (especially students) using Bayesian Networks if they are so useful and wonderful? We believe this is largely due to an overwhelming lack of comprehensible sources. Overall, textbooks for Introduction to AI courses do not handle this subject matter well, providing a sketchy or vague treatment of Bayesian Networks. Often, the descriptions rely heavily on prior knowledge of probability theory, uncertainty, and reasoning – areas that many students have difficulty grasping. Instructors working with these textbooks have a hard time communicating effectively the concepts of Bayesian Networks and need an additional resource in the form of a visual teaching aid.

In light of this, **how** can we make Bayesian Networks easier to understand? More specifically, **how** can we teach students in Introduction to AI courses about Bayesian Networks? We have developed BayNEx (Bayesian Network Explorer) as an answer to these questions. The goals for the tool are to:

- Make learning Bayesian Networks fun.
- Provide examples that clarify concepts.
- Teach the mechanical processes of Bayesian Networks, as well as the theory behind it.
- Enable the use of Bayesian Networks to become more widespread.

2 DESIGN MOTIVATIONS

In creating BayNEx, there were several design issues to be taken into consideration. The next sections describe some of these issues and our solutions.

2.1 How To Best Facilitate Learning

There is no one set way that students learn. Rather, students learn through various combinations of methods. The most common methods are:

1. Learning through using pre-built examples.
2. Learning through building self-designed examples (modifying existing examples or creating new examples both fit into this category).

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/463 ©1996 FLAIRS

3. Learning through putting the concepts into practice via a direct application.

We wanted our software to be able to support diverse learning methods (these specifically), so we included a facility through which students could explore various sample networks that are included with the software (or their own networks if they so choose), another that allows building of networks from scratch and one for modifying, and still another in which students participate in a question-and-answer game. These facilities are the Explorer, the Network Builder, and the Knowledge Tester, respectively, and will be described in more detail in sections 3.2, 3.1, and 3.3.

2.2 Interface Considerations

After functional considerations, the design of the interface was the next important aspect of the tool to consider. Students tend to be less intimidated by streamlined, uncluttered packages whose purpose can be easily grasped from the placement and labeling of key buttons and graphics. Furthermore, we decided upon a modal interface for the bulk of the tool, as several other UNIX-based packages (such as xfig, xpaint, and vi) employ the same principle. To build our interface, we used the SUIT [3] [4] package.

3 BayNEx FACILITIES

As was mentioned previously, BayNEx is divided into three main sections: the Explorer, the Network Builder, and the Knowledge Tester. The following sections give more details on the functionality and design considerations of each, as well as describing another component of the tool, the Help Facilities.

3.1 Network Builder

The Network Builder component allows students to build their own Bayesian Networks (learning via method #2). Either a singly-connected (one unique path exists between two nodes) or a multiply-connected (more than one path between two nodes) network may be created. Cyclic networks are not permitted. Building a network involves adding (or deleting) nodes, adding (or deleting) links between nodes, and editing nodes to specify their probabilities. Figure 1 shows a singly connected network example from [2] in the editing stage. The left portion of the window shows graphically the network that is being built, and the right portion is the editing window that allows the probabilities for the current node (in this case, *dog-out*) to be entered.

To aid students in the editing process, the Network Builder provides a clean modal interface which notifies the user of needed actions. If any necessary information is left out (either in the editing or saving process), the Network Builder flags the error and prompts students to enter the pertinent data.

Since student-built networks may be at first less likely to demonstrate the features of Bayesian Networks or to make explicit the important relationships between nodes (evident in the exploration process described in the next section), some prebuilt networks are included for loading: singly- and multiply-connected networks are both available for students to modify if they so choose. This functionality is helpful later on in the learning process as well, as students begin to build more complex networks. Each network is specified in its own file.

3.2 Explorer

The Explorer is the heart of this learning tool (as it shows Bayesian Networks in action) and facilitates learning via method #1. It begins by representing the network specifications graphically such that students can easily find where all the information is located. The nodes are shown named, and are connected via directed arcs. Each node has a belief associated with it, represented by an attached, smaller node that is shaded according to its value (black = no belief in the node, white = complete belief in the node, and various shades of grey represent the belief values in-between). The belief nodes are buttons that, when pressed, show the numerical belief value and allow changes to that value to be made. Associated with root nodes (no parent) is a prior probability node button, and associated with the other nodes are conditional probability node buttons, all of which yield the appropriate information and allow for changes when pressed. Figure 2 shows a singly-connected network that was taken from [2] as it appears in the BayNEx Explorer (70% belief was associated with the *family-out* node).

We believe that the graphical representation of the system and information is crucial to the understanding of Bayesian Networks. The clearer the display, the more comfortable students will be with the information and the process, the more likely students will be to fully explore the network, and the less likely students will be to get frustrated.

Multiply-connected networks require several more steps than singly-connected networks before they are ready for evaluation. The software automati-

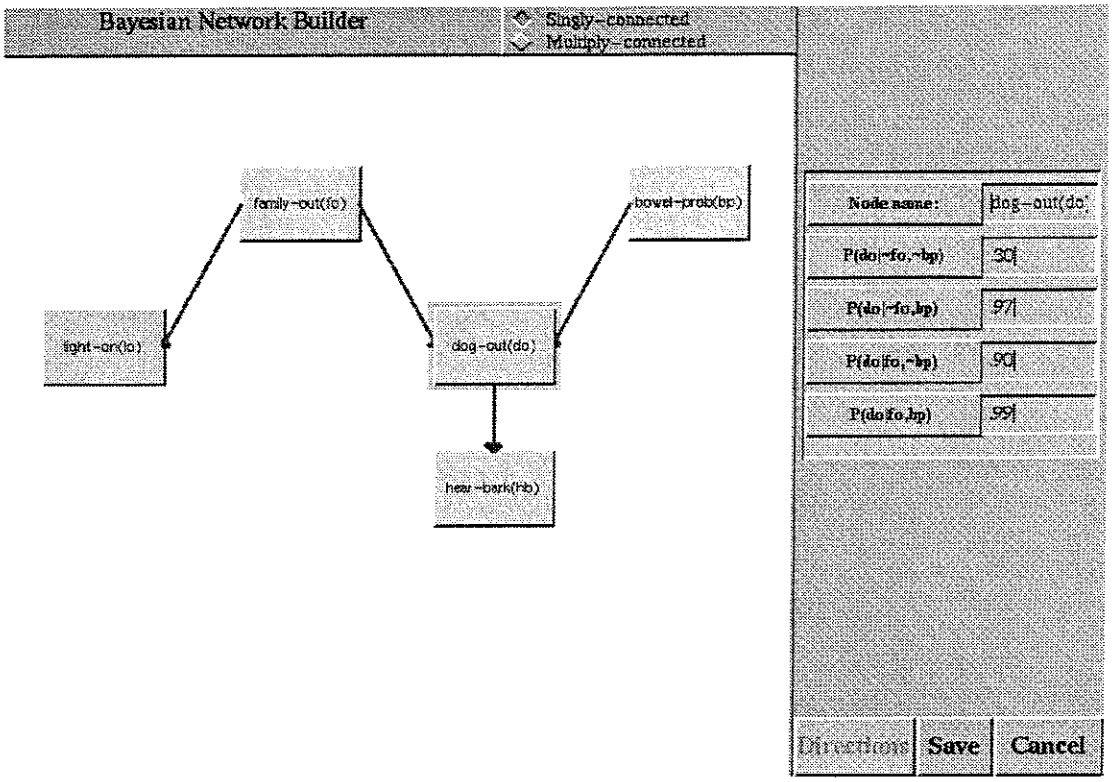


Figure 1: Editing the *dog-out* node in Charniak's singly-connected network example [2].

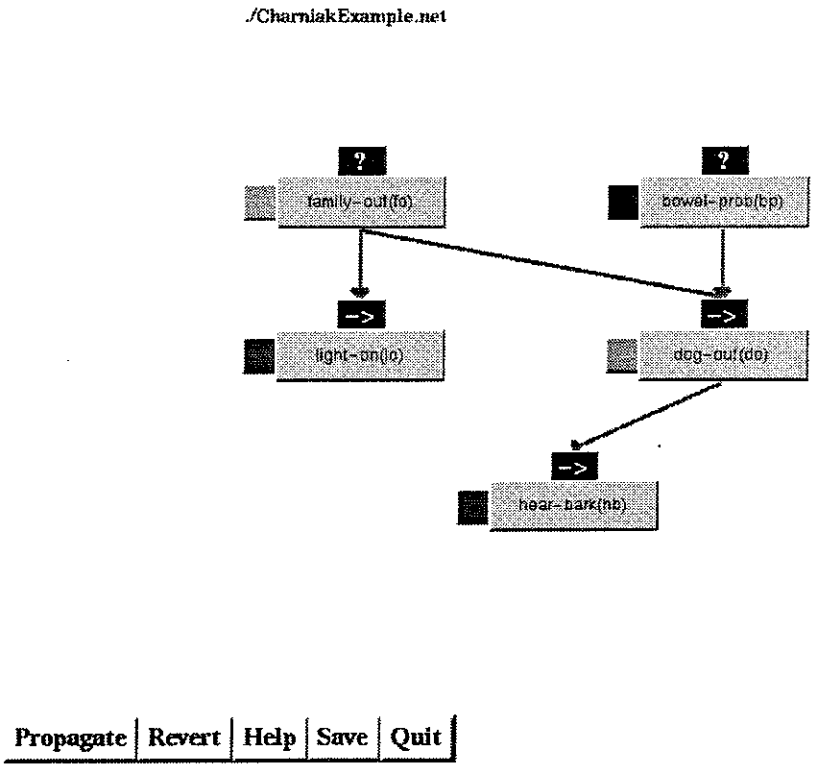


Figure 2: Showing Charniak's example [2] in the Explorer window.

cally forms a junction tree of belief universes (combinations of nodes), using algorithms developed by Anderson, Jensen et al [1] [6] and Draper [5]. The onset of each step in this process is flagged by text that students may click on to continue with the process or to receive more information, thus enabling the students to learn about the process at their own speed. Students may choose to have the junction tree displayed next to the original network for clarification and completeness, as the software uses color schemes that enable students to clearly see the correspondence between the two.

Upon changing belief or probability values, students needs a way to update the network to reflect these changes. In order to "run" the network, students click on a button to propagate the new information. Propagation is shown on the original network for the singly-connected case and on the junction tree for the multiply-connected case. Updated probabilities and belief information are readily viewable in the appropriate nodes. The speed of propagation is controllable by students, and the propagation itself is represented by an animation of arrows moving along the directed arcs on the network or junction tree.

The students also have the capability to undo, or revert, the last propagation that was completed. This way, if altering certain beliefs or probabilities did not yield the results that the students were expecting, they can undo the propagation, change other beliefs or probabilities, and re-propagate. The process of changing beliefs and re-evaluating to yield a new result helps students learn some of the theory and mechanics of Bayesian Networks.

3.3 Knowledge Tester

To further facilitate learning of the theory behind Bayesian Networks (via learning method #3), the tool has a game-playing mode, based on a type of sleuthing/guessing game. Games can be played on the pre-built networks that are provided and all the usual information about beliefs and probabilities is still available.

The steps of the sleuthing game are:

1. The system provides students with a list of possible alterations to the network (eg: different belief or probability values).

2. Students choose one modification from the list.

3. Students are prompted to choose from a list of possible outcomes what they believe will be the correct result of the chosen modification.

4. The system propagates, and pronounces the answer right or wrong, providing immediate feedback to the students, much like an actual instructor.

A scoring system is kept in order to provide incentive to keep playing and learning. This would work well in an environment where students are working side-by-side, as in a contest.

A student who has spent time using the Explorer and the Network Builder will of course be more adept at the guessing game, but even the novice user can benefit from playing the game.

3.4 Help Facilities

In order for the software to be more useful as a learning aid, on-line help and documentation is readily available to the students. We believe that cogent information incorporated into the system serves to enhance the learning process and helps to give insight into both the theory and mechanics of Bayesian Networks.

A "Getting Started" section explains loading, building, and altering networks. A short section is devoted to explaining in general the process of propagation and the idea behind beliefs and probabilities. Additionally, we include this information in different, appropriate areas of the tool so that the students are able to access pertinent information in just a few clicks.

Required with each network, pre-built or student-designed, is a textual description of the network. Oftentimes it is not clear from just the graphical representation and naming convention what is happening in the network, and a description can increase understanding in two ways:

1. Reading the description of a pre-built network may be likened to reading a story, so the network makes more sense to students.

2. Writing a description for a network that the students build serves to help them design and better organize the network.

In addition, we have designed an integrated reference utility within this tool to encourage students to read and explore more on the subject of Bayesian Networks. It contains an enumeration of various references to topics in Bayesian Networks and a short commentary about each reference.

4 USE OF BayNEx IN THE CLASSROOM

The BayNEx tool can be used in several ways within an introductory artificial intelligence course: as an in-class demo, within a closed laboratory, or as an outside-class homework assignment in a class that does not use closed labs. In a classroom equipped with computer projection equipment, the BayNEx software can be used as the primary introduction to Bayesian Networks, using the Explorer to demonstrate a relatively simple network whose domain is familiar to the students (such as the network shown in Figures 1 and 2). First the static network structure would be presented, then belief propagation could be demonstrated. By the nature of the tool, the major features and advantages of Bayesian Networks would be presented qualitatively first, then the mathematical basis for the behavior could be presented using the Network Builder.

The major purpose of the tool, however, is not its use for classroom demonstration, but its use by the students themselves. This use would be encouraged in a closed laboratory session or as a homework assignment. In either case, students would be given a worksheet to complete by creating and exploring networks with the BayNEx tool. Questions on the worksheet would lead them through all components of the tool. In the Explorer, for example, students would be asked to bring up a specified network, make specified changes, and explain the results. In the Network Builder, students would be asked to build a network to represent a given situation, and to use their network to answer "what if" questions (and explain the results). Finally, students would be asked to play a game from the Knowledge Tester for a given amount of time (in a closed lab) or until they achieve at least a minimum score (in a homework assignment). Alternatively, a competition could be arranged, with extra credit awarded to the winners.

5 CONCLUSION

In order to enable Bayesian Networks to be a more easily understandable concept for students in Introduction to AI courses, we have developed a software tool called BayNEx (Bayesian Network Explorer). This tool supports various learning methods so that it is useful to a wide range of students.

We designed BayNEx with three main components: the Explorer, the Network Builder, and the Knowledge Tester. Students may choose to begin their learning in any of these areas. The Explorer allows students to load a network and change its values, learning about the way information is propagated through a network. The Network Builder helps students understand the underlying structure of a network and learn what *cannot* be specified in a network. The Knowledge Tester is a game environment that some students would find more enjoyable for learning. Students should be able to gain a working knowledge of Bayesian Networks more quickly and easily through the use of this software.

References

[1] Andersen, Stig K., Oleson, Kristian G., Jensen, Finn V., and Jensen, Frank, "HUGIN: A Shell for Building Bayesian Belief Universes for Expert Systems", *Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, Volume 2, pp. 1080-1085, 1989.

[2] Charniak, Eugene, "Bayesian Networks Without Tears", *AI Magazine*, Volume 12, Number 4, pp. 50-63, 1991.

[3] Conway, M., Pausch, R., and Passarella, K., *A Tutorial for SUIT, the Simple User Interface Toolkit*, Computer Science Department, University of Virginia, 1992.

[4] Conway, M., *The SUIT Version 2.3 Reference Manual*, Computer Science Department, University of Virginia, 1992.

[5] Draper, Denise L., "Clustering Without (Thinking About) Triangulation", *Uncertainty in Artificial Intelligence-95*, 1995.

[6] Jensen, Finn V., Oleson, Kristian G., and Andersen, Stig K., "An Algebra of Bayesian Belief Universes for Knowledge-Based Systems", *Networks*, Volume 20, Number 5, pp. 637-659, August 1990.

[7] Huang, T., Koller, D., Malik, J., Ogasawara, G., Rao, B., Russell, S., and Weber, J., "Automatic Symbolic Traffic Scene Analysis Using Belief Networks", *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-84)*, Volume 2, pp. 966-972, 1984.

[8] Sarkar, Sudeep and Boyer, Kim L., "Integration, Inference, and Management of Spatial Information Using Bayesian Networks: Perceptual Organization", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 15, Number 3, pp. 256-273, March 1993.

PEDAGOGIC RESOURCES FOR ARTIFICIAL INTELLIGENCE IN THE UNDERGRADUATE COMPUTER SCIENCE CURRICULUM

Bill Manaris
Computer Science Department
University of Southwestern Louisiana
Lafayette, LA, 70504
manaris@usl.edu

Ingrid Russell
Department of Computer Science
University of Hartford
West Hartford, CT 06117
irussell@uhavax.hartford.edu

Keywords: Artificial Intelligence, Curriculum Issues, Instructional Software, Laboratory Environments.

Abstract

The past decade has witnessed a rapid growth in practical applications of Artificial Intelligence (AI) in a variety of fields. This, coupled with the fact that AI has influenced many of the more established disciplines in computer science, has made clear the importance of AI in today's undergraduate curriculum. Several challenges confront instructors of the Introduction to AI course. In addition to the decisions regarding what to cover in the course, and how to present the material so that topics connect together as a unified whole, a major challenge faces these instructors, namely how to locate available educational resources and other material for the course.

This paper presents preliminary results from an on-going collaborative project with the American Association of Artificial Intelligence involving the establishment of an AI educational repository. This repository is intended to provide support for instructors in locating and integrating available educational resources into the AI course.

1 INTRODUCTION

Artificial Intelligence appears to have finally emerged from its "winter," a period characterized by minimal to non-existent coverage in Computer Science (CS) curricula, significant cuts in funding, and a strong skepticism by the CS mainstream [9]. For instance, last year's ACM Turing Award (viewed by many as the most prestigious award in Computer Science) was presented to two AI pioneers, namely Edward Feigenbaum and Raj Reddy. Moreover, a 1993 U.S. Commerce Department survey indicated that international AI software sales reached $900 million, and that 70% of the top 500 U.S. companies had incorporated AI tools into their operations [12]. As indicated by Munakata [9], during the last decade AI has found its way into commercial and industrial applications, and has quietly changed our lives. This trend is also evident in CS, in that AI has influenced other more established CS subdisciplines, such as Databases, Software Engineering, User Interfaces, and Computer Graphics. As Aiken [1] points out, AI provides the "cultural foundation" of CS by coherently joining together ideas and concepts from other disciplines, such as Mathematics, Engineering, Psychology, Neurosciences, and Philosophy into a unifying computational framework.

In the past few years, coverage of AI topics in the CS curriculum has been addressed numerous times. For instance, the ACM/IEEE-CS Curricula 1991 makes specific recommendations regarding coverage of AI topics in the curriculum [19]. Additionally, a number of papers have appeared discussing the various topics and concepts to be included in undergraduate introductory AI courses and providing pointers to related resources [5, 18]. Finally, several papers have identified various delivery methods for presenting such material, ranging from preparing a three-credit course in AI to spreading the material over a number of courses in the curriculum [7, 11, 18].

Currently, there seem to be two popular approaches to presenting the AI course material, namely the more traditional approach, as presented in several AI textbooks [2, 3, 13, 16], and a recent alternative, the agent-oriented approach, which introduces models of and development techniques for independent entities which are capable of exhibiting intelligent behavior within limited and well-defined application domains [16, 17]. Regardless of the approach used, the material covered includes the following fundamental topics: Knowledge Representation, Logic and Reasoning, and Search and Game Playing.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/468 ©1996 FLAIRS

Additionally, an introduction to several advanced AI topics may be offered so that students can obtain a broader foundation in the area and be prepared for subsequent courses, graduate school, and/or the job market. These advanced topics include: Expert Systems, Fuzzy Logic, Natural Language Processing, Neural Networks, Robotics, and Vision.

Finally, in competition with the above topics, traditional AI programming languages, such as LISP and Prolog, may also be introduced in the course(s).

2 CHALLENGES IN AI EDUCATION

Concerns voiced by AI educators regarding difficulties associated with teaching the Introduction to AI course was a major motivation for the American Association of Artificial Intelligence (AAAI) Fall Symposium on "Improving Instruction of Introductory AI," held in New Orleans in November 1994. One of the main goals of the symposium was to provide a forum for the exchange of ideas/experiences among instructors teaching the course and facilitate discussions on related issues [4].

Some of the main issues addressed by the sumposium participants were: (a) themes used to structure the AI course, (b) course goals and outcomes, (c) the role of programming in the AI course, and (d) course content and its place in the undergraduate CS curriculum.

A similar forum took place at the NSF Workshop on "Providing and Integrating Educational Resources for Faculty Teaching Artificial Intelligence," and its subsequent Reflection Day held in Philadelphia in June 1995, and February 1996, respectively. It was generally agreed that preparing material for the Introduction to AI course is a major challenge, mostly due to the diversity of the topics that might be covered. This difficulty is more pronounced at institutions that do not have a strong AI research core to provide the discussion support needed by those teaching the course.

Additionally, the diversity of student backgrounds, as they enter the course, raises an additional challenge in preparing materials for and deciding which resources to use in the course. Although numerous AI resources are available, a large percentage of them are commercial and production-environment oriented and thus not necessarily appropriate for or desirable in AI pedagogy.

Finally, and most importantly, in addition to determining what to cover in the course and how to present the material so that topics connect together as a unified whole, a major challenge faces instructors of the Introduction to AI course. This challenge is how to locate available educational resources and material for the course.

3 MOTIVATION FOR THE AI REPOSITORY

Within the last three years, due to the rapid development of the World-Wide-Web (WWW), a wealth of material has become available that could be used in AI pedagogy [5, 6, 18]. Nevertheless, in most cases, this material is distributed and usually interleaved/hidden among other AI-related material that would not necessarily be of direct help to the AI educator. Therefore, it would be extremely beneficial for AI educators to be able to locate and have access to AI-related resources that are education oriented. Moreover, such benefit would be maximized if these resources were provided in a well-organized and structured fashion, in conjunction with information on how to integrate them into the course and how to best utilize various other pedagogical techniques and material.

Specifically, in her concluding remarks during the 1994 AAAI Fall Symposium mentioned above, Barbara Grosz (who was president of AAAI at that time) committed to accomplishing the following two goals:

- Establishment of an AI Educational Repository sponsored by AAAI geared towards the Introduction to AI course.
- Development of tutorials on advanced topics which are normally covered in the AI course, such as Expert Systems, Natural Language Processing, and Neural Networks, and Vision. These tutorials are to be presented during subsequent AAAI and IJCAI conferences.

This paper presents results from the continuing collaborative effort with AAAI on the creation of such a repository. This repository is a centralized distribution point which (a) focuses on educational resources and material to be used in undergraduate AI pedagogy, (b) identifies and organizes such resources by topic (as specified above), and (c) provides a wide selection of resources applicable to various student expertise levels and budgetary considerations/constraints.

4 OVERVIEW AND STRUCTURE OF THE REPOSITORY

As indicated above, the repository is intended as a central registry and distribution point for resources and materials related to AI pedagogy. It is designed to contain:

(a) existing tools and environments that may be utilized in the classroom to facilitate learning of various aspects and topics in AI,

(b) instances and pointers to syllabi, sample programming assignments, sample written assignments,

(c) on-line tutorials on specific AI topics,

(d) papers related to AI pedagogy, and

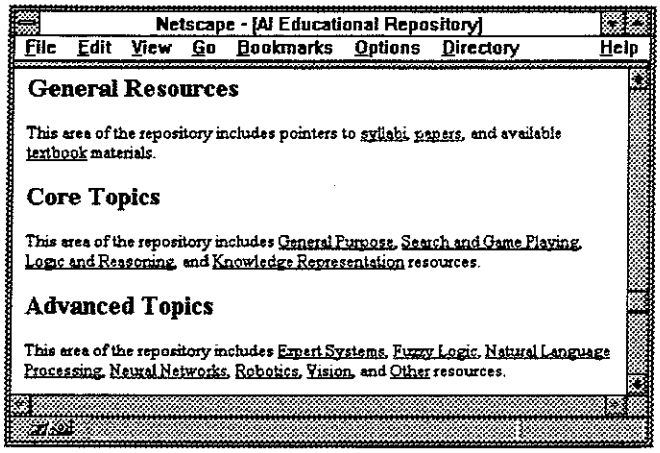


Figure 1. Excerpt from the AI Educational Repository's First-Level Entry.

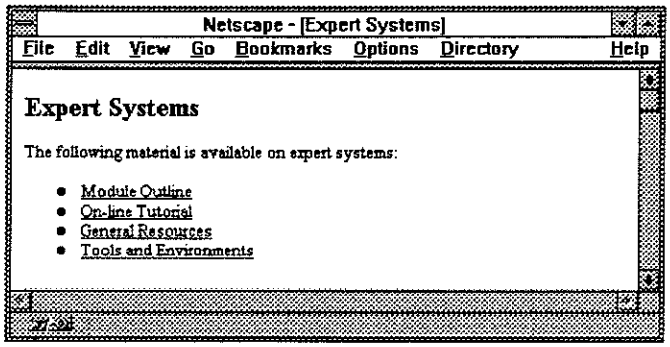


Figure 2. Expert Systems Entry.

(e) pointers to source code included in popular AI textbooks (as they become available on the WWW).

Figure 1 shows an excerpt from the top-level page of the repository.

4.1 Selection Criteria

While researching the currently available resources related to AI, we discovered that the majority of them are geared towards production environments as opposed to education environments. Since it is generally agreed that it takes much longer for students (and the instructor) to get acquainted with the former, their adoption may be undesirable for a regular one-semester/quarter Introduction to AI undergraduate course. However, for graduate-level courses where a more advanced and focused approach is desired, it might be appropriate to concentrate more on the production-oriented systems, as opposed to the education-oriented ones. Therefore, the repository provides a balanced mix of resources in terms of required hardware/software platforms, acquisition cost, system complexity, and specific AI topics.

In terms of platform, we include systems that are available for as many platforms as possible. We believe that, the more platforms a system has been ported to, the stronger the indication that (a) the system is mature/stable enough in terms of functionality, (b) the system is supported by its developers, and (c) more educators and students could benefit from its inclusion in the repository. In some cases, however, we may deviate from this heuristic, due to exceptional reasons, such as uniqueness in a topic area, system development philosophy, and price range.

In terms of price, we include a balanced mix between freeware/shareware and commercial systems. As the reader may suspect, the fact that a system is free may have nothing to do with its quality or support commitment from its developer(s). For commercial systems, we decided to include systems costing less than $5,000, since we believe that very few schoools might consider investing more than this amount in a specialized educational resource for a single undergraduate course.

In terms of complexity, we provide a wide spectrum of systems, ranging from very simple ones that could be used for a single assignment, to more advanced ones that could be used for a sequence of assignments or a subsequent special topics course.

In terms of topic and classification of these resources, we use the following taxonomy: Expert Systems, Fuzzy Logic, Knowledge Representation, Logic and Reasoning, Natural Language Processing, Neural Networks, Robotics, Search and Game Playing, Vision, and Other. This taxonomy is not necessarily the only possible one, since there exist many opinions on which general topics constitute AI, and what are the names and scopes of these topics. Nevertheless, the above taxonomy is consistent what the majority of available textbooks and course syllabi that we examined.

4.2 Structure Of The Repository

The repository is structured as follows (see Figure 1):

- General resources, which include pointers to syllabi, papers, and available textbook materials.
- Core topics, which include: General Purpose, Search and Game Playing, Logic and Reasoning, and Knowledge Representation.
- Advanced topics, which include: Expert Systems, Fuzzy Logic, Natural Language Processing, Neural Networks, Robotics, Vision, and Other.

For each of the advanced topics, we have included the following components (see Figure 2, for the Expert Systems entry):

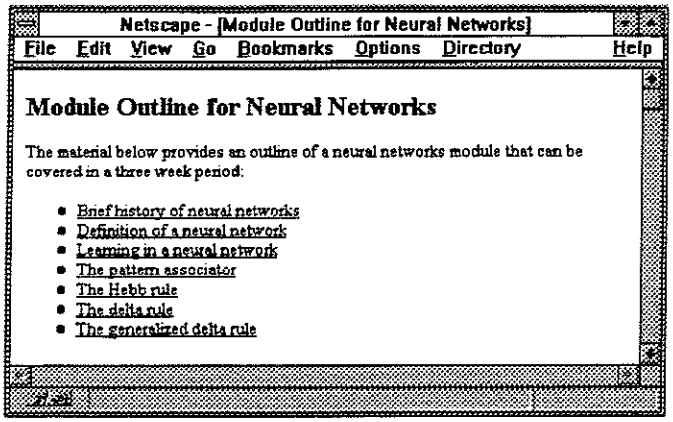


Figure 3. Neural Networks Module-Outline.

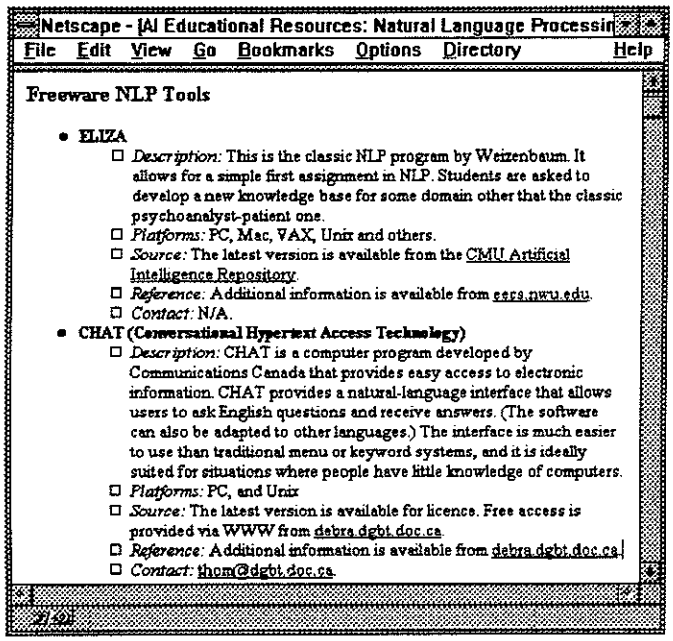


Figure 4. Excerpt from the Natural Language Processing Tools and Environments Entry.

- A module outline that may be used as a guideline for instructors who wish to include such a module in their AI course (see Figure 3, for an excerpt from the Neural Networks entry). It identifies a structured decomposition of the topic into specific subtopics, necessary prerequisites in terms of other AI and related knowledge/skills, and a suggestion on the time that may be spent on each for a reasonable coverage of the material.
- An on-line tutorial that may be made available to the students as a supplement to the class lectures on and textbook coverage of the given topic.
- Available resources for this topic including pointers to sample homework assignments, sample programming projects, and sample tests.
- Tools and environments which may be adopted for use in classroom activities (see Figure 4, for an excerpt of the Natural Language Processing entry). For each of the selected tools and environments, the following information is provided: (a) its acronym, name, and version (when available), (b) a short description of its functionality and goals, (c) supported platforms and system requirements, (d) a set of related universal resource locators (URLs) pointing to additional information, distribution sites, and demonstration sites (when available), and (e) information on how to contact system developers, maintainers, or distributors.

5 CONCLUSION

The AI Educational repository is an on-going effort to collect, organize, and make available resources to educators involved in teaching the Introduction to AI course. It is available at http://www.aaai.org/Education-Repository/education-repository.html.

The AI "winter" appears to be finally over. We expect to see extensive changes in the CS curriculum during the coming years, with increased coverage of artificial intelligence theory and applications. The repository described by this paper will be one of many tools that will play a major role in facilitating the process of course creation and modification for instructors of AI. This repository is under development and will continue to grow and evolve over time in response to the feedback, contributions, and requirements of the AI teaching community.

ACKNOWLEDGEMENTS

This work has been partially supported by US Department of Education/FIPSE Grant No. P116B40956. The authors would like to acknowledge the support of Barbara Grosz and Mike Hamilton of AAAI, and the influence/contributions of the following individuals: Robert Aiken, Joshi Anupam, Asuncion Gomez, Marti Hearst, Jonathan Hodgson, Giorgio Ingargiola, Deepak Kumar, George Luger, Lisa Meeden, Rebecca Skinner, Deborah Ann Vastola, Ellen Walker, Marilyn Walker, and Judith Wilson.

REFERENCES

[1] Aiken, R. The new hurrah: creating a fundamental rôle for artificial intelligence in the computing science curriculum. *Education & Computing* 7, (1991), 119 - 124.

[2] Dean, T., Allen, J., & Aloimonos, Y. (1995). *Artificial Intelligence – Theory and Practice*. Benjamin/Cummings Publishing Company.

[3] Ginsberg, M. (1993). *Essentials of Artificial Intelligence*. Morgan Kaufmann Publishers.

[4] Hearst, M. (1995). Introduction to the symposium. *SIGART Bulletin* 6(2), (1995), 3-5.

[5] Ingargiola, G., & Wilson, J.D. The introductory undergraduate AI course as observed on WWW. *SIGART Bulletin* 6(3), (1995), 2-6.

[6] Kantrowitz, M. CMU Artificial Intelligence Repository. *http://www.cs.cmu.edu/Web/Groups/AI/html/repository.html*

[7] LaRusch, M.R. Teaching artificial intelligence as the year 2000 approaches. *SIGCSE Bulletin* 25(1), (1993), 38-42.

[8] Luger, G.F., & Stubblefield, W.A. *Artificial Intelligence – Structures and Strategies for Complex Problem Solving*. Benjamin/Cummings Publishing Company.

[9] Manaris, B., Aiken, R., Koutsougeras, C., Munakata, T., & Valtorta, M. Report on the ACM CSC'95 AI panel: "Artificial intelligence: finally in the mainstream?". *SIGART Bulletin* 6(3), (1995), 7-11.

[10] Munakata, T. (Guest Ed.). Commercial and industrial AI. *Communications of the ACM* 37, 3, (1994).

[11] Noyes, J.L. Teaching AI: a breadth-first approach. *SIGCSE Bulletin* 25(1), (1993), 33-37.

[12] Port, O. Computers that think are almost here, *Business Week*, Jul. 17, 1995, 68-72.

[13] Rich, E., & Knight, K. (1991). *Artificial Intelligence – 2nd Edition*. McGraw-Hill, Inc.

[14] Russell, I. Neural Networks in the Undergraduate Curriculum. *Collegiate Microcomputer*, 9(1), 1991, 1-9.

[15] Russell, I., & Manaris, B. An AI Repository as a Course Development Resource. *Proceedings of ASEE-96 Conference*, June 1996 (to appear).

[16] Russell, S., & Norvig, P. (1995). *Artificial Intelligence – A Modern Approach*. Prentice Hall.

[17] Russell, S., & Norvig, P. A modern, agent-oriented approach to introductory artificial intelligence. SIGART Bulletin, 6(2), (1995), 24-26.

[18] Thomas, R. A consideration of some approaches to course organization. SIGART Bulletin, 6(2), (1995), 27-28.

[19] Tucker, A.B., et al. (1991). *Computing Curricula 1991 – Report of the ACM/IEEE-CS Joint Curriculum Task Force*. ACM Press.

USING ROBOTICS AS AN INTRODUCTION TO COMPUTER SCIENCE

Lisa Meeden
Computer Science Program
Swarthmore College
500 College Ave
Swarthmore, PA 19081
meeden@cs.swarthmore.edu
http://www.cs.swarthmore.edu/~meeden/

Abstract

This paper describes an introductory-level course for non-CS majors entitled *Building Intelligent Robots* that is part of the computer science curriculum at Swarthmore College. Founded in 1864, Swarthmore College is a small liberal arts school of about 1,300 students. The College seeks to provide its undergraduates with extensive laboratory experience, and this course offers an exciting hands-on setting in which to learn about computer science. The students develop the hardware and software expertise necessary to construct a robot from scratch and to program its behavior.

1 INTRODUCTION

The study of robotics offers a unique method for teaching students about some of the central concerns of computer science. When constructing a physical robot in conjunction with the programs to control it, students must contend directly with the important issues of:

- Hardware and software interaction;
- The design and implementation of software to solve real-world problems;
- Space complexity (in terms of the memory limitations of the robot's controller); and
- Time complexity (in terms of the desired speed of the robot's action decisions).

An added benefit of using robots is that students are highly motivated to master the material so that they can improve the performance of their creations.

This paper describes a robotics course, called *Building Intelligent Robots*, that can serve as a broad introduction to computer science as well as artificial intelligence. Since the course has no prerequisites, students have come from a wide variety of disciplines including economics, engineering, literature, mathematics, psychology, and religion. The course stresses both application and theory by dividing class time equally between laboratory sessions and discussion sessions. In laboratory, students work in teams to build small, mobile, Lego-based robots to perform tasks such as obstacle avoidance and light following.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/473 ©1996 FLAIRS

In discussion, students read and summarize selections from the primary literature on robot control and consider ways of applying the theoretical concepts to their practical concerns.

2 THE APPROACH

Not surprisingly, a majority of the students come to the course with wildly unrealistic impressions of the current state of the art in robotics. The balanced emphasis on theory and practice is intended to give the students a more accurate and well-rounded picture of what currently exists and what will be possible in the future of robotics. The objective of the discussion component of the course, is to survey the most important approaches in robotics to date and to understand their strengths and weaknesses. The objective of the laboratory component of the course is to allow the students to experience first-hand what is involved in trying to encode robot behavior into computer programs.

2.1 The Discussion Component

The discussions address how the problem of controlling robots to effectively accomplish tasks in dynamic, unpredictable environments has been solved. The emphasis is on the relatively recent reactive approaches, but the more traditional, deliberative approaches are also examined (though not implemented). Furthermore, an overarching theme is that adaptability is necessary if truly robust robots are to be achieved.

Each class meeting focuses on a single topic with a collection of articles from the primary research literature serving as the starting point for discussion. Particular students are designated as the discussion leaders for each reading and are expected to briefly summarize the material as well as guide the subsequent discussion by raising relevant questions.

Below are some examples of the types of topics that are covered and background materials used:

- Topic: Philosophical considerations
  Reading: Two contrasting views on the kinds of representations that are needed to create intelligent robots [2, 5].
- Topic: Building complexity bottom-up
  Reading: A series of thought experiments on how to build intelligent robots from very primitive to quite complex models[1].
- Topic: Reactive methods of robot control
  Reading: Selections of articles from a survey of recent reactive approaches [6].
- Topic: Current state of the art
  Reading: Summaries of the most recent robot competitions held at the AAAI conference [9, 10].

Using students as discussion leaders can be problematic, since it is difficult for them lead discussion well when they are novices in the field. Their tendency is to spend too much time reviewing the basic facts presented in the reading and not enough time focusing on the larger issues. To alleviate this problem, the student-led topics are delayed until the middle of the semester. The hope is that by then the students will have some foundation in the area to draw upon when deciding what aspects of the reading are worth discussing. It is also important as the instructor to be a very active participant in the discussions, correcting misunderstandings of the technical details as they arise and providing the appropriate context in which to place the research findings.

Despite these difficulties, student-led discussions enhance the course by providing the students incentive to read the material much more carefully than they would if the instructor were to present it in a standard lecture format. On the course evaluations, students were asked to rate the presentations given by their classmates from "not so good," "ok," "pretty good," to "great." and the average response was between "ok" and "pretty good." I have found that the presentations are much improved if the students are given a clear set of guidelines about what is expected of the discussion leaders.

2.2 The Laboratory Component

The laboratory exercises address the problem of actually building an intelligent (albeit simple) robot from scratch. There are two main steps involved: developing the hardware and then writing the software to control that hardware. The hardware should be robust and provide good driving control, while striking a balance between speed and torque. The software should encode a mapping from perceptions (the sensor readings) to actions (the motor settings) to accomplish a given task.

Typically the task only provides an abstract description of the desired mapping from perception to action. Much of the detailed specifications about how to behave in a particular situation is not given and may not be known. For example, a typical robotics task is to navigate from a starting point to a goal point. How will the robot recognize that it has arrived at the goal location? What obstacles can the robot expect to encounter along the way? If a map is available, how do positions on the map correspond to the robot's incoming sensor readings?

Due to the open-ended nature of robotics problems, a lengthy process of trial and error is often necessary to answer these types of questions clearly enough to develop a working algorithm. The students must repeatedly:

- Run their program on the robot and watch how well it succeeds;
- Analyze the failures and determine how they correspond to the program;
- Correct the problems in their program;

and continue in this cycle until the results are satisfactory. During this debugging period, the students learn a great deal about the interaction between the robot's sensors and its physical environment, and in turn must translate this knowledge back into their program. The programming skills needed are actually quite basic: sequencing, looping, and conditionals; the real work is in understanding the response of the robot's sensors to changes in the environment. Ultimately, the students should recognize that they, the programmers, are the bottleneck in this development process, and a much more efficient strategy would be to "design a behavior for the robot that would, itself, adapt to the environment it was in" [4, p 3]. In light of this realization, the latter third of the course focuses on learning and adaptation, introducing neural networks and genetic algorithms.

The laboratory exercises given each week build on the previous results; here are some typical kinds of assignments:

- Introduction to the programming language: a much simplified version of C, variable declaration, assignment, while loops, if-then-else.
- Introduction to the programming environment: editing programs, compiling programs, downloading programs from the computer to the robot controller, running programs on the robot.
- Introduction to the robot kits: sturdy design techniques, gear ratios, sensor responsiveness
- Robot base: construct a basic mobile robot
- Obstacle avoidance: add bumper sensors to robot base and update control software.
- Light seeking: add photo receptors to robot base and update control software.
- Combine obstacle avoidance and light seeking: test in a series of increasingly complex situations.

A key to success in this enterprise of constructing robots from scratch is that the students work in teams. As one student noted in the course evaluations, by working together they are able to accomplish more than they could do alone: "not everyone in the class came in with an intimate knowledge of programming, engineering, and structural design—this way we all have an opportunity to learn from each other instead of being frustrated by the gaps in our knowledge." Ideally there should be three members per team (two is also viable, but four or more is

too many). To encourage the students to experiment, each team should have unlimited access to a robot building station. A station consists of a computer and a robot construction kit. The robot kit should include motors, a variety of sensors, components for the robot base, and a controller.

There a number of options to consider when equipping a robot building station. For the robot kits, by far the best option is to use Lego resource sets with over 1,500 building components such as motors, battery packs, gears, and wheels. These sets are inexpensive, extensible, infinitely adaptable, and most importantly very accessible to the students. Sets of this kind have been used for the last five years at MIT for its extremely popular Robot Design Competition [3]. I supplement these sets with sensors (both from Lego and from other sources) and a controller. For the controller there are three good options; from simplest to most complex, they are the Mini board, the Handy board, and the 6.270 board (all designed by the MIT instructors of the design competition). In the *Building Intelligent Robots* course I have used the Mini board [8], which is easy for the students to master, but has a very limited 2K of program memory. The Mini board can be tethered to a computer, allowing the program to reside off-board and thus side stepping the memory limitations. The Handy board [7] is the newest controller, and it incorporates most of the more advanced features of the 6.270 board, such as a larger memory and an on-board battery, but in a much more compact and simplified form. Since the Handy board has more features than the Mini board and is approximately the same price, I plan to use the Handy board in future offerings of the course. The programming environment for the control boards can operate on either DOS or Unix platforms. Since low-end DOS machines are generally more affordable than Unix machines we have been using this platform.

2.3 The interaction of the two components

Whenever possible, the discussion component and the laboratory component are closely integrated to provide opportunities for cross-fertilization to occur. The theoretical material being discussed in class should lead the students to discover new methods for solving problems in the hands-on setting of the laboratory. Similarly the experience gained through robot construction should develop the intuitions the students need to begin critically analyzing the proposed theories presented in the readings.

3 CONCLUSION

Robots fascinate the typical undergraduate student, and we should use this interest to draw students into the computer science curriculum. Building a robot from start to finish is an intensely engrossing experience that motivates the students to learn about many of the less glamorous theoretical aspects of computer science. Based on student feedback this course has been quite successful at integrating theory and practice:

> We get to learn how to actually do something as opposed to learning about how it works.
>
> Being able to design completely from scratch has been extremely helpful.
>
> Although reading about things is interesting, actually having to apply what we've learned increases our retention rate tremendously. In addition, its much more involved to have to actually deal with the physical problems (like carpets!). Knowing theory is nice, but it just doesn't beat a good, solid working knowledge built upon experience.
>
> You learn so much more when you actually have to put theory into practice with your own hands.

References

[1] Valentino Braitenberg. *Vehicles: Experiments in Synthetic Psychology.* MIT Press, Cambridge, MA, 1984.

[2] Rodney Brooks. Intelligence without representation. *Aritificial Intelligence*, 47:139–159, 1991.

[3] Matt Domsch. Mit 6.270 lego robot design competition. World Wide Web, URL is http: // www / mit / edu:8001 / courses / 6.270 / home.html.

[4] Leslie Pack Kaelbling. *Learning in Embedded Systems.* MIT Press, Cambridge, MA, 1993.

[5] David Kirsh. Today the earwig, tomorrow man? *Aritificial Intelligence*, 47:161–184, 1991.

[6] Pattie Maes, editor. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back.* MIT Press, Cambridge, MA, 1990.

[7] Fred Martin. The handy board. World Wide Web, URL is http: // lcs.www.media.mit.edu : 80 / groups / el / Projects / handy-board /.

[8] Fred Martin. Mini board 2.0 technical reference. MIT Media Lab, Cambridge MA, 1994.

[9] I. Nourbakhsh, S. Morse, M. Balabanovic, E. Gat, R. Simmons, S. Goodridge, H. Potlapalli, D. Hinkle, K. Jung, and Van Vactor D. The winning robots from the 1993 robot competition. *AI Magazine*, 14(4):51–62, 1993.

[10] Reid Simmons. The 1994 AAAI robot competition and exhibition. *AI Magazine*, 16(2):19–30, 1994.

A Proposed Framework for Automating Software Testing

Ibrahim F. Imam

SRA International
4300 Fair Lakes Court, FS-5
Fairfax, VA 22033
email: iimam@verdi.iisd.sra.com

ABSTRACT

Software testing is a complex process that requires special expertise, tools and other resources. This paper introduces a framework for automating and optimizing the process of software testing. The proposed framework is a blackboard architecture with multiple agents. The framework consists of a software agents that provide heuristics and data about each phase of the software life cycle, a planning agent is responsible for generating a testing plan at a given phase, an adaptive agent is responsible for maintaining the information obtained from the software agent and supporting the planning agent, and an optimizing agent that eliminates irrelevant heuristics, data tuples, or attributes. The agent-to-agent communication is monitored by a control unit and information is transferred through the blackboard. The paper also introduces an application of an adaptive system for selecting the "best" tools for testing. The experiment shows the capability of the adaptive system to suit different testing situations.

Key words: machine learning, software testing, decision making, decision trees.

1. Introduction

Software testing is a multi-stage process that aims at discovering defects in software systems and determining whether or not these systems are reliable in different operational situations (Sommerville, 1994; Rine & Dandashi, 1994). Designing a standard plan for testing a software is extremely difficult and expensive. One of the main problems associated with the generation of a testing plan is that it should provide information to ensure the availability of hardware and software resources that are needed for the testing process. Using intelligent systems for designing software testing plans could be very successful and an effective approach (Anderson, 1993). Such an apprach can produce similar or better plans as those designed by an expert.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/478 ©1996 FLAIRS

Intelligent plan generation could be more effective if supported by an adaptive system to select the best resources for a given testing phase.

This paper introduces a framework for planning and optimizing the process of testing software systems. The proposed framework is designed as a blackboard architecture with a set of agents. Agents can be classified into two groups, software and intelligent. Software agents are responsible for managing data and heuristics associated with each testing phase (e.g., requirement, design, etc.). Intelligent Agents provide testing plans, optimize the information provided by the software agents, or adapt the testing plans, and search for the optimal preconditions of each step of the plan. The agent-to-agent communication is supervised by a control unit. Information is transferred from one agent to another through the blackboard.

The paper introduces also an application of an adaptive agent for learning knowledge describing the resources that are most suitable for testing a software. This is done by the learning system AQDT-2 (Michalski & Imam, 1994) for learning a task-oriented decision structure from decision rules or examples. A *decision structure* is an acyclic graph that describes the testing plan and it specifies an order of tests to be applied to a software system to arrive at a decision about that software. The nodes of the structure are assigned individual tests (which may correspond to testing different items of the software system during various phases of its development), the branches are assigned possible results of a test, and the leaves are assigned specific decision(s) and/or recommendations.

2. Brief Description of the Framework

The proposed framework is a blackboard architecture with multiple agents. A blackboard architecture consists of a control unit, blackboard, and knowledge sources. The control unit is responsible for all communications among the knowledge sources (agents), and between the user and the whole system. The blackboard is a structured memory that is shared by all agents, and managed by the control unit. The agents are systems that are responsible for performing certain

tasks or controlling other agents. Figure 1 shows a description of the proposed framework.

To explain how the blackboard works, consider the following scenario. Assume that the user acquires a less expensive way of testing the requirement of a given software. The control unit will interface with the adaptive agent to retrieve such information. The adaptive agent will acquire data and heuristics for testing the requirement of that software. The control unit will retrieve this information from the requirement agent (through the software agent). All transactions take place through the blackboard.

3. Illustration of the Framework

Consider the following example to explain the communication process among the different agents of the proposed framework. Assume that a user requested to test a software. The initial input to the framework is the software itself. The expected output is a plan describing the process of testing the given software. A testing plan consists of initial node (called starting node), final node (called the goal node), and one or more intermediate nodes. Each node in the plan represents a state of the testing process. To move from state to the next (from a node to the next in the plan), a testing procedure has to be performed. For each testing procedure, a set of tools and/or resources are needed to complete the testing process. To execute a testing procedure, all pre-processing requirements should be satisfied, and the required resources should be available.

Whenever a new software is to be tested, the control unit sends a signal to the software agent to acquire the necessary information about the given software. The software gathers information about the objectives of the software as well as other implementation and life-cycle aspects. The software agent stores these information in the blackboard and sends a signal to the control unit that it accomplished its job. The control unit sends a message to the optimizing agent to optimize the information and data associated with the software.

The optimizer runs generic search to determine relationships among different components of the software, and to remove irrelevant data, attributes, or other resources that are not needed for the current testing process. The optimizing agent updates the data and the information in the blackboard and sends a signal to the control unit when it finishes its job. The planner agent generates a plan for testing the given software using the optimized data and information produced by the optimizing agent. The adaptive agent assists the planner in building the testing plan. We consider that at each stage of the testing process there are a set of rules that control the testing phase. To pass from one node to another, these sets of rules should be executed and tested. The role of the adaptive agent is to optimize these rules or knowledge-base to speed up the process of testing these rules.

The adaptive agent, described here, transfers this knowledge-base into a task-oriented decision structure. One of the advantages of transferring these decision rules into a decision tree is to speeding the reasoning process. Therefore, instead of testing almost all conditions of the rules only one path from the root to the leaf is to be tested. The planner uses these task-oriented decision structures to speed up the process of generating a testing plan.

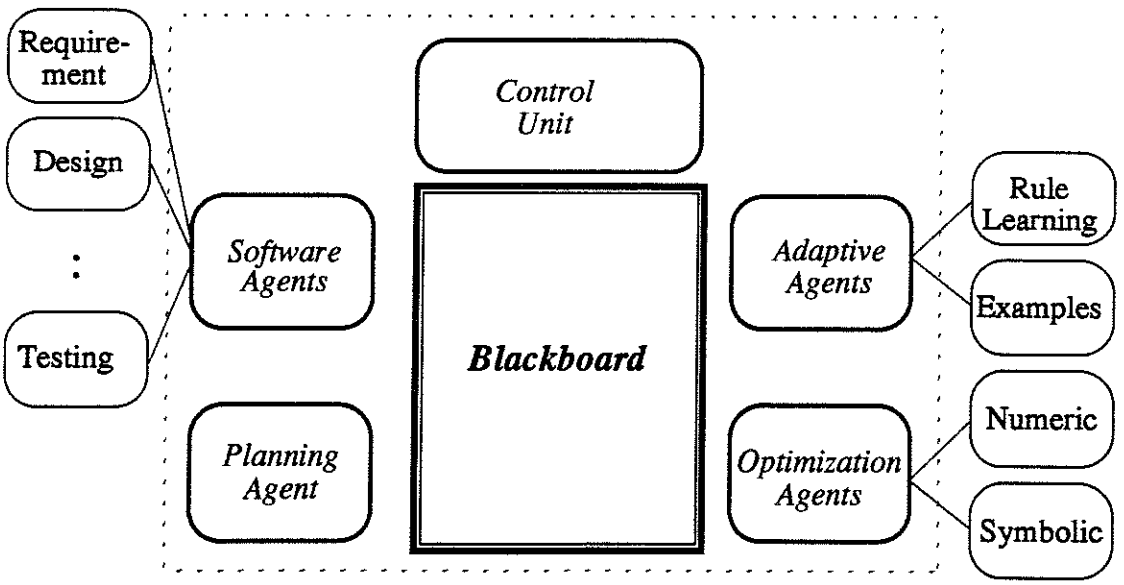


Figure 1: A framework for planning and optimizing a software testing plan.

Table 1: A description of factors affecting the selection of testing tools.

Attribute Name	Testing Factor	Possible Values 1	2	3	4
x1	Cost	Low	Average	High	Very_High
x2	Metric	Error_rate	Requirement	Sys_usage	Intractability
x3	Phases	Requirement	Design	Analysis	
x4	Execution	Automated	Manual	semi-automated	

4. An Application of an Adaptive Agent

The AQDT-2 system (Michalski & Imam, 1994) learns task-oriented decision structures from these decision rules or from examples. AQDT-2 is able not only to generate and store knowledge about testing, but also to use that knowledge effectively for decision making (e.g. selecting optimal resources).

For a given set of rules, AQDT-2 usually generates a large number of logically equivalent decision structures, which can differ in the test ordering. A decision class is the decision or conclusion that is of interest to the user. For example, possible decision classes of testing a software are "faulty" and "not-faulty". Another example of decision classes is a classification of the testing plans into "sufficient", "insufficient", and "incorrect".

Example: The following simple example illustrates how AQDT-2 can be used in selecting the optimal set of testing resources. Assume that there are three tools for testing a software: 1) modeling (T1); 2) checklist (T2); and 3) parallel-simulation (T3). Also, assume that there are four different factors that may affect the selection of any tool: 1) the cost of using the tool (x1); 2) the metrics that support the tool (x2); 3) the best phase for using the tool (x3); and 4) type of the tool (automated, semi-automated or manual) (x4). Table 1 shows these attributes and their possible values. Figure 2 shows a set of rules describing the conditions of using each tool.

When AQDT-2 learns the decision structure in Figure 3 from the decision rules in Figure 2. The decision structure in Figure 3 can be used in making decisions on which tools to be used for testing a given software. In that decision structure, when x1 (the cost of testing) is equal to 4 (very high), it is better to use the third tool (parallel-simulation).

Let us assume that it is very costly to know which metric support the required tools. In other words, suppose that we would like to select the best tools independent of the metric it supports (this information is indicated to AQDT-2 by assigning very high cost to attribute x2).

{T1 <= [x1=2] & [x2=2]
T1 <= [x1=3] & [x3=1 v 3] & [x4=1] }
{T2 <= [x1=1 v 2] & [x2=3 v 4]
T2 <= [x1=3] & [x3=1 v 2] & [x4=2] }
{T3 <= [x1=1] & [x2=1]
T3 <= [x1=4] & [x3=2 v 3] & [x4=3] }

Figure 2: Rules used for illustrating the method.

The algorithm assigned attribute x1 to the root of the tree. When x1 takes value 1 or 2, it is impossible to provide a specific decision without measuring attribute x2. For the value 1 of x1, the recommended tool can be either T2 or T3 (see the diagram in Figure 4a), and for the value 2 of x1, the recommended tool can be either T1 or T3. However, for other values of x1, one can make a specific decision after measuring attribute x4. Figure 4b shows another decision tree were the type of the tool, x4, was unknown.

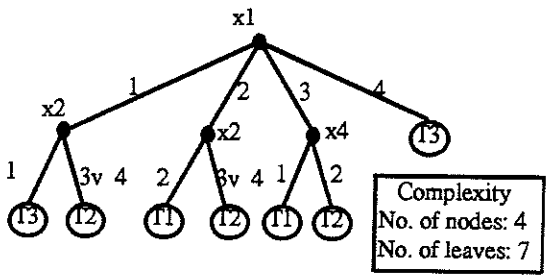
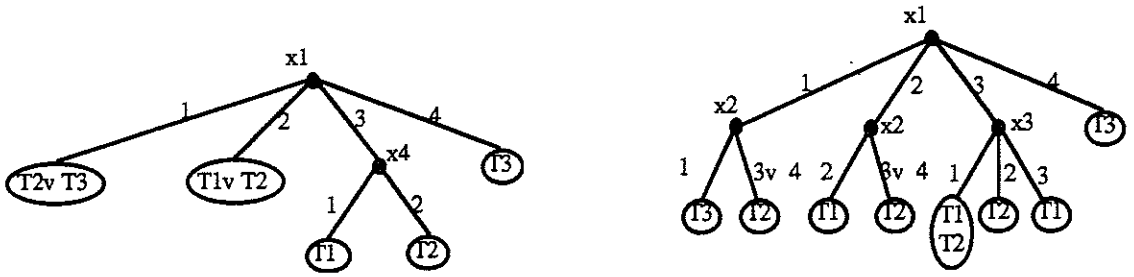


Figure 3: A decision structure learned from the rules in Figure 2.

It is clear that the given set of rules depend highly on amount of money can be spent. Let us now suppose that the cost attribute x1 cannot be measured. The algorithm selected x4 to be the root of the new decision tree. Figure 5 shows a decision structure obtained from the decision rules in Figure 2 given that the cost of x1 is very high. All leaves assigned to one class indicate situations in which it is possible to make a specific decision regardless of the cost of tools.



a) Ignoring the supporting metric b) Ignoring the type of the tool.

Figure 4: Decision trees learned for selecting testing tools without knowing some attribute values.

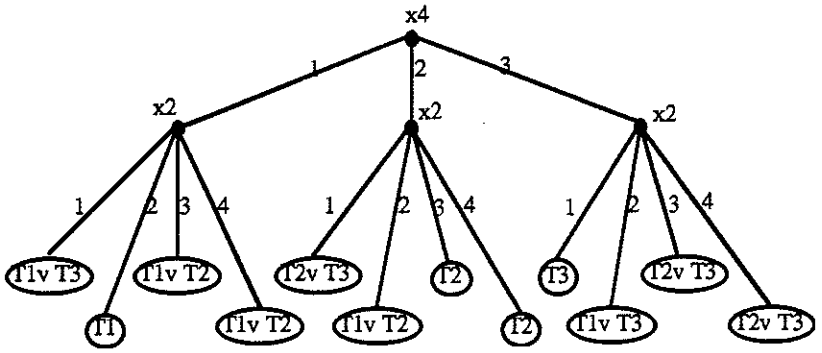


Figure 5: A decision tree learned without the cost attribute.

Conclusion

The paper presents a framework for automating and optimizing the testing process of software systems. The framework is a blackboard architecture with multiple agents. The paper introduces also an application of adaptive agent for determining the best set of tools for testing a given software. The experiment demonstrated the capabilities of the AQDT-2 system to provide knowledge, based on the given expertise, about the best tools to be used with different decision making situations.

Future work include incorporating a planning system to design plans for a given testing phase. Also, an interesting problem is to use some non-symbolic tools (e.g. genetic algorithm or neural network) for optimizing the heuristics and data provided by the software agent. The functions of such tools include simplifying the testing process by focusing only on relevant heuristics, attributes, or other information.

ACKNOWLEDGMENTS

The author thanks Srinivas Gutta for careful review of the paper.

This research was partially conducted in the Machine Learning and Inference Laboratory at George Mason University. The Lab's research is supported in part by the Advanced Research Projects Agency under the grant No. N00014-91-J-1854, administered by the Office of Naval Research, and under the grant No. F49620-92-J-0549, administered by the Air Force Office of Scientific Research, in part by the Office of Naval Research under grant No. N00014-91-J-1351, and in part by the National Science Foundation under grant No. IRI-9020266.

References

Anderson, J. S., "Automating Requirements Engineering Using Artificial Intelligence Planning Techniques", Ph.D. dissertation, CIS-TR-93-28, University of Oregon, 1993.

Michalski, R.S. and Imam, I.F., "Learning Problem-Oriented Decision Structures from Decision Rules: The AQDT-2 System," *in Lecture Notes in Artificial Intelligence, Methodology for Intelligent Systems, No. 869, pp. 416-426, Ras, Z.W. & Zemankova, M. (Eds.),* Springer Verlag, 1994.

Rine, D. and Dandashi, F., "A metrics suite for object oriented software development", Proceedings of the International Conference on Technologies for Object Oriented Languages and Systems, 1994.

Sommerville, I. "Introduction to Software Engineering Principles", 4th edition, Addison-Wesley Publishers, 1994.

EFFICIENTLY VERIFYING AND USING A LARGE CLASS OF HIGH-LEVEL SIMPLIFICATION RULES IN AUTOMATED REASONING SYSTEMS

David E. Leasure
Texas A&M University-Corpus Christi
leasure@tamucc.edu

Abstract

The problem of applying meta-level reasoning is attacked by recognizing that a large number of meta-lemmas used in automated theorem provers can be reformultaed at the object level by selective application of associative-identity matching. A method is presented that improves dramatically on the time to define, time to prove, and time to understand high-level lemmas. The method is simple and pervasively used in Schemata, and it appears to require few changes to NQTHM to achieve the same effect. Empirical evidence suggests that many automated reasoning systems could be more effectively built by incorporating high-level simplification laws.

1. INTRODUCTION

When proving theorems, human reasoners use a variety of generalizations of theorems to solve problems, One example is the "cancel like quantities in equal sums" rule. For example,

$3x + 2y + 7z = w + 2y + u + 3x$

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/482 ©1996 FLAIRS

simplifies easily using this rule. Automated theorem provers can perform similar simplifications using the associative, commutative, and inverse lemmas, but it is desirable for efficiency and directness of proof to directly incorporate such rules into automated reasoners. The question is how to do so in a fashion that is both sound and expedient. Boyer and Moore (1981) point out that such rules can be viewed as meta-theorems. For example, if two sums are equal, then each of the sums can be treated like a list, and syntactically equal terms from each list may be removed. Boyer and Moore (1981) describe an extension to NQTHM that treats the theorem prover as a definition of meaning for meta-level expressions. Employing this definition, a meta-level operation can be proven to both produce syntactically legal expressions and to be correct with respect to meaning as defined by the interpreter. Proofs done in this manner, however, are somewhat involved and can be time-consuming.

This paper describes an alternative approach that more quickly proves, allows easier devlopment, and reduces code volume by using a large class of high-level simplification rules. It begins in Section 2 by detailing the meta-level approach. Our approach is described in

Section 3, including a method of establishing correctness of lemmas. and empirical evidence for the pervasiveness of the class of rules covered by our approach. We conclude in Section 4.

2. THE META-LEVEL REASONING APPROACH

A meta-level level lemma is one that treats expressions in a proof as data. For example, the object-level expression (+ x y 2) may be treated at the meta-level as a list, (list '+ 'x 'y 2). Human reasoners make heavy use of meta-knowledge when solving problems. For example, the strategy "when solving equalities between sums, cancel like quantities" is a meta-lemma that is justified by the underlying laws of arithmetic. Other examples of meta-knowledge are "collect like terms" and "cancel all inverses". A general-purpose meta-knowledge facility that allows arbitrary functions to be introduced into the verification system NQTHM as meta-operators is described in Boyer and Moore (1981).

In NQTHM, meta-knowledge comes from the user in the form of meta-functions (LISP code) that manipulate object level terms as data. Meta-functions are incorporated into NQTHM after they have been proven correct. The proof of correctness for a particular *meta-function* in NQTHM is performed by proving the following conjecture:

$$\forall x A\ (\text{term? } x) \rightarrow \quad (1)$$
$$((\text{term? } (\textit{meta-function}\ x))$$
$$\wedge\ (\text{mg } x\ A)=(\text{mg } (\textit{meta-function}\ x)\ A)).$$

In other words if NQTHM can prove that for all objects x and interpretations A (represented by association lists between symbols and values), if x is a syntactically legal term, then the meta-function applied to x is also a legal term, and the meaning (mg) of x and the meaning of the result of the meta-function applied to x are equal. This method is still used in NQTHM, unchanged in over ten years of continuous development.

A large problem with this approach is the difficulty of specifying the theorem and the mechanism for proving it correct. The meta-theorem for cancellation mentioned above took Boyer and Moore 2 days to specify, was made up of approximately 50 lines of LISP code representing the cancellation law, and required 7 minutes of CPU time to prove correct. The large amount of CPU time (the age of the system, notwithstanding) is due to the necessity of working through both the large amount of lisp code and the large size of the meaning function itself, which is essentially a LISP function representing the entire NQTHM and all of its knowledge to that point. In general, this approach is useful, but in a large subclass of meta-theorems, also unnecessary, as is described in the next section.

3. SEGMENT VARIABLES: SELECTIVE APPLICATION OF ASSOCIATIVE-COMMUTATIVE MATCHING

As Boyer and Moore (1981) points out, the cancellation law mentioned above can be implemented using associative and commutative matching. AC-matching extends traditional syntactic matching by allowing terms of associative commutative operators to be matched using asso-

ciative and commutative properties. For example, syntactic matching does not match (+ x (+ y z)) with (+ (+ x z) y), but by re-arranging during the matching process (+ (+ x z) y) to (+ x (+ z y)) using the associative property and then to (+ x (+ y z)) using the commutative property, the two terms are matched. If these two terms appeared in an equality, then AC-matching could be used to reduce the equality to true.

Unfortunately, AC-matching is expensive, and its unrestricted use would greatly slow any system employing it. Furthermore, not all meta-functions we might like to employ can be represented using AC-matching. Given these two problems, what choice is there but the approach of Boyer and Moore (1981)? The solution lies in selective application of not AC-matching, but AI-, or associative-identity-matching, with explicit commutative matching specified when required. In spite of not being a general approach to meta-functions, in practice, a large number of what would otherwise be meta-functions can be handled at the object level with this approach.

3.1 Using Segment Variables

The language Schemata (Araya 1990) incorporates a matching operator called a segment, and written "...". When attached to a variable, as in "...x", it signifies to the matcher that 0 or more terms should be matched with x. For example, the pattern "(= (+ ...a x ...b) (+ ...c x ...d))" matches with "(= (+ p q r) (+ s p t))" using the following bindings: ...a=(), ...b=(q r), ...c=(s), and ...d=(t). The cancellation law, as opposed to the 50 lines of LISP code in the meta-level approach, is represented directly as the following definition, where the left-hand expression is matched and rewritten to the right-hand expression.

```
(define (= (+ ...a x ...b) (+ ...c x ...d))
        (= (+...a ...b) (+ ...c ...d))).        (2)
```

Using this definition, "(= (+ p q r) (+ s p t))" is re-written to "(= (+ q r) (+ s t))".

To say that segment matching is associative-identity-matching is motivated by the representation of the above patterns in append-normal-form. For example, treating the pattern "(+ ...a x ...b)" as "(append (+) (append a (append (x) b)))" allows it to match with the data "(+ p q r)" when it is represented as "(append (+) (append () (append (p) (q r))))". This process of translation to an *append-form* does not happen in the implementation of the matcher, but is useful for recognition of the mathematical properties involved. The append operation is associative and has the empty list () as its identity element, and thus forms a semi-group with identity, also known as a monoid, in algebraic terms.

3.2 Proving Lemmas That Use Segment Variables

The proof technique employed for verifying the correctness of lemmas such as (2) consists of defining varyadic operators, such as +, in terms of binary operations applied to lists, and treating segment variables as syntactic shorthand for an application of the defined operation to a list. For example, first we define + in terms of $+_2$ (binary plus).

```
(define (+ L)                                        (3)
  (if (not (list? L))
   0
    (if (null? L)
      0
      (if (null? (cdr L))
        (car L)
        (+2 (car L) (+ (cdr L)))))))
```

Definition (3) may now be used in the proof of lemmas containing segment variables. The A proof of (2) and any lemma like it is performed by first translating it to append-normal-form and then proving it through the normal verification mechanism. In other words, we translate (2) to the following, which we then prove:

```
(define (=  (+ (append a (cons x b)))
            (+ (append c (cons x d)))
        (=  (+ (append a b)
               (+ (append c d))))).
```

Although append-normal-form is used during verification, it is not used during application of the lemma to subsequent problems.

3.3 Empirical Evidence For Segment Variables

Three substantial applications have been implemented in L, a sequent calculus theorem prover for arithmetic (37 lemmas), an algebraic simplifier (105 lemmas), and a modal reasoner (418 lemmas). We call them substantial because an equivalent implementation in LISP would require approximately 10-20 times as many lines of code as lemmas. In the sequent calculus system 34 (92%) of the lemmas used segment variables; in the algebraic simplifier 51 (49%) of the lemmas used segment variables, and in the modal logic reasoner 265 (63%) of the lemmas used segment variables for a total of 350 out of 560 lemmas.

3.4 Comparing the Approaches

A direct empirical comparison with Boyer and Moore (1981) is not possible without implementation of both approaches in a single system, but is suggested by the numbers in Section 3.3, since the meta-functions would have to be coded in LISP. In the case of the cancellation law (2), the ratio of LISP to lemma-with-segment-variables is roughly 50 to 1. This ratio not only has an impact on the time to develop (5 minutes versus 2 days), but also on the perspicuity, or directness of purpose, in each of the two approaches; the one line law being preferable. At verification time, our experience with our own verification system suggests a two-orders of magnitude difference in time-to-verify, with the meta-level approach taking longer. On the other hand, the use of the lemmas in Schemata should be expected to run equivalently well to meta-functions in NQTHM, since both perform search, although fine tuning of the meta-functions may allow a performance advantage.

4. CONCLUSIONS

We have defined a method that improves dramatically on the time to define, time to prove, and time to understand high-level lemmas. The method is simple and pervasively used in Schemata, and it appears to require few changes to NQTHM to achieve the same effect. Empirical evidence suggests that many automated reasoning systems could be more effectively built by incorporating high-level simplification laws. The same evidence shows that a large percentage of what would otherwise be meta-level

lemmas in the approach of Boyer and Moore (1981) are more simply treated at the object level following the guidelines presented here.

References

C. Araya. 1990.*SCHEMATA – A Language for Deduction and Its Application in Nonmonotonic Reasoning.* Dissertation. Computer Science, University of Kansas.

R. S. Boyer and J. S. Moore. 1981. "Metafunctions: proving them correct and using them efficiently as new proof procedures." *The Correctness Problem in Computer Science*. R. S. Boyer and J. S. Moore, Eds. Academic Press.

A Toolset To Support The Formal Specification of AI Systems

Richard Hibberd, Jawed Siddiqi[1], Ian Morrey[1], Graham Buckberry[2]
The Nottingham Trent University
Burton Street, Nottingham, NG1 4BU, UK
Email: rbh@doc.ntu.ac.uk

[1] School of Computing and Management Sciences, Sheffield Hallam University, UK

[2] GPT Limited, Beeston, Nottingham, UK

Abstract: AI developments have been based on an exploratory approach whereas software engineering approaches can be characterised as specification based. We present an integrated toolset that accommodates and encourages a mixed mode development thereby maximising the strengths and minimising the weaknesses of both approaches. Moreover, the interactive nature of the toolset provides an effective means for both the user and developer to jointly validate the adequacy of the developing system.

Keywords: formal specification, exploratory programming, validation.

1. Introduction

Artificial intelligence practitioners have advocated an approach known as "exploratory programming", which is based on the idea of developing an outline informal specification from which an initial implementation is developed and repeatedly refined through exposure to user comment until an adequate system has been constructed. This is clearly different from a traditional life cycle approach. One of the key characteristics of this approach is the use of high level programming languages such as LISP and Prolog which enable rapid systems iterations to be carried out. Exploratory programming has been most effectively used to develop AI systems where it is difficult (or impossible) to establish the requirements of the system because the problem to be solved is ill-formed. It is less appropriate for large systems that have a long lifetime. An important difference between exploratory programming and the life cycle based approach is in the verification and validation phases. Since verification is only meaningful when a program is compared to its specification, clearly verification for a system developed using an exploratory approach is impossible. Moreover, the validation process is an attempt to demonstrate the adequacy of the program, that is, its conformance with respect to some user perception rather than a specification.

In the last decade there has been an increase in the use of formal methods for the construction of software systems [7]. In particular, formal methods known as model-based approaches such as those using Z and VDM have been used for several industrial applications [4]. The aim of formal methods is "to establish a mathematical foundation for software development in two crucial and related areas where precision and rigour are of the utmost importance: specification and verification" [12].

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/487 ©1996 FLAIRS

This is achieved through the provision of a framework for the development of provably correct programs (i.e. programs which will not deviate from their intended behaviour when executed). Proponents of this approach claim it has advantages, which include: the increased confidence in the reliability of the parts which have had their correctness established; freedom from the inadequacies and limitations of testing; and perhaps most significantly, the need for corrective maintenance is diminished.

This approach might appear considerably removed from the practicality and exigencies of everyday software development, but the method can be seen as a variant of the traditional life cycle with the following stages :

- requirements analysis
- formal specification
- specification verification and design verification

The first stage delivers a set of requirements, developed in the traditional way, which we assume are expressed informally in a requirements definition document. The requirements definition is then transformed using formal notations, such as Z and VDM, based on predicate calculus and set theory to arrive at a formal specification. At this stage this specification can be checked for correct syntax and internal consistency. However, it cannot be tested for adequacy because this involves validating the specification - that is, demonstrating the properties of the system to the user. The next stage involves refining the abstract specification into a concrete design and verifying the design. Several approaches have been developed [10].

Both approaches have strengths and weaknesses; neither is completely satisfactory for all circumstances [3][8]. It might be argued that formal methods should be used in situations where reliability is a high priority, and the requirements of the system are well understood, whereas exploratory programming should be used where the requirements of the system cannot be easily established and the primary aim is to understand what they are. However, we feel that this situation can be improved, and this paper reports on a toolset that supports a hybrid approach to the construction and validation of systems. Section 2 characterises the development approach and the toolset. Section 3 details the application of the tool to a chosen case study. Finally, in Section 4, the implications of the mixed approach of formal specification and rapid prototyping are briefly discussed.

2. Features of the toolset

In this section we look at two initial prototypes of tools that will form part of the environment to support the development of Z specifications. The tools are WiZE, an editor and syntax analyser, and ZAL, a tool for prototyping Z specifications. WiZE is a graphical, window-based editor package and built-in syntax analyser for writing formal specifications in Z. The aim is to allow specifiers to enter Z specifications interactively and have their syntax checked for correctness automatically.

Z is a specification language based on the typed set theory, which was introduced by Jean-Raymond Abrial in 1979, and is now being developed by the Programming Research Group at Oxford University. The evolution of a Z standard [13] has facilitated the development of computer-based tools that assist in the writing and manipulation of Z specifications. In what follows, we assume a reading knowledge of Z. Those unfamiliar with the notation are referred to [11]. We also assume a reading knowledge of the LISP programming language.

WiZE understands the format of the Z notation and provides utilities for automatically manipulating and formatting schema box graphics directly on the editor screen. When invoked, the built-in syntax analyser identifies any Z notation syntax errors in the file currently being edited and highlights their location in the file. The syntax analyser is designed to conform to the standard Z notation syntax [13], and provides appropriate error messages if errors are discovered. In addition to the low-level editor functions found on most full screen editors, WiZE supports features specific to the Z notation, such as schema graphics and axiomatic and generic definitions. The original grammar has also been extended to allow for additions to the notation including text strings, schema renaming and comments. The syntax analyser includes a simple semantic analyser which implements variable scoping rules and usage rules but which does not currently include a full type checker. Schema inclusion is supported, automatically accounting for decoration, as well as the standard Z Δ and Ξ naming conventions.

The Z Animator in LISP (ZAL) "executes" a Z specification, though not directly. The specification must be transformed into an extended LISP format which is directly executable. The transformation is largely mechanical for that subset of Z which has been implemented. The ZAL (extended LISP) code is arguably only marginally less abstract than the equivalent Z; this is an important aspect of the approach, as will be demonstrated. The most useful view is of ZAL as an animator that implements Z constructs rather than any particular specification; the subset of Z that can be animated is that for which equivalent constructs have been developed in ZAL. Our primary concern is to retain a high level of abstraction; one way of achieving this to make the correspondence between the Z notation and the prototyping notation as close as possible. Having achieved the functionality suggested by [6], development of the ZAL system continues, supporting an increasing subset of Z

The benefits of early animation have been identified by [2] to include

- executable components are available much earlier than in the traditional life-cycle, allowing earlier (less expensive) detection and correction of problems;
- requirements that are unclear can be clarified by interaction with the specification via the animation;
- execution of the specification supplements inspection and formal reasoning as a means of validation.

We would add to the above list the shared ownership of a (better) specification, as a consequence of the enhanced accessibility of the specification to the sponsor.

The originally perceived place of the tools in the traditional development cycle was in the area of requirements capture (and specification). In this life cycle approach we intend that each evolution of the specification that is synthesised by the requirements engineer/specifier should be formally recorded using Z, and that specification should then be animated, i.e. translated into ZAL and executed with the sponsor as part of the validation process. The specification can then be reasoned with and expected behaviour confirmed. However it was seen that exploratory programming could be supported; this is achieved by using an alternative "route", but one that still yielded a formal specification. Both routes are described below in Figure 1.

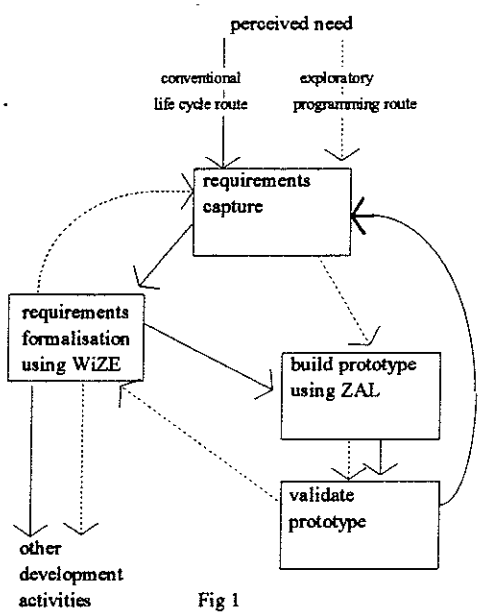


Fig 1

The system maximises the strength of a life cycle approach based on formal specifications, in that it results in a document for developers that is a clear, concise and unambiguous statement of the system requirements; and the strength of an exploratory approach, in that it provides feedback through an accessible prototype. It also minimises the weaknesses of an exploratory approach which often results in a poorly structured ad-hoc solution, and of formal specifications, which can result in a document incomprehensible to users.

The system can therefore support development in either mode whether it be specification-led as is advocated traditionally or prototype-led as is advocated in the exploratory approach. The latter approach allows the development and maintenance of a formal specification of systems for which it might have thought inappropriate.

In the next section, we demonstrate a small case study in which a mixed development approach was used.

3. A Case Study Using ZAL

Prototyping a given Z specification in ZAL simply involves rewriting each schema in the ZAL notation, and executing the prototype with test-case data in the ZAL shell. Alternatively, a Z specification can be derived from a ZAL program which has been incrementally designed, refined and validated using an exploratory approach. We shall illustrate both of these approaches by specifying a system for classifying mammals according to their habitat. First, the basic types:

[mammal, habitat]

mammal is the set of unique mammal names, and **habitat** the set of unique habitat names. The model for the database will be a partial function (called **classify**) mapping habitats to sets of mammals, with the additional requirement that a given mammal can be found in one and only one habitat. The system state and data invariant can thus be written as follows:

___world_____

classify : habitat ⇸ ℙ mammal

∀ i,j : habitat | i ∈ dom classify ∧ j ∈ dom classify •
i ≠ j ⇒ classify(i) ∩ classify(j) = {}

In transforming the Z schema into ZAL, each expression is rewritten from infix to prefix, and Z operators are replaced by their ZAL counterparts:

```
(schema world
 :predicate
  (forall i (dom classify)
   (forall j (dom classify)
    (imply
     (not (eqz i j))
      (eqz (inter (applyz classify i)
                  (applyz classify j))
           {}) ))))
```

forall, dom, imply, eqz, inter and **applyz** are ZAL constructs. **forall, dom** and **imply** have the obvious interpretation. **eqz** tests for the equality of its arguments if they are bound, but can also bind an unbound argument to the value of the other, bound argument. **inter** is ZAL's equivalent of the set intersection operator, and **applyz** applies a function, in this case **classify**, to an argument.

Once a schema has been translated into ZAL, the prototype produced can be used to demonstrate its validity, and to test assertions about the system behaviour.
A user can interact with the ZAL system at the command line or via windowed menus and dialogues; the command line responses are more concise and will generally be shown here, though Figure 2 shows a typical "windowed" interaction. After entering the interactive ZAL environment (indicated below by the "ZAL" prompt), we first create global instances of a world of mammals and habitats using the ZAL **make** construct:

```
ZAL: (make classify
       [#('desert {'camel 'addax 'kangaroo-rat})
        #('salt-water {'blue-whale 'dolphin 'walrus})
        #('grassland {'prairie-dog 'bison 'anteater})
        #('fresh-water {'beaver 'muskrat})
        #('tropical-forest {'bongo 'fruit-bat })])
```

```
ZAL: (make habitat
       {'desert 'salt-water 'grassland 'fresh-water 'tropical-forest})
```

The invariant included in the system state can be validated for the given instances of **classify** and **habitat** by invoking the ZAL version:

```
ZAL: (execute world)
True
```

The system response *True* indicates that the invariant is satisfied by the current values of **classify** and **habitat**.

The **who-lives** operation reports which mammals live in a given (known) habitat:

___who-lives_____

classify, classify' : habitat ⇸ ℙ mammal
h? : habitat
m! : ℙ mammal

---

h? ∈ dom classify
m! = classify(h?)
classify = classify'

In ZAL:

```
(schema who-lives
  :? h?
  :! m!
  :show (m! classify classify')
  :predicate
   (and
    (mem h? (dom classify))
    (eqz m! (applyz classify h?))
    (eqz classify classify')))
```

:? declares the schema inputs, and **:show** flags those components of the state which are to be displayed after execution, though **:show** is not used in the window environment, as the objects are available for browsing via the various ZAL "tools". **mem** implements the set membership operator in ZAL. We now validate the **who-lives** operation by testing it firstly with a known habitat:

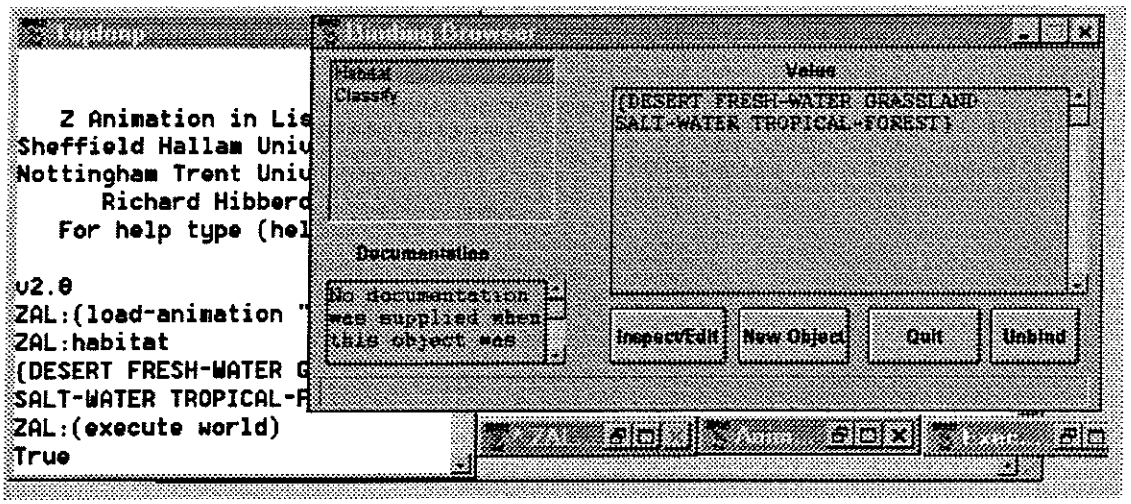


Fig 2

ZAL: (execute who-lives 'grassland)
M! {ANTEATER BISON PRAIRIE-DOG}
CLASSIFY [(DESERT {ADDAX CAMEL KANGAROO-RAT})
(FRESH-WATER {BEAVER MUSKRAT})
(GRASSLAND {ANTEATER BISON PRAIRIE-DOG})
(SALT-WATER {BLUE-WHALE DOLPHIN WALRUS})
(TROPICAL-FOREST {BONGO FRUIT-BAT})]
CLASSIFY' UNCHANGED
True

and secondly with a habitat which is not part of our database:
ZAL: (execute who-lives 'prairie)
M! UNCHANGED
CLASSIFY [(DESERT {ADDAX CAMEL KANGAROO-RAT})
(FRESH-WATER {BEAVER MUSKRAT})
(GRASSLAND {ANTEATER BISON PRAIRIE-DOG})
(SALT-WATER {BLUE-WHALE DOLPHIN WALRUS})
(TROPICAL-FOREST {BONGO FRUIT-BAT})]
CLASSIFY' UNCHANGED
False
M! is not instantiated with a new value because a condition of **who-lives** was not satisfied and the execution "fails", returning *False*.

A typical query of the system is that which determines whether a given mammal lives in a given habitat. Rather than attempting to write the specification for this operation directly in Z, we shall take an exploratory approach by first writing a ZAL schema which tries to capture its expected behaviour, and then experimenting with it:

```
(schema lives
  :? m?
  :! h!
  :show (rep! h! classify classify')
  :predicate
    (and
     (exist h (dom classify)
      (and  (mem m? (applyz classify h))
            (eqz h h!)))
     (eqz rep! 'confirmed)
     (eqz classify classify')))
```

The **exist** construct implements existential quantification in ZAL.

To validate the **lives** operation, we test it by choosing test cases as follows:

ZAL: (execute lives 'fruit-bat)
REP! CONFIRMED
H! TROPICAL-FOREST
CLASSIFY [(DESERT {ADDAX CAMEL KANGAROO-RAT})
(FRESH-WATER {BEAVER MUSKRAT})
(GRASSLAND {ANTEATER BISON PRAIRIE-DOG})
(SALT-WATER {BLUE-WHALE DOLPHIN WALRUS})
(TROPICAL-FOREST {BONGO FRUIT-BAT})]
CLASSIFY' UNCHANGED
True

The test-case confirms our expectations, so we can now transform the ZAL version of **lives** into an equivalent Z specification:

report ::= confirmed

lives

classify, classify' : habitat ⇸ ℙ mammal
m? : mammal
h! : habitat
rep! : report

∃ h : habitat | h ∈ dom classify • m? ∈ classify(h) ∧ h = h!
rep! = confirmed
classify = classify'

We now specify the operation which adds a new mammal and its habitat to the database, on the assumption that the habitat is known to the system. Again, we shall take an exploratory approach, experimenting with an initial version written in ZAL:

```
(schema new-mammal
 :? (h? m?)
 :show (classify classify')
 :predicate
 (and
  (mem h? (dom classify))
  (eqz classify'
  (override classify
    [#(h? (unionz (applyz classify h?)
                  (m?)))]
 )))))
```

override and **unionz** are ZAL constructs with the obvious interpretation. **[#(...) #(...) ... #(...)]** constructs a mapping by enumerating its maplets.

Executing **new-mammal** with test-case data produces the following output:

ZAL: (execute new-mammal 'desert 'rat)
CLASSIFY [(DESERT {ADDAX CAMEL KANGAROO-RAT})
(FRESH-WATER {BEAVER MUSKRAT})
(GRASSLAND {ANTEATER BISON PRAIRIE-DOG})
(SALT-WATER {BLUE-WHALE DOLPHIN WALRUS})
(TROPICAL-FOREST {BONGO FRUIT-BAT})]
CLASSIFY' [(DESERT {ADDAX CAMEL KANGAROO-RAT RAT})
(FRESH-WATER {BEAVER MUSKRAT})
(GRASSLAND {ANTEATER BISON PRAIRIE-DOG})
(SALT-WATER {BLUE-WHALE DOLPHIN WALRUS})
(TROPICAL-FOREST {BONGO FRUIT-BAT})]

which confirms that the new mammal is correctly added to the database. However, a further test case reveals an error in the definition of **new-mammal**:

ZAL: (execute new-mammal 'grassland 'addax)
CLASSIFY [(DESERT {ADDAX CAMEL KANGAROO-RAT})
(FRESH-WATER {BEAVER MUSKRAT})
(GRASSLAND {ANTEATER BISON PRAIRIE-DOG})
(SALT-WATER {BLUE-WHALE DOLPHIN WALRUS})
(TROPICAL-FOREST {BONGO FRUIT-BAT})]
CLASSIFY' [(DESERT {ADDAX CAMEL KANGAROO-RAT})
(FRESH-WATER {BEAVER MUSKRAT})
(GRASSLAND {ADDAX ANTEATER BISON PRAIRIE-DOG})
(SALT-WATER {BLUE-WHALE DOLPHIN WALRUS})
(TROPICAL-FOREST {BONGO FRUIT-BAT})]

The addax has been added to the grassland habitat despite the fact that it is already included as a desert inhabitant. This is in conflict with the data invariant. We thus revisit **new-mammal** and add the correcting precondition

```
(not-mem m? (union-dis (ran classify)))
```

to its ZAL definition. **not-mem**, **union-dis** and **ran** are ZAL constructs. **union-dis** is the ZAL version of the distributed set union operator. Now **new-mammal** behaves as required:
ZAL: (execute new-mammal 'grassland 'addax)

CLASSIFY [(DESERT {ADDAX CAMEL KANGAROO-RAT})
(FRESH-WATER {BEAVER MUSKRAT})
(GRASSLAND {ANTEATER BISON PRAIRIE-DOG})
(SALT-WATER {BLUE-WHALE DOLPHIN WALRUS})
(TROPICAL-FOREST {BONGO FRUIT-BAT})]
CLASSIFY' UNCHANGED

This validated and corrected version of new-mammal now suggests the following Z schema:

new-mammal

classify, classify' : habitat ⇸ ℙ mammal
h? : habitat
m? : mammal

---

h? ∈ dom classify
m? ∉ ⋃ (ran classify)
classify' = classify ⊕ { h? → classify(h?) ∪ {m?} }

The above testing of the prototype has demonstrated the validity of the required operations of the system. If the testing reveals errors or inconsistencies, the specification can be revisited, corrected and revalidated. Thus an iterative (and exploratory) specify-validate-respecify cycle is suggested.

4. Concluding Discussion

The application of the ZAL system to the chosen case study has shown that it is possible to take a Z specification and translate it through a series of simple transformations into an equivalent executable form and then execute this in order to demonstrate the functionality of the intended system. The case study chosen was for the sake of brevity a small one, but nevertheless it illustrates the characteristics of our prototyping environment. We have constructed a library of case studies of Z specifications that have been prototyped and demonstrated. Doing this has produced two benefits additional to the production of the library. First, the chosen case studies have formed undergraduate and postgraduate projects involving the rapid prototyping of Z specifications; they have, therefore, provided a valuable educational tool to increase our students' awareness of formal methods. Second, the case studies have also been an effective vehicle for testing and providing valuable feedback on the ease of use, robustness, usefulness etc. of the system. Some of the case studies prototyped are relatively simple and other non-trivial ones include the specification of the telephone network [9].

The development process advocated in the life cycle approach is one possible mode of development. It corresponds to a view in which "one first completely specifies a system in a formal language at a high level of abstraction in an implementation free manner. Then, as a separate phase, the implementation issues are considered and a program realising the specification is produced". It stresses the primacy of a declarative approach over a procedural approach; in this respect it is diametrically opposite to the exploratory approach favoured by AI programmers [14]. More generally, Abelson and Sussman [1] convincingly argue that:

> *"The computer revolution is a revolution in the way we think and in the way we express what we think. The essence of this change is the emergence of what might be called procedural epistemology - the study of the structure of knowledge from an imperative point of view, as opposed to the more declarative point of view taken by classical mathematical subjects."*

In our ZAL environment it is possible to adopt the exploratory approach. Indeed we have found that many students who initially adopted the classical declarative approach switched to the exploratory approach in order to deepen their understanding of the system they were modeling. In particular, various properties relating to the invariant are often not stated (i.e. the system is not completely specified), but are *discovered* through the execution of the ZAL prototype. AI problems have been described as *ill-formed*, in the sense that it is difficult to know when we have solved them. Specifications are similar in this respect, because we are trying to give a formal description of something which may not be fully understood, or which has been previously described in potentially ambiguous terms. What often happens in practice is that students do intertwine specification and implementation. A current postgraduate project is attempting to produce a formal specification and its corresponding executable prototype for a blackboard architecture [5]. It is starting with a specification of what is assumed to be a characterisation of this architecture, as its precise definition is not known, then it will develop and execute the prototype and experimentally explore the behaviour in order to modify the characterisation. It is our opinion that this iterative exploration will prove to be the most fruitful approach in that the exact characterisation desired will be achieved through successive iterations, but the characterisation itself (in Z) will be a precise account of what one understands about the architecture.

5. References

1. C B Jones, *Software Development A Rigorous Approach*, Prentice-Hall, 1980.
2. A Hall, *Seven Myths of Formal Methods*, IEEE Software, September, 1991, pp 11-19.
3. B Ratcliff, *Software Engineering: Principles and Methods*, Blackwell, Oxford, 1987.
4. C Morgan, *Programming from Specifications*, Prentice-Hall, 1990.
5. D D McCracken, M A Jackson, *A Minority Dissenting Position*, in *Systems Analysis and Design - A Foundation for the 1980's*, W W Cotterman et al (eds), 1981, pp 551-553.
6. G R Gladden, *Stop the Life-Cycle, I want to get off*, ACM SIGSOFT Software Engineering Notes, Vol7, No 2, April 1982.
7. B Potter, J Sinclair, D Till, *An Introduction to Formal Specification and Z*, Prentice-Hall, 1991.
8. J M Spivey, *The Z Notation*, Prentice-Hall, 1989.
9. S Hekmatpour, *Lisp and Symbol Manipulation*, Open University, 1988.
10. N E Fuchs, *Specifications are (preferably) executable*, Software Engineering Journal, September 1992, pp 323-334.
11. C Morgan, *Telephone Network*, in *Specification Case Studies*, I Hayes (ed), Prentice-Hall, 1987, pp 73-78.
12. W Swartout, R Balzer, *On the Inevitable Intertwining of Specification and Implementation*, CACM July 1982, pp 438-440.
13. H Abelson, G Sussman, *The Structure and Interpretation of Computer Programs*, McGraw-Hill, 1985.
14. B Hayes-Roth, M V Johnson, A Garvey, H Hewitt, *Building Systems in the BB* Environment*, in *Blackboard Systems*, R Englemore and T Morgan, Addison-Wesley, 1988.

Documenting the Portable AI Lab

Michael Rosner
Dept Computer Science and AI
University of Malta
mros@cs.unimt.mt

Paolo Cattaneo
IDSIA, Lugano
Switzerland
paolo@idsia.ch

Abstract
The Portable AI Lab evolved as a collection of programs for which the documentation played a secondary role. As the system has begun to enjoy wider distribution, the role of the documentation has shifted. It now has two roles to play: to document system operation and to provide educationally relevant support material. In this paper we examine the difficulties inherent to this position, and suggest that the goals of the system would be better served by having the documentation serve as the interface to the programs. We discuss the role of hypertext in this respect, and examine the wider issue of distributing the system over the internet.

1 INTRODUCTION

The Portable AI Lab is an integrated collection of well-established artificial intelligence (AI) tools and techniques intended for use as a resource for teaching or learning AI. The first prototype of the system was developed under a mandate from the Swiss National Fund for Scientific Research (PN23 AI and Robotics) by a consortium consisting of IDSIA, Lugano, the Laboratoire d'IA at the Federal Institute of Technology, Lausanne (EPFL), and the Department of Informatics at the University of Zürich. The National Research Programme which funded the project is now concluded, but development continues on a reduced scale in Switzerland and Malta.

2 STRUCTURE OF MODULES

The system consists of a collection of *modules* in different sub domains of AI which span reasoning, learning, knowledge acquisition, and natural language processing.

Each module comprises

- a computational kernel containing implementations of domain-specific algorithms, special purpose graphics, and a presentation interface;
- some online pedagogical materials including autonomous demonstrations, runnable examples with editable data files;
- documentation (eg exercises, bibliographies, user-guides) for background reading, implementation details etc.

We believe that as a result of this structure the Portable AI Lab should in principle offer a broad-based "tools-oriented" approach to AI that is suitable to students having a wide variety of educational backgrounds, interests etc. We wish if possible for the structure to support four aspects of learning about AI techniques: observation, exercise, background reading and prototype-building. These are in approximate order of difficulty, although background reading is a rather special case.

Observation, a step towards finding out about an AI technique typically taken by the novice student consists in experimenting with demonstrations. A typical module will have two to three of these and we have tried to grade them so that they illustrate some kind of "story". For example, the demonstrations in the planning module are oriented around the story that confronts the theme of achieving several multiple goals with that of linear versus non-linear planning.

Once the student has found out what a tool can do, **prepared exercises** can help him/her discover how to use it in a practical sense. There

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/492 ©1996 FLAIRS

are essentially two kinds of exercises available depending on the nature of the module. Some modules (eg natural language processing, planning, rule-based systems, truth maintenance and theorem proving) are essentially engines whose "programs" are expressed in special-purpose formal languages or notations. Expertise for such modules consists in acquiring an understanding of the formalism, its semantics, and its utility for solving problems. A typical exercise takes the form of augmenting the functionality of a given "program" expressed in the formalism. For example, an exercise from the NLP module might require the student to augment the coverage of a grammar.

Not all modules involve programming languages, however. Repertory Grid, Neural Networks, ID3, for example, all involve the use of dedicated, graphics-based tools. Exercises are designed to develop expert patterns of usage, i.e familiarity with the operation of the tool itself and competence in applying the tool to a given problem. The Repertory Grid exercises, for example, encourage the student to use the tool in order to build a set of structured objects given a set of examples chosen by the student.

Background reading is another aspect of learning that is essential to achieve depth of understanding. Off-line documentation consisting of classic papers from the AI literature or references to them helps to provide this. In addition, implementation details for some modules are included in the form of abstract algorithms. These provide an ideal starting point for those wishing to modify or augment the source code.

Prototype building, i.e. augmenting the functionality of one or more modules by modifying the source code, is a rather open-ended activity that requires the highest level of competence on the part of the student. Support for it comprises source-code documentation, background reading and contributed code. The incomplete nature of this support is offset by the fact that students who build prototypes are typically highly motivated. Term projects in the areas of Natural Language Processing [5] and Genetic Algorithms[7] have been realized using the Portable AI Lab. Eventually, it is planned to incorporate this kind of project work into the distribution.

3 PAIL AND DOCUMENTATION

Historically, the computational side of the PAIL evolved before the documentation side. Although PAIL is intended as an environment in which users can in principle carry out all the above activities, it turns out that for the most part, the full potential of the system has rarely been exploited. Users tend to concentrate upon the most visually engaging aspects of the system (e.g. demonstrations, prepared examples) without exploring the more subtle perspectives that the written documentation, in conjunction with the rest of system, is actually capable of revealing. Ultimately, we take this underexploitation to be a shortcoming in the system documentation itself – or to be more accurate, in the supplementary materials apart from program code that make up system modules.

One of the problems is the fact that this supplementary material taken as a whole is both *non-homogeneous*, by which we mean that it is physically fragmented, dependent on a variety of support media, and yet highly *self-referential*. To illustrate this, consider the supplementary material supplied with the ATP (automated theorem proving), which we consider to be one of the most complete modules to date.

1. A pedagogical document entitled "Logic with Pail" [1] which explains the main concepts of computational logic.
2. Various exercises on the concepts introduced by 1.
3. A user guide which explains the functionality of the module.
4. Online help files.
5. A set of theorem files, runnable descriptions of theorems to be proved.
6. Demonstrations.
7. Lisp source code.

Figure 3 summarizes some of the relations between these materials. Note that although much of it is text-based, 6 and 4 require computational support and so does 5 if it is to be interpreted. Note also the variety of different

No	Subject	Medium	Links
1.	logic concepts	text	2 6 7
2.	logic concepts	text	1
3.	system function	text	3,4
4.	system function	help file	3,4
5.	module data	data file	1,3,4
6.	module demo	program	?
7.	module impl	lisp	3

Figure 1: ATP supplementary materials

formats (essentially text can be put on paper and read whilst everything else requires specialized browsing software). Finally, observe that cross references in many cases span different formats. For instance, we might have a certain logical concept mentioned in 1 for which there exists a demonstration in 6. Currently, such a reference could only be realized though textual description. Similarly, explanatory text accompanying such a demo might need to make a reference to a concept mentioned in 1. Again, the only way that this can happen is through a textual description of some kind.

4 A PROPOSAL

To overcome the shortcomings mentioned above, we suggest that the pedagogical function of the system could be enhanced by taking the documentation as primary and having the programming environment support the documentation instead of vice versa.

Given the diversity and connectivity of the information mentioned in the last section, it seems clear that hypertext would be the ideal material from which to construct documentation since it provides a uniform framework within which to embed both cross-references and specialized browsing and program invocation facilities. However, a major difficulty is the lack of guidelines for designing such a document. The main issues seem to be:

- Whether it is necessary to provide guidance to users to counteract their natural tendency to to wander and get lost, and how to do so in a positive way and at an appropriate level.
- How to implement specialized browsing and program invocation so that they can be easily invoked from the hypertext browser.
- How to organise remote access to the system

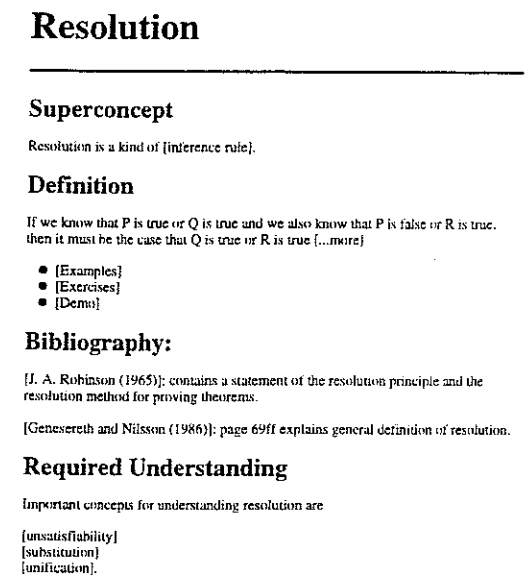
4.1 Providing Guidance

Although hypertext frees both author and reader from the mould of sequential access, there are certain obvious disadvantages that the freedom brings. It is well known that completely unrestricted hypertext can lead to "disorientation" whereby users forget where they came from, and thus the line of thought that brought them to where they happen to be. Equally, the presence of large numbers of links can detract from any attempt to get the reader to adhere to some pattern of accessing the data, as is normally provided by the devices available in textual discourse. Neither of these phenomena is desirable in a hypertext concerning AI, which already suffers from the "smorgasbord" problem (see Hearst [6] for a discussion of this problem and some proposed solutions). Even when aimed at non-specialists, it is desirable to present the subject from some reasonably well-defined perspective.

The attempt to design a hypertext that will overcome these problems involves two stages: (i) Providing a definition of a *perspective* which provides constraints on what constitutes a relevant link; (ii) deciding how those constraints will be employed with respect to an actual hypertext.

In the case of PAIL, the first stage is in some sense partially accomplished since the perspective is provided by the intended usage patterns mentioned above. So for example, the motivated student who wants to find out about a certain domain concept, which might be, say, **resolution** might plausibly want a description of the concept together with pointers to one or more examples, exercises and bibliography as shown in figure 4.1, where the items in square brackets denote links.

If we are prepared to accept that this is a good design (it will not be the only one of course), we then ask how it can be used to constrain the generation of links. At one extreme we can simply write each page by hand, inserting links manually, bearing the design in mind. This is extremely laborious. The other extreme is to attempt to generate generate links *dynamically*



Resolution

Superconcept

Resolution is a kind of [inference rule].

Definition

If we know that P is true or Q is true and we also know that P is false or R is true. then it must be the case that Q is true or R is true [...more]

- [Examples]
- [Exercises]
- [Demo]

Bibliography:

[J. A. Robinson (1965)]: contains a statement of the resolution principle and the resolution method for proving theorems.

[Genesereth and Nilsson (1986)]: page 69ff explains general definition of resolution.

Required Understanding

Important concepts for understanding resolution are

[unsatisfiability]
[substitution]
[unification].

Figure 2: An HTML page for the concept of resolution

making use of conceptual information about the domain and information about the user in the style of Espinoza and Hook [3]. This is laudable in principle but we are not yet in a position to be that confident about the structure of the domain.

We have therefore decided to adopt a more conservative approach in which we initially attempt to codify good page design into abstract graph structures called *idioms*. An example of the idiom type for domain concepts of which figure 4.1 is an instance is given in figure 3.

```
TYPE : domain-concept
TEXT : <concept description>
LINK : exercise
LINK : biblio
LINK : example
LINK : demo
LINK : algorithm
LINK : domain-concept (eg isa)
```

Figure 3: Concept Idiom Type

```
TYPE   : code
CONTENT: <Lisp Code>
LINK   : algorithm
LINK   : code (caller)
LINK   : code (called by)
```

Figure 4: Code Idiom Type

An idiom type consists of some literal data (type information and text) together with typed links to other idiom types such as examples, exercises and bibliographic references (another example of an idiom type for pieces of Lisp code is given in figure 4).

Structures such as these are very close in spirit to the concept graphs discussed by Gaines and Shaw [4] and there is clearly a sense in which an instantiated concept graph is also a fairly detailed representation of both domain knowledge and the way it is organised. There is clearly a huge potential for exploiting knowledge-based techniques to enhance both navigation and authoring of hypertext documentation, a subject which unfortunately goes beyond the scope of this paper.

4.2 Specialised Browsing

One of the stumbling blocks to the development of a system that has core documentation in HTML form and a programming environment like PAIL written in Lisp as programming support has been the lack of a reasonable interface between Common Lisp and HTML. With the appearance in 1994 of CL/HTTPD, a hypertext transport protocol daemon written in Common Lisp, the task has become easier. PAIL, including its documentation, can actually be hidden inside or be distributed as part of CL/HTTPD. The main problem to his approach boils down to rewriting major parts of the user interface in HTML. Even though this may seem a very large task to accomplish we have noticed that much of the *input* to the modules, such as data and parameters, can be easily recoded through the use of forms and much of the *output* from the modules can be easily redirected to requesting browser. A lot remains to be done to let the user interact or play the graphical simulations which are now available in PAIL, possibly the main strength of

the whole system. The best candidate for the support of highly graphical content across the Web presently is Java, a young and free language with all the problems of software in its infancy.

4.3 How to organize remote access to the system

In conjuction with what has been presented above we propose here a way to distribute PAIL accross the Internet which has to take into consideration the main drawback to using the Internet as a programming environment vehicle is its speed. The Internet is clearly too slow to support dynamic and graphical simulations and probably no site would accept to provide computing power and bandwidth for this application. The distribution of PAIL could therefore be divided in two parts. We would have one site acting as repository for documentation, source code, exercises and patches, i.e. all the parts of PAIL that need to be centralized. This site would provide no "active" pedagogical material as such nor would it run any parts of the system.

The second part would be the actual distribution of the working system which should be deployed in an intranet environment, where access to the computing facilities is relatively easy and fast. PAIL would include all the modules presently available plus CL/HTTPD and the code, documentation and guidance systems to let PAIL run as an internet application. We believe that this architecture is rather suitable in teaching environments like universities where the distribution of study material needs to be centralized and the student body distributed. Probably, it falls short for the single user who can take greater advantage of one single environment obeying the standard user interface guidelines supported by the operating system.

5 Conclusion &Future Work

The ideas discussed in this paper are clearly rather premliminary, although some of them will be tested out on a new module that concerns the domain of search. The design of the search module presented certain difficulties concerning which search methods to include and how to provide a visual illustration of their operation. However, these are now largely resolved. The programming is now essentially complete and we owe a great debt of thanks for the inspiration provided by chapters 3 and 4 of Russell and Norvig's book [8]. Space limitations prevent further discussion of the architecture and contents of this module but these will appear in a forthcoming publication [2].

References

[1] F. Baj. LENprover, an automated theorem prover for Portable AI Lab. Technical Report 2, IDSIA, IDSIA, Corso Elvezia 36, 6900 Lugano, Switzerland, 1990.

[2] Paolo Cattaneo and P. Bano. Sps: A search module for the Portable AI Lab. Technical Report 1996/10, IDSIA, IDSIA, Corso Elvezia 36, 6900 Lugano, Switzerland, forhcoming.

[3] F. Espinoza and K. Hook. An interactive www interface to an adaptive information system. *Adaptive Hypertext Archive*, 1996.

[4] B. Gaines and M. Shaw. Concept maps as hypermedia components. Technical report, Knowledge Science Institute, University of Calgary, 1995.

[5] R. Grau. Semantic analysis within the Portable AI Lab. Thèse de diplome, Ecole Polytechnique Fédérale, Lausanne, Laboratoire d'IA, EPFL, 1015 Ecublens, 1994.

[6] Marti Hearst. Preface: Improving instruction of introductory AI. In Marti Hearst, editor, *Proceedings of the AAAI Workshop on Improving Instruction of Introductory AI, New Orleans.* 1994.

[7] R. Limeres. Building a learning classifier system with the Portable AI Lab. Thèse de diplome, Ecole Polytechnique Fédérale, Lausanne, Laboratoire d'IA, EPFL, CH-1015 Ecublens, 1994.

[8] S. Russell and P. Norvig. *Artificial Intelligence, A Modern Approach.* Prentice Hall, Englewood Cliffs, 1995.

Robots in the Classroom

Carl Turner, Kenneth Ford, Steve Dobbs & Niranjan Suri
Institute for Human and Machine Cognition
University of West Florida
{cturner,kford,sdobbs,nsuri}@ai.uwf.edu

Patrick J. Hayes
Beckman Institute
University of Illinois
phayes@cs.uiuc.edu

Abstract

The design and implementation of autonomous robots provides experience with hard problems in artificial intelligence, experience that traditional classroom lectures and programming assignments cannot match. We describe how the use of robots complements graduate-level classroom lectures, show how readings and robots aid students' understanding of philosophical issues in AI, and discuss the practical aspects of introducing robots into the curriculum.

Introduction

Teaching introductory AI is a hard problem. One reason lies in its scope as a discipline: AI spans both engineering and philosophy. To do applied research in AI requires a depth of technical knowledge equal to any other part of computer science or engineering. At the same time, AI has a rich tradition of insight into philosophical questions about human mind and memory, epistemology, and symbolic representation (Glymour, Ford, & Hayes, 1995). AI and philosophy can inform one another, to mutual advantage. Where discussions on the philosophical side of AI predominate, however, some tend to lose sight of the ultimate goal of AI: to design and build smart machines that can carry out tasks in the real world.

Any first course in AI should give students the tools necessary with which to pursue this goal, as well as the philosophical underpinnings on which to base a productive program of research. Addressing the philosophical side of AI leads to a second difficulty in teaching introductory AI: the problem of foundations.

The Foundations of AI

One aim of an introductory course should be to establish the theoretical foundations of a subject. It's not clear or universally agreed, however, what counts as the foundations of AI. Unlike mainstream computer science, there is no generally accepted base of knowledge that serves as foundational material in AI. In one sense, the foundations of AI are the basic technical skills and pragmatic knowledge that should be part of the repertoire of an AI professional. The "pragmatics" approach is well represented in a number of AI texts (e.g., Winston, 1984). However, few current AI textbooks give a proper discussion of some of the deeper issues surrounding AI. In fact, many texts (and AI courses) simply recite ill-informed criticisms of the subject from the philosophical literature, rather than attempt to give students a proper grasp of these foundational issues (Hayes & Ford, 1995a).

We have found that, in contrast to traditional methods of teaching strictly in the abstract, a course which combines a program of critical reading of the literature with the task of building working autonomous robots can provide very motivating and powerful insights into the foundations of AI and computer science. Clearly, building a robot can help students acquire applied skills in system design, planning, and programming. Less obviously, experience with robotics can shed light on some rather difficult philosophical arguments in AI. Issues that appear intractable when discussed in the abstract seem less so in the context of autonomous robots which explore, form limited understandings of, and usefully modify a physical environment.

This article grows out of our ongoing experience using the MIT 6.270 Lego-based robots in an introductory AI course. We describe the coursework, a combination of readings, discussion, and a robot project. We discuss the ways in which the robot project contributes both to applied computer science skills and philosophical issues in AI. We describe the robot kits used in the class, the robot project, and the end-of-semester robot competition. Finally, we analyze the practical aspects of introducing robots into the curriculum and offer suggestions to those who are considering using robots in their own classes.

Readings and Robots

The coursework consists of two principal components: readings and robot building. The readings consist of a quick review of computer science theory from an AI perspective, a collection of foundational AI papers, current AI work, and philosophical discussions. As the semester progresses, the instructor and TA keep abreast of the students' progress and try to connect theoretical issues in the classroom to students' robot building experiences.

A well-conceived robotics project is central to the course. The projects are carefully designed to direct the student toward confronting one or more important topics in AI during the course of the semester-long project. The emphasis is on building a smart machine that can exploit and interact with a given environment. The class culminates in a rather energetic competition between student teams at which they cheer for their

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/497 ©1996 FLAIRS

own robot and often cruelly mock the efforts of the other teams.

What are AI Techniques?

Throughout the course, the point is made that AI is not a particular collection of methods, or a programming style, but that AI is characterized by its goals, not its methods. Roughly, it is the business of using computation to make machines act more intelligently, or to somehow amplify human intelligence. Any technique can be used by a program to do something intelligent, or to display a cognitive ability. In our view, any attempt to identify AI with any particular collection of methods or techniques is an artificial restriction: there is no such thing as an "AI method." AI practitioners should take what works, employing novel means to solve messy, ill-defined problems. Students struggling to build "smart" robots intuitively reject any *ad hoc* constraints on what computational methods they can use, which tends to reinforce this attitude. AI should be allowed to be absolutely eclectic; it is difficult enough without having to conform to arbitrary standards of conduct.

An Appreciation for the Physical World

There is one very important practical problem that the robot builders face when designing their machines. The programmable onboard computer has just 32K memory which must hold the operating system, interpreter, and source code. The limitations of the machine mean that the robot cannot simply sense and store great amounts of data; efficiency is vital. The students must carefully analyze the task and the environment, find constraints, design good representations and write tight, effective algorithms.

This necessity provides several lessons for our students which are hard to obtain from the kind of programming exercise more traditionally used in introductory AI courses. The robots they are building are thoroughly "situated" in their small worlds, and must be designed to take advantage of that situation. This forces the designer to think hard, both about which aspects of the physical environment must be represented (and which need not be represented) in the robot's software, and therefore about the physics involved. The robots must compete in performing a series of tasks in an environment which is deliberately designed to force student designers to consider issues of perceptual ambiguity, simple action-sequence planning, spatial representations and the various temporal tradeoffs involved in real-time search.

Working within limitations on time and resources is a ubiquitous problem in the real world. "If I had a bigger computer I could. . ." is an oft-heard refrain. In fact, there are practical limits to computation, as well as theoretical ones, and the robot project is an object lesson in dealing with these limits.

Robots and Philosophers

It is important to note that the robot project is not simply an interesting (or, in many cases, an over-stressful) diversion. The process of designing and implementing an autonomous robot can illuminate some of the philosophical issues raised in AI.

AI is a dynamic field, characterized by excitement on the one hand, and confusion, disagreement, and lack of consensus on the other. Contributing to the confusion is the claim that AI cannot exist. A small industry has grown up around the field, consisting of philosophers who would demonstrate, through argument and thought experiment, the impossibility of an artificially intelligent machine or of a thinking computer. These arguments are sometimes found persuasive even by people working in the field.

For example, one such argument states that symbols manipulated by machines can have no objective semantic referent, i.e., they aren't "grounded" to things in the real world (Harnad, 1989). Building an autonomous robot which performs as it does only by virtue of its internal representations being firmly "grounded" is an excellent antidote to this argument. It is then easier to understand the import of the "levels" idea which several AI writers have emphasized: McCarthy's "epistemological adequacy," Dennett's "intentional stance," and Newell and Simon's "physical symbol system." All of these accounts are rather abstract and hard to follow for a beginning student, but they can be related very clearly to the student's own practical and immediate problems in designing a simple robot's thoughts. Moreover, the lessons learned in building these small, limited robots extend beyond the classroom. There is a clear extrapolation from these robots to, for example, a robot van which self-navigates around Carnegie Mellon University, a much more complex and impressive machine with a substantially more elaborate representation of things in the real world (Broggi & Berte, 1995).

Through the act of creating their own autonomous robots, students learn that the philosophical issues raised are interesting, and can inform, but that the objections don't prevent the real work of AI from getting done. Presenting the philosophical objections in this context acts, in a sense, as a sort of inoculation against silly, misdirected arguments that might otherwise discourage a potential AI researcher (Hayes, Ford, & Adams-Webber, 1992).

Another very important lesson is that AI, even the apparently "easy" parts, is hard. The skeptical question, "You mean computers can't do *that* yet?" usually assumes that some hard problem is trivial, were it solvable at all. Lay beginners often imagine that while passing the Turing Test may be tricky, just moving around is pretty simple. The robot project, although seemingly simple at first glance, requires a larger investment of time and more previous knowledge of programming and design than students usually anticipate. After the robot project, students need little convincing that there are no easy victories in real-world AI. These lessons form the basis of a realistic view towards the field as a whole, and convey an appropriate attitude of skepticism to such grand ambitions as Turing's imitation game; students learn, as it were, to keep their wheels on the ground (Hayes & Ford, 1995b).

Our Robots: the Good, the Bad, and the Ugly

The particular robot kit that was chosen for our classes was the MIT 6.270 kit. It consists of a programmable battery powered board with a number of slots for motors and sensors to be plugged into it, a number of sensors, and a wide variety of Lego bricks, gears, and wheels for chassis and appendage construction. The sensors used in the kit range from micro switches and buttons to light sensitive photocells and infrared emitters and detectors.

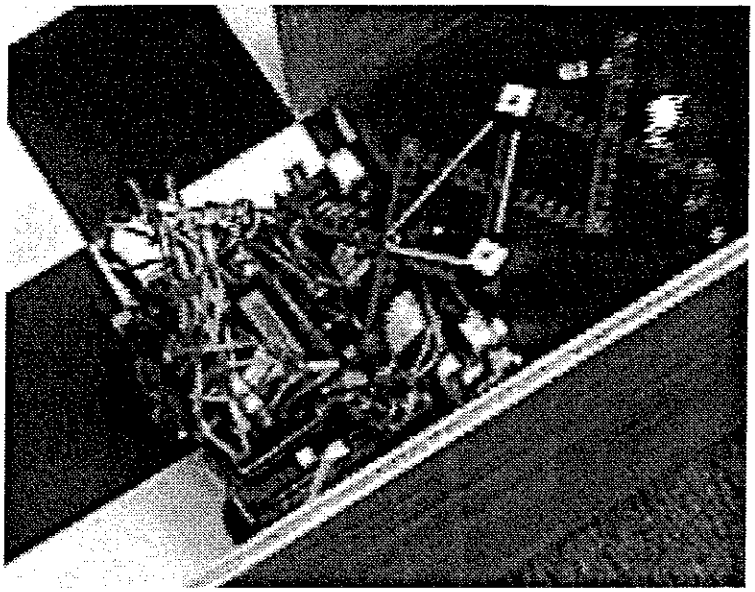


Figure 1: Robot in a maze strikes a target

Additional lights and sensors can be purchased commercially (at a Radio Shack or the like) and constructed easily and inexpensively. The programming environment is IC (Interactive C), which is a specialized version of the C language, and runs on UNIX, PC, and Mac machines. Students write code on the host machine and download the compiled result onto the board. This kit has a number of advantages, including relatively low cost (approximately $650 US) and design flexibility. It does not have the computational power of some of the higher priced kits, but the power is adequate for an introductory level class. The sensors used in the kits are relatively inexpensive and have a fairly low resolution. This impacts on project design, but allows students to realize first hand problems in acquiring information from the world. Figure 1 shows one of the robots in action.

Appropriate Assignments

The robot projects are intended to force students to confront at least one complex issue involved in real world AI applications, such as planning, learning, perception, and navigation. Ideally, the project should be difficult but not impossible to complete within a semester. Students are assumed to have a background equivalent to a B.S. in computer science or closely related field, and to have a working knowledge of C and UNIX. We presently have four robot kits, enough to divide the class into teams of three or four students each.

In an early robot project, mobile robots navigated a wooden maze and scored points by locating and striking illuminated targets. The targets were set on top of platforms located throughout the maze, which required the robot designers to equip their creations with extensible arms, as seen in Figure 1. During an initial learning phase, robots had the opportunity to explore the maze and discover the locations of the targets. During the subsequent scoring phase, the robots returned to the maze and earned points by knocking down as many of the targets as possible within a limited time period. Robots were penalized for touching the walls, or for climbing the target platforms in order to strike the targets.

The project was intended to emphasize issues of learning and planning. Instead, the major difficulty faced by student teams turned out to be that of basic movement; the maze corridors were narrow and difficult to navigate without touching the walls. Students discovered that maze learning was relatively unimportant, within this particular task, if the robot could turn corners reliably, negotiate straight paths, and randomly wander through the maze during the scoring phase. The outcome of the project points out a characteristic of working in a unique environment on a novel task: the constraints assumed to exist by the project organizers were not those uncovered by students during the actual interactive testing and development of the robots.

A more recent robot project stressed object identification and problem solving. The task required robots to navigate a straight corridor, pick up foam bricks at one end of the corridor and place them in scoring bins at the other end. Robots needed to identify the foam bricks which were valued from one to six depending on size and color. Points were earned by filling a scoring bin with any combination of bricks that totaled eleven; bins with bricks totaling more than eleven points were "busted." This project simplified the navigational aspect of robot control but introduced the hard problems of visual perception and physical manipulation of objects. The prototype in Figure 2 has a light sensitive photocell at the base of its "claw" that allows it to discriminate between light and dark colored foam bricks.

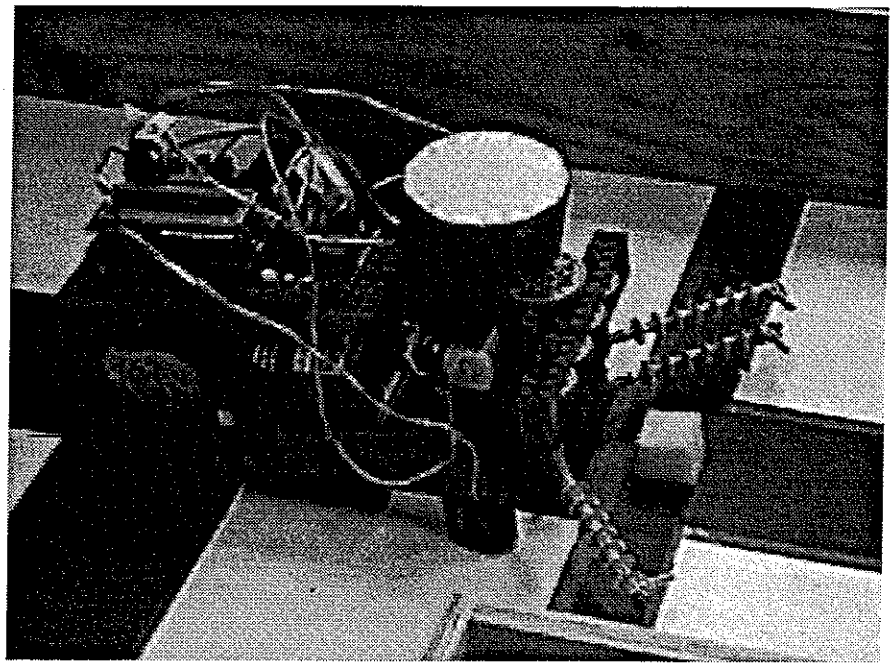


Figure 2: Robot prepares to grasp and identify a small foam brick

At semester's end, student teams are evaluated not only on the success of their robot in competition, but on individual aspects of robot programming and mechanical design. Teams earn points based on the elegance of their solutions to specified problems of navigation, object identification and classification, grasping, and so on. Students are required to evaluate the successes and failures of their project in a final written report. They also are asked to rate the contribution of other team members to the design and construction of their robot. Two essay tests during the semester cover the readings discussed during class lectures.

Robotics is not specifically taught in the class. In fact, there is still a great deal of disagreement over what constitute the proper procedures in robotics, a fact that in itself makes the robot project attractive for this course. Students get help from a few resources in the form of a web page and links to a page that is maintained by MIT.[1] Otherwise, there is no formal instruction in robot building. Those interested in formal instruction in robotics are encouraged to register for the upper-level graduate robotics course.

Summary

Designing and constructing physical robots helps draw students into some of the hard problems in AI, and facilitates their immersion into this difficult subject. The usual introduction to AI treats it as part of computer science, on a par with data structures or compiler design. It strives to convey as much information as possible about techniques which have been developed in AI. We aim, rather, to convey a deeper sense of the attitudes and problems of AI, to demonstrate just how difficult it is, and to proclaim its extraordinary intellectual audacity and importance.

References

Broggi, A., & Berte, S. (1995). Vision-based road detection in automotive systems: A real-time expectation-driven approach, *Journal of Artificial Intelligence Research, 3*, 325-348.

Harnad, S. (1989). Minds, machines, and Searle. *Journal of Experimental and Theoretical Artificial Intelligence, 1*, 5-25.

Hayes, P. J., & Ford, K. M. (1995a). Intellectual archeology. *SIGART Bulletin, 6*, 19-21.

Hayes, P. J., & Ford, K. M. (1995b). Turing Test considered harmful. *Proceedings of the International Joint Conference on Artificial Intelligence* (IJCAI—95), Montreal.

Hayes, P. J., Ford, K. M., & Adams-Webber, J. R. (1992). Human reasoning about artificial intelligence. *Journal of Experimental and Theoretical Artificial Intelligence, 4*, 247-263.

Glymour, C., Ford, K. M., & Hayes, P. J. (1995). The prehistory of android epistemology. In K. M. Ford, C. Glymour, & P. J. Hayes (Eds.), *Android epistemology* (pp. 1-21). Cambridge, MA: MIT Press.

Winston, P. J. (1984). *Artificial intelligence* (2nd ed.). Reading, MA: Addison Wesley.

[1] The URL for our Lego robot homepage is http://amaru.coginst.uwf.edu:80/lego/

DESIGN CONSIDERATIONS FOR AN AI COURSE FOR THE TYPICAL CS STUDENT

Richard Wyatt
Department of Computer Science
West Chester University
wyatt@checkov.wcupa.edu

Abstract

This paper discusses design criteria, and the specific course that resulted from them, for a course in Artificial intelligence at what might be described as a "typical", or non-elite, institution; that is, one from which the students are unlikely to go on to pursue doctorates or be employed as high-level AI researchers.

1 INTRODUCTION

There is, of course, no "correct" way to teach an AI course. AI differs, in at least the extent if not in kind, from most other computer science courses, for which there is usually a fairly well accepted set of topics that constitute it's content. The AI course I taught in the Fall semester of 1996 was designed to suit my own needs and those of my students, whom, because of their abilities, interests, and aspirations, I regard as "typical" of the computer science students taking AI. Nonetheless, I venture to think that perhaps the considerations I have used in designing the course might be of more general interest. I present these considerations and an outline of the course that resulted.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/501

2 DESIGN CONSIDERATIONS

The many considerations that influenced the design of my AI may be grouped together under the following four headings:

2.1 The Students

I keep in mind the abilities and aspirations of my students. Almost all my students will complete either a baccalaureate or masters degree in computer science, concentrating in the more "technical" areas, or those they perceive to offer the best employment prospects. My students are likely to be employed only in what may be termed *marginal AI*, work that is only marginally regarded as AI by the AI community. The feedback from my students is that they have made specific use in their jobs of OPS5, KR via productions, expert systems in general, and search methods. I therefore try to emphasize more than I otherwise might those aspects of the topics I cover that will be useful to my students in their likely careers.

I also try on keep an eye on what topics students find exciting or "sexy"; genetic algorithms and neural nets at present seem to be so. Since such topics are candidates for inclusion in the course anyway, I include them if possible. I have found that more than an occasional student has taken my AI course only because some such topic was being covered. As often as not, of course, the student turns out to find other topics more interesting.

2.2 Course Topics

I try to choose topics that are more easily related to other areas of computer science; for example, most of my students have already seen logic in Discrete Math, and language theory and parsing in Compilers. I favor topics that are interconnected and lead into each other as much as possible; there are, for example, many such interconnections among expert systems, logic, inference, chaining, search, KR, and NLP. In this way, I hope to avoid the all too common impression that AI is merely a set of disconnected topics or special techniques. I also do not try to cover too many topics: the range of topics covered is, by and large, one of my least important factors in determining the course content.

These consideration have led me to focus my course around three areas: search, inference, and NLP. I do not mean that these are the only topics I cover; rather, I try to present the topics I do cover in a way that focuses or emphasizes these three topics. For example, I try to let the NLP flavor emerge from the discussions of the Turing Test, ELIZA, logic and KR, CYC and common sense, neural nets, and of course NLP itself. I find that emphasizing or highlighting these three areas focuses the course considerably.

2.3 Programming

I require about as much programming as any other typical computer science course, since, I believe, students learn "at a deeper level" when they must implement things. Programming projects and exercises are a mix of computer science type coding problems (for example, solving the 8-tile puzzle, or implementing a resolution theorem prover) and problem solving using existing code but without – or with little – actual coding (for example developing rules for an ELIZA system, or writing a grammar for an ATN).

I try to avoid teaching any programming in my AI course. The problem of exposing students to Lisp, Prolog, or some other language suitable for AI is partly addressed at my school by having recently implemented a course, required of computer science majors, in which Lisp and Prolog are examined.

2.4 Theory

Because AI problems remain unsolved, it is important to expose students to AI theory, and thus to the level of work conducted by actual AI researchers. Because of the interests and abilities of my typical students, however, I do not do much truly "hard core" theory. Instead, I give reading assignments, in which students read a mix of classic and research level papers in the field. The intention here is not that the students should absorb very nuance of the papers, but they should get some understanding for the material presented, develop a sense of the complexity and difficulty of AI problems and research, and, most importantly, come to see that AI is more about trying to solve very difficult problems we only partly understand, than about trying implement solutions to relatively straightforward problems that we already fully understand.

The assigned papers are discussed, even if only briefly, in class; and the exams include a question or two on the papers, though, in reality, because I do not expect the papers to be fully grasped, such questions test little more than whether the student has read the relevant paper.

3 CURRENT COURSE

The above design considerations led to the following course, which was taught in the Fall of 1996.

3.1 Text

The text used in the course was Tanimoto [6]. It contains a good deal of (Common Lisp) code, which is also ftp-able. Tanimoto's approach is to present a brief introduction to a topic and then to offer, for examination and detailed discussion, Lisp code that implement solutions to one or more of the matters discussed. The code, although not overly difficult is design for most AI students, is rather substantial in terms of quantity: the code and the discussion thereof occupies much of the text.

In the past, I have shied away from such texts on the ground that AI should be, and should be seen by the students to be, entirely separate from any given implementation of a solution, which will in

any case be at best a crude and partial solution to the problem at hand. However, when I have taught this course in the past, it has always been my sense that a number of students have viewed it as insufficiently connected to the day to day nuts and bolts of programming, which is their prime love, or at least their prime concern. So I thought I might give Tanimoto a shot.

Although not an utter disaster, the text did not live up to expectations. The constant focus on code made it very difficult to focus on AI *per se* and provided little or no compensation as regards an increase in the extent to which students viewed the course as a "mainstream" computer science course. I would not select the text again.

3.2 Introduction

A general introduction to AI. I survey the AI landscape, air some of the philosophical matters, and, in this class, tried out for the first time the "human robot" experiment in which students serve as the eyes, brain, and two arms of an intelligent robot and thus attempt tasks such as the stacking of one box on another. The reading assignments are selected parts of Turing's paper [7] on the Turing Test, and some newspaper articles on the use of AI in industrial or "real world" applications: on a checkers playing program and on AI used in a Compaq printer. Many AI researchers would not, of course, be inclined to regarded either as real AI, but the point of the articles is to help convince students that AI is perhaps not entirely *ir*relevant to their vocational aspirations.

3.3 Matching

Pattern matching, Eliza, production and expert systems, and forward and backward chaining. The reading assignment is Weizenbaum's ELIZA paper [8].

3.4 Search

Standard search methods: optimal, non-optimal, heuristic, and game playing, but with an emphasis on heuristic search using examples such as the 8 and 15 puzzles. There is no reading assignment.

3.5 KR1

Frames, scripts, semantic nets, the frame problem, CYC and so on. The reading is on CYC [2].

3.6 KR2

After a rapid review of first order logic, which students have already seen in depth, I cover modal and belief logics, nonmonotonic reasoning, TMS, logical reasoning, unification and resolution. The reading assignments are McDermott's *A Critique of Pure Reason* [3] and Moore's *Problems in Logical Form* [4], which leads directly into NLP.

3.7 NLP

Syntax, semantics, pragmatics, grammars, parsing, generation, and discourse. The reading assignments are Woods' *Transition Network Grammars for Natural Language Analysis* [9] and parts of Grishman's *Computational Linguistics* [1] .

3.8 Genetic Algorithms and Neural Nets

Elementary neurophysiology and neural networks, connectionist and symbolic AI, elementary genetics, crossover, and mutation. The reading assignment is Rumelhart and McClelland's *On Learning the Past Tenses of English Verbs* [5].

3.9 Projects

I gave four project. They focused a little less on the NLP than I might otherwise have liked: I did not want to give more than four projects, and I wanted to give fairly standard projects on search and resolution.

3.9.1 Project 1 – Eliza

Students are given SHRINK, which is Tanimoto's Eliza-style program (in Common Lisp). After reading about the Loebner Prize and some transcripts of the actual competition in 1993, they chose a conversation topic — either one of the Loebner Prize

topics (which were baseball, hockey, relations with your real or imaginary spouse, whimsy, and men versus women) or, subject to my approval, a topic of their choice — and develop a set of rules to yield "convincing" conversations. Students then modify the code to read in the input sentences directly, change the behaviour of the default responses, and add a "topics" feature so that SHRINK can sometimes, instead of giving a default response, return to a previous topic in the conversation.

The focus is on developing a good set of rules and thus, it is hoped, reinforcing the idea that AI resides much more in the design of a solution to a problem than in the "hacking out" of the code.

3.9.2 Project 2

Students implement from scratch, in either Common Lisp or Prolog (whichever they prefer), a solution to the 8-tile puzzle using best-first search. They assess the performance of the algorithm with each of the following heuristics: no heuristic, the number of tiles in correct position, the manhattan heuristic, and another heuristic of their own design. Only 5% of the grade was allocated for this last heuristic since I did not expect most of them to come up with much. My expectation was fulfilled, although one student employed, not really a heuristic, but a worthwhile strategy: move tiles 1-3 to their correct positions, then without moving 1-3 move 4 and 7 to their correct positions, and finally without moving 1-3, 4, and 7, move 5,6, and 8 to their correct positions. (The goal state was: top row: 1, 2, 3, middle row: 4, 5, 6, bottom row: 7, 8, blank.)

3.9.3 Project 3

Students implement a resolution theorem prover for propositional logic in either Lisp or Prolog and test it on a suite of examples. Algorithms for CNF and resolution were discussed in class in sufficient detail to facilitate the coding.

3.9.4 Project 4

Students are given an ATN and some sample grammars and asked to write a grammar to parse a number of sentences. Most of the sentences are straightforward, although they do include the distinction between mass nouns and count nouns, as illustrated by the following phrases:

much gold	many dogs
less gold	a few dogs
a lot of gold	every dog

4 POST-COURSE REPORT

The choice of text book notwithstanding, the course seemed to be successful, although I do not think it was noticeably more or less successful than the AI courses I have taught in the past. Although it certainly seemed to me that this course was more unified, more in accord with a single theme and so less disjointed, it is unclear to me that the students so regarded it, perhaps because they had so little exposure to alternatives. Nonetheless, I was pleased with the course and will continue to develop it along these same lines in the future.

References

[1] Ralph Grishman. *Computational Linguistics.* Cambridge, 1986.

[2] R. V. Guha and D. B. Lenat. Cyc: A midterm report. *AI Magazine*, 1990.

[3] Drew McDermott. A critique of pure reason. *Computational Intelligence*, 1987.

[4] R.Moore. Problems in logical form. *Proceedings of the 19th Meeting of the ACL*, 1981.

[5] D. E Rumelhart and J.L. McClelland. On learning the past tenses of english verbs. In *Parallel Distributed Processing, Vol. 2.* MIT Press, 1986.

[6] Steven Tanimoto. *The Elements of Artificial Intelligence.* 2nd Edition. W. H. Freeman, 1995.

[7] A. M. Turing. Computing machinery and intelligence. *Mind*, 1950.

[8] J. Weizenbaum. Eliza – a computer program the study of natural language communication between man and machine. *CACM*, 1966.

[9] W. A. Woods. Transition network grammars for natural language analysis. *CACM*, 1970.

HEART — A MODEL OF EMOTION FOR NATURAL LANGUAGE INTERPRETATION. 4. APPLICATION TO TEXTUAL ACCOUNTS OF HUMAN ABDUCTIONS.

Mark S. Schmalz
mssz@cis.ufl.edu

Douglas D. Dankel II
ddd@cis.ufl.edu

Department of Computer and Information Science and Engineering
University of Florida, Gainesville, FL 32611

ABSTRACT

The rising frequency of kidnappings by terrorists and child abductors has yielded numerous natural language (NL) narratives, including abductors' suicide notes and voice messages [1]. Increasingly, hypnotic regression of victims under interrogation or therapy yields transcribed dialog that supports investigation [2]. The semi-automatic interpretation of such narrative, supported by an abduction KB, would be a useful tool for law enforcement investigators. The KB could be searched for correspondences, inconsistencies within testimonies, and could possibly be a source of clues for future investigations. However, such interpretation requires understanding of human emotions associated with key features of abduction events.

We are developing a model of affect (emotional state) called HEART-III [3], which is *macrostructural*, i.e., models emotion production and modulation above the stimulus-response level and well above the physiological level [4,5]. This paper overviews work in progress pertaining to HEART-III's role in interpreting NL abduction accounts. For example, we examine the degradation of objectivity in therapist-abductee dialogue due to priming, as manifested in emotional reverberation, decay, and modulation in abductors, abductees, and interpreters (therapists). Additionally, we summarize the ongoing construction of an abduction KB for terrorist, for-profit, and child abductions that would facilitate future research in abduction classification.

1. INTRODUCTION

Increasing terrorist attacks upon developed nations is a sequel of the Cold War that consumes few material resources but appears to be socially debilitating. In particular, victims of terrorist kidnappings are subjected to psychological and physical traumas that are detrimental to post-release behavior. Understanding the underlying emotional interactions and processes inherent in abductor-abductee dialogue is crucial to successful post-release therapy [6,7]. Interaction between abductee and therapist, which is often intense, can unfortunately compromise the objectivity of abduction account analysis [8]. Such *priming effects* are especially crucial during hypnotic or hypnoregressive therapy, which is increasingly employed as an investigative tool.

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/505 ©1996 FLAIRS

Similar observations can be made in the cases of: (1) abduction of wealthy or prominent persons for profit and (2) the tragic and widespread abduction of children for sale or abuse. In the latter case, post-release trauma is experienced primarily by the survivors of the abductee, who frequently disappears. Thus, therapy for Case 1 could have requirements similar to terrorist abduction therapy, while Case 2 would require a different approach due to abductee absence.

Alleged therapist manipulation of abductees has fueled controversy in a fourth type of abduction belief or experience, called *alien abduction*, where kidnapping is purportedly followed by interaction with bizarre creatures [9,10]. Several well-defined corpi [11] of transcribed therapeutic dialogue (a) describe in detail this widely sensationalized phenomenon, (b) have been gathered during professional therapy sessions, and (c) have been transcribed in conjunction with therapists [11]. Variance in overall episodic content is generally low, but environmental details vary widely. Due to unusually intense emotional content, such accounts are useful initially only for probing the limitations of our model and NLI theory. The authors do not assert the validity of such bizarre accounts within well-accepted limits of reality and existence.

In contrast to few data on alien abductions, accounts of terrorist, for-profit, and child abductions are widespread, but have been gathered under controls that frequently range from nonclinical to nonexistent. Dialogue may be obtained at a crime scene, police station, abductee's home, or therapist's office. Initial data gathering and analysis, frequently performed by personnel with whom abductees may have compulsory involvement (e.g., police or legal personnel), tends to negatively bias abductees' subsequent feelings. Even when rendered somewhat more objective by physical evidence, the human interpretation of terrorist, for-profit, and child abduction accounts can be confounded by sparse, unreliable, or conflicting testimony.

This paper presents (in Section 2) a general model for abduction experiences based upon reputable, published analyses of the four preceding abduction types. Classification is validated by analysis of over 50 abduction accounts. Section 3 overviews methodology for converting abduction accounts to model data structures [3]. Section 4 summarizes preliminary results, including a discussion of priming as well as implementational issues of consistency and the effect of belief systems. Preliminary conclusions and suggestions for future work are summarized in Section 5.

2. GENERALIZED ABDUCTION MODEL

Our development of an abduction KB is based upon a two-level model (Section 2.1) derived from analysis of 22 cases of terrorist and for-profit abductions [12,13], 13 documented cases of child abduction [14,15], and 17 published summaries of alien abductions [9,16]. Section 2.2 emphasizes comparison of modelled abduction types.

2.1. Model Overview.

We model abductions using five major tasks, where PPC denotes psychological or physical control:

1. **Attract:** Abductors (*agents* in HEART-III) alert, approach, and attract an abductee (*recipient*).
   - 1.1. **Announce:** Agent emits stimulus.
   - 1.2. **Attend:** Recipient attends to stimulus.
   - 1.3. **Approach:** Agent nears recipient (or vice versa, in some child abductions).
   - 1.4. **Arrest:** Agent obtains partial PPC of recipient.
2. **Capture:** Agent(s) reduce(s) the recipient's psychological and physical capabilities to a level nonthreatening to the agent.
   - 2.1. **Subdue:** Agent gains psychological or emotional control of recipient.
   - 2.2. **Secure:** Agent physically restricts recipient, further limiting recipient's capabilities.
3. **Transport:** Agent relocates recipient to an impoundment facility (can be repeated).
   - 3.1. **Register:** Agent positions or prepares recipient physically or emotionally for transfer.
   - 3.2. **Transmit:** Agent moves recipient (often quickly) to impoundment facility.
4. **Interact:** Agent interacts physically, verbally, or psychologically with the recipient.
   - 4.1. **Subdue, Secure:** Agent regains or may increase PPC of recipient.
   - 4.2. **Modify, Inform, Interrogate, Instill:** Retaining primary PPC, agent interacts physically, didactically, or emotionally with recipient to modulate information input or storage.
5. **Return:** Agent relocates recipient to a place of relative safety, partially restoring recipient's physical or psychological capabilities.
   - 5.1. **Subdue & Secure:** Reference Subtask 4.1.
   - 5.2. **Transmit:** Agent relocates recipient.
   - 5.3. **Register:** Recipient is positioned for release.
   - 5.4. **Restore:** Agent returns partial PPC to recipient, while retaining partial control until the agent's departure or demise.

The **Subdue, Secure, Register,** and **Transmit** operations can be considered identical at a high level, although implementations may differ with time, entity identity, purpose, or location. Similarly, Subtask 4.2 can vary widely with abduction type, entity, or environment.

2.2. Abduction Classification.

The preceding model facilitates classification of abduction accounts based upon task frequency, per Table 1. Successful abductions require completion of Tasks 1–3 (*attract* through *transport*), and were the only types of abductions studied. In cases where attraction was employed little or not at all, the abductee was merely captured against his will. *Interaction* (Task 4) is termed *slight* (e.g., feeding and exercising the abductee); *moderate* (e.g., verbal dialogue or instruction, which may be abusive); or *high* (e.g., physical abuse or intense dialogue). In the majority of all cases across all abduction types, high interaction implied moderate as well as slight interaction. For example, interaction appears to be slight to moderate in terrorist abductions except for political indoctrination or interrogation, which may be accompanied by physical abuse. In child abductions, interaction may range from slight (maintenance) to high (rape or murder), but is difficult to verify due to the tragically low return frequency. For-profit abductions of elderly recipients may feature provision of medical services (moderate interaction if agents are not service providers). Alien abduction accounts often describe severe psychological or emotional interaction, with bizarre accounts of rape and vivisection [9,10,16,17] that frequently test model limits.

Table 1. Abduction type as a function of task and frequency-of-occurrence, computed in terms of N accounts (bold numbers). Italics denote data abstracted primarily from abductors' statements.

Task	Abduction Type			
	Terrorist	For-Profit	Child	Alien
Attract	0.42	0.40	0.85	0.47
Capture	1.00	1.00	1.00	1.00
Transport	1.00	1.00	1.00	1.00
Interact-slight	*1.00*	*0.90*	*0.84*	1.00
Interact-medium	*0.75*	*0.80*	*0.54*	0.82
Interact-high	0.58	0.10	*0.23*	0.76
Return	0.58	0.70	0.16	1.00
$N_{accounts}$	**12**	**10**	**13**	**17**

Politically-oriented terrorist kidnappings can lead to abductee death, often due to interrogation procedures. Understandably, for-profit abductions and alien abduction accounts usually involve recipient return. (With tongue in cheek, we note that the 100 percent rate of abductee return in alien abductions studied correlates well with the absence of abductor testimony.) In all but alien abductions, the frequency of certain interaction types exceeds frequency of return (expressed in italics in Table 1). In such cases, interaction data was partially provided

by abductors and may have been falsified to mitigate the perception of abductee mistreatment, especially if the abductee disappeared.

3. EMOTIONAL MODELLING AND NL

3.1. Translating NL to Emotion Frames.

We previously summarized [5] our technique for translating NL text into data structures called *emotion frames*, which describe: (a) entities' emotional states, (b) influence of dialogue and environment on emotional state, and (c) flow of emotion in a sentence. The following steps are performed:

1. Parse the NL text and determine superficial semantics in the customary manner;
2. Locate the emotion-related words and examine their syntactic and grammatical environments to obtain preliminary conjectures about flow of emotion.
3. Using the information obtained in Step 2, resolve modifier, pronoun, and verb references in the customary manner, then in terms of transfer-of-emotion. For example, transitive verbs imply emotional transfer from verb subject (*agent*) to verb object (*recipient*), while intransitive verbs usually transfer emotion to the agent. Occasionally, emotion is diffused outward from agent to discourse environment, which may include the environment of interpretation or verb action.
4. Translate the preceding flow of emotion into numerical *perturbation vectors* that are applied to numerically-represented *emotional states* of agent, recipient, and interpreter. Perturbations are implemented in terms of a *transition function* that accepts the emotional state and perturbation vector specific to each entity [3], as well as environmental descriptors and entities' assessments of other entities' emotional states.

Our parser is driven by a linguistic KB that has a multi-level grammar which facilitates parsing at the clausal and phrasal levels, and tags words by part-of-speech to facilitate ambiguity resolution [12,13]. Emotion-laden words are contained in a watchlist that is applied to parser's token stream. Each emotional word is tagged with a list of pointers that reference contexts, perturbation vectors, and emotional states that are deemed a priori to be appropriate to that word. This avoids I/O bottlenecks incurred by information retrieval from the KB during semantic interpretation, and provides a preliminary estimate of emotional semantics.

Pronoun reference directs the flow of emotion toward the subject (and object) of an intransitive (transitive) verb. Consider a terrorist's utterance, "I don't feel too badly. I felt jittery before, you know, months ago or even weeks ago just thinking of it" [19], where *it* refers to an aircraft hijacking attempt with hostage taking. Tagged emotion-related words are *feel, badly, felt, jittery, thinking*, and relevant modifiers are *too* and *just*. In Step 3, we tag colloquial phrases (e.g., *you know*) with a tentative emotional perturbation to aid the construction of perturbation vectors in Step 4. For example, *you know* indicates willingness to relate to others, implemented in HEART as a positive change in emotions of *faith* and *ambition* [3,5].

Unresolved references are determined using heuristics stored in our linguistic KB, with search of surrounding utterances to determine non-standard pronoun referents. For example, ambiguous pronoun reference is resolved by first examining candidate referents' person, number, and case, then choosing the best match nearest the pronoun. Occasionally, resolution may require additional clues. For example, consider "Mr. [X] was ahead and Mr. [Y] were behind, so we put him in the truck first. So...we had to leave *him* behind". The preceding heuristic indicates that Mr. X (abductee) should have remained behind, not Mr. Y., but correct resolution requires discourse- and corpus-based search. Extent of search depends on narrative fragmentation that can reflect the narrator's emotional state. We are also investigating a qualitative temporal reasoning model [20] for determining temporal precedence, which (in this case) implies physical/spatial precedence.

Given a tentative picture of emotional flow, we apply HEART to known states to produce a revised emotional state. When sentences contain customarily sparse information about the passage of time, little physical basis exists for inferring temporal effects such as emotional decay and reverberation. This is of less concern to automated interpretation, versus computer simulation of a human interpreter who could prime the recipient as the abduction narrative unfolds [3,21].

3.2. Types of Interpretive Interactions.

In [3], we listed three cases of interpreter interaction with modelled entities:

Case 1. An emotionless automaton interprets an abduction account a posteriori and thus cannot interact with the agent or recipient.

Case 2. An interpreter's emotional state can be perturbed during a posteriori analysis of an abduction account, but agent or recipient can neither perturb nor assess the interpreter's emotional state.

Case 3. An interpreter perturbs and assesses emotional states of agent(s) or recipient(s), but the agent (abductor) is not able to perturb or assess the interpreter's emotional state.

A fourth case, full interaction, is a trivial enhancement of Case 3 where entities perturb each other.

Case 1 is useful for objective analysis of textual accounts a posteriori. Case 2 is employed in simulating interpreter bias during textual or dialogue analysis. Cases 3 and 4, which are more realistic, require difficult simulation of rich interaction among entities that is predicated upon knowledge of emotional stimulus propagation and filtering in a group dialogue setting. As mentioned in [3], the simulation of different belief system influences, based on ontologies or paradigms of existence, perception, or cognition held by various entities, remains an open problem. Such ontological conflict rarely occurs in for-profit and child abductions, where motives are generally straightforward. However, the political context of terrorist abductions often requires deep knowledge of cultural semantics (e.g., notions of justice or jurisprudence). Additional research is required in alien abduction accounts, where concepts of physical

reality and existence tend to be confounded by bizarre event descriptions [9].

3.3. Examples of Emotional Interpretation.

Consider the following run-on sentence, where an abductor (agent) describes his motivation for premeditated air piracy and hostage-taking. Tagged emotional words and phrases are in bold face, while corroborating words that point to emotional expressions are italicized.

> "It's always easy for the **authorities** to *look outward* for the causes to *say this guy acted as a* **madman**, a **mad dog** and *this guy acted as* a **maniac** when basically the *causes of* these **hostile** *actions*, or at least *my* **hostile** *actions* are *inward*; that of *being* **robbed** and **cheated** out of *my* **dignity** and *seeing my country* being **raped** and **ravished** almost *before my very eyes* and *I won't* **stand idly by** and **allow it to happen** [19].

Following the procedure given in Section 3.1, we first transform NL into the high-level graph shown in Figure 1, which depicts non-interaction between an interpreter (Authority) and the agent (Byck), who projects emotion to the recipients (hostages) as aggression. Byck also observes a passive recipient (his country or fatherland) being "raped and ravished". As shown in Figure 1, Byck's emotion complexes of *concern* and *anger*, which can be encoded as state vectors in HEART, are self-reinforcing. For example, given the emotional order of (*ambition*, *faith*, *generosity*, *humility*, *joy*, *love*, *peace*, and *tolerance*), *concern* corresponds to the state vector (0,1,[1,2],1,-1,[1,2],[-1,0],[-1,1]), where $[x,y]$ denote an interval on the set of emotional degrees $\{-3,-2,...,3\}$. Since the account was tape-recorded a priori, neither the agent nor recipients influence the interpreter beyond the recorded content, and the interpreter influences (primes) neither the abduction process nor the other entities. The agent interacts fully with the hostages (recipient #1), and observes his fatherland (recipient #2). The agent's self-projected concern derives from his observation of the fatherland, which is a posteriori, distal, and introspective. Thus, recipient #2 remains unmodified by the agent.

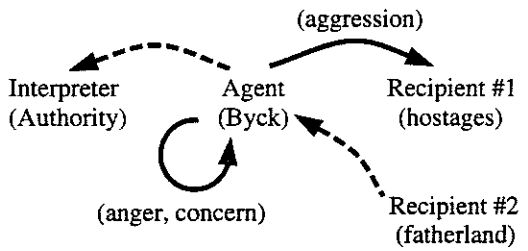


Figure 1. Schematic representation of emotional flow in a terrorist hijacking narrative, where dotted lines denote observation (e.g., of Byck by Authority), and solid lines denote flow of emotion.

We next simplify idiom such as *look for*, *acted as*, *stand idly by*, *allow it to happen* by translating (for example) *look...for* into *research* and *stand idly by* into *observe*, to reduce semantic processing and possible ambiguity. A narration frame is then constructed from simple sentential utterances (*subject*, *verb*, *object*) with an optional prepositional phrase (*link*, *environment*), after [12]. Compound sentences and verbs are denoted by `compound-s` and `compound-v`, while utterances are denoted by `ut-<n>`, where `<n>` denotes an integer. For purposes of brevity, the narrator is denoted as `nar1`.

```
(narration-frame
  :narrator "Byck"
  :context  (:mode tape-recording :when a-priori
               :time (< 1 hour))
  :action   (compound-s $ut-1 $ut-2 $ut-3 $ut-7))
  :narrative
    (ut-1 (:subj authority
           :verb research
           :obj  (cause-of $event))
    (ut-2 (:subj authority
           :verb state
           :obj  (occur $utterance-3)
           :link (:qualifier explain contrast)
           :env  $ut-4)
    (ut-3 (:subj $nar1
           :verb perform
           :obj  (:function-of (attrib human
                    (male angry insane maniac))))
    (ut-4 (:subj (cause-of
                   (attrib $event hostile
                   (attrib (emot-of $nar1) hostile)
           :verb is
           :obj  (compound-s
                   (exist (emotion-of $nar1))
                   $ut-5 $ut-6)))
    (ut-5 (:subj %someone
           :verb (rob cheat)
           :obj  $nar1
           :link from
           :env  (:emotion-complex dignity)))
    (ut-6 (:subj $nar1
           :verb observe
           :obj  (occur $ut-7)))
    (ut-7 (:subj %someone
           :verb (rape ravish)
           :obj  (possess $nar1 fatherland)))
    (ut-8 (:subj $narrator
           :verb (compound-v (not observe)
                             (not allow))
           :obj  (%tentative &ut-7)))
```

The modifier `%tentative` describes a conjectural construction resulting from the pronoun resolution in the phrase *allow it to happen*. Such resolution is consistent since *happen* corresponds with our interpreter's internal word *occur*, which was most recently seen in the verb-object of `ut-6` that references `ut-7`. Thus, it is reasonable to make `ut-7` the tentative construction of the object in `ut-8`.

Predicates such as `cause-of` and `occur` imply that our KB is based on a deterministic ontology. Implementationally, such predicates can be hardcoded in an NLI system's command interpreter, which transforms an utterance representation into KB actions that retrieve, modify, and store knowledge about abduction events. To accomodate additional on-

tologies or paradigms of existence in future versions of our system, such predicates could (for example) indicate assignment of a probability measure to a given entity. This area, which is not central to the topic of this study, will be considered in future research.

The preceding narration frame has emotion descriptors that help determine flow of emotion in the narrative sequence denoted by `(compound-s $utt-1 $utt-2 $utt-3 $utt-7)`. For example, the subject of utterances 1 and 2 is **authority**. Note that HEART's emotional measure of *faith* ranges from extremes of *terror* and *fear* to *hope* and *expectation*, and *humility* ranges from *arrogance* to *submission*. It is reasonable to state that the presence or exertion of authority would generally tend to perturb the *faith* and *humility* emotions negatively. Per utterance 2, *authority* states that the narrator is angry, a maniac, and is insane (in the context of his emotions described in utterance 4). In contrast, the deep semantics of Byck's narrative (per Figure 1) imply that Byck's hostile emotions are inwardly derived and directed (i.e., `(cause-of (attrib (emotion-of $nar1) hostile) (emotion-of $nar1))`. Here, the tagged word *hostile* perturbs the emotion *love* negatively toward *anger* and *hatred*, which is reinforced by utterance 3. Finally, enumerating entities in the narration frame and posing/validating conjectures about emotional flows helps us obtain a more detailed version of the graph shown in Figure 1, whose arcs are labelled with a perturbation vector or emotional state assessment.

3.4. Modelling of Emotional Priming.

In [3], we showed how reverberation of an emotional state e can be modelled as an exponentially damped sinusoidal perturbation of emotional degree having temporal frequency ν, amplitude ξ, and decay constant τ. Implementationally, HEART causes an emotional oscillation by applying a perturbation $\mathbf{p}(t)$ to $\mathbf{e}(t)$, where t denotes time. Gradual emotional changes are simulated using small τ and large ν, while a reduction in ν can simulate pathological states (e.g., manic-depressive mood swings). As shown in Figure 2, a therapist (interpreter) could:

(a) Estimate τ to effect near-optimal priming by producing a periodic forcing function $\mathbf{p}(t)$ that biases e negatively or positively. For example, an interrogator could prime an abductor to remain agitated ny irritating the subject at t_a, when his anger began to subside.

(b) Force an emotional oscillation by periodically perturbing emotional degree $\mathbf{e}(t)$ at t_b, when the derivative de/dt (momentum) has the appropriate sign, (e.g., positive perturbation of e by $\mathbf{p}(t)$ to increase e).

Our model implies that (1) an interpreter must estimate α, ν, or τ to apply stimuli that elicit a desired emotional response and (2) entities' response can be characterized by one emotional model. The first assumption is tenable for trained therapists, negotiators, etc. The second assumption indicates a weakness of our approach, i.e., entities must have the same paradigms of emotional response and similar ontological perspectives. Although this may hold in for-profit and child abduction cases, it is unlikely in terrorist abductions, given abductors' highly contrived ontologies (e.g., fanatical religious beliefs) or disparate cultural backgrounds. Additionally, in alien abduction accounts, Mack [9] has emphasized the role of ontological disparities in motivating lack of correspondence between physical reality and descriptions of abduction features. As mentioned in Section 3.3, HEART-III is equipped with constructs for describing such ontologies.

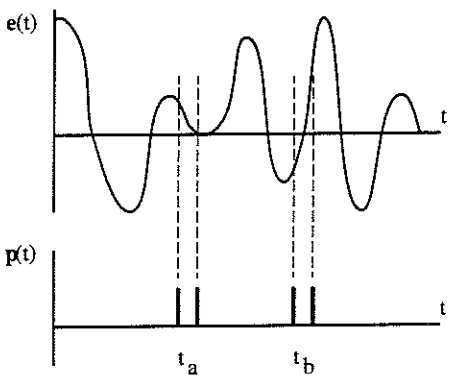


Figure 2. Temporal perturbation of emotional degree e(t) by a perturbation vector p(t), to effect a positive bias in e.

4. IMPLEMENTING AN ABDUCTION KB

We are currently designing and implementing an abduction KB per the model of Section 2.1, whose components and subtasks have slots that describe salient circumstances. We plan to use this KB as a basis for analyzing and comparing NL abduction accounts. For example, a terrorist abduction could have as an *impoundment facility* a room in an unknown location. The room could have *furniture* such as chairs, a table, or a bed, as well as *appliances* or *utensils* such as tableware, a coffee pot, and a small stove or hotplate. Such descriptions are often elaborate in for-profit or child abductions, where abductees can be detained in hotel rooms or homes. Additionally, alien abduction accounts often describe room interiors including detail such as symbols on equipment, which are recalled under hypnosis. Such data have been used to determine commonality of place or experience [9], but can be subject to interpretation and priming.

Additionally, emotional KBs can depict abduction experiences as processes having hierarchical or sequential organization. Although the deduction of event causality from temporal reasoning is gradually being established in automated analyses of KB data, event causality does not imply emotional causality. That is, if event A causes event B, the emotional changes in entities associated with event B may not be predictable solely from existing emotional state and the emotional perturbations associated with event A. For example in Figure 1, the agent (Byck) interacts with recipients (hostages), observes the fatherland being "ravished", and observes his emotions. Thus, Byck's emotional state is not due solely to his planned hostage-taking. Additional non-physical factors are involved, for example, in child abductions, where innate fears and characteristic sensitivity to others' emotions may influence a child's behavior.

5. CONCLUSIONS AND FUTURE WORK

We have presented work in progress pertaining to emotional modelling of four types of human kidnapping given natural-language (NL) accounts of abduction events. We process the NL to yield data structures from which conclusions about emotional perturbation and flow can be drawn. Our emotional model applies such perturbations to recipients (e.g., hostages) and to agents (abductors). If human interpretation or interaction is being modelled, one or more interpreters are also perturbed. As we showed, the resultant emotional state and flow diagram can be used to resolve NL ambiguities that frequently occur in abduction accounts, whose narrative can be fragmented by experiencers' emotions.

We plan to apply this technique to develop semi-automatic methods for analyzing textual accounts of terrorist, for-profit, and child abductions. Given a KB of abduction features, classified by type (e.g., *terrorist kidnappings, for-profit abductions, child abductions*, and the bizarre cases of *alien abductions*), we would first classify an abduction account. Then, the reported experience would be characterized via the featural model given in Section 2.1. By resolving ambiguities as shown in Section 3, emotional flow can be clarified, and conclusions could be drawn about an abductor's motives and emotional/mental state.

Future work will concentrate upon building an abduction KB from a wide variety of sources. Additionally, we plan to implement consistency checking beyond the linguistic level, to facilitate cross-correlation of accounts for investigative purposes. A further goal is the development of a systematic representation of ontological bases for paradigms of emotional perception and expression. Although this will be a long-term effort, due to the work involved in KB construction, such work may further advance the art of determining deep semantics from natural language.

6. REFERENCES

[1] Persinger, M.A. "Neuropsychological profiles of adults who report 'sudden remembering' of early childhood memories: Implications for claims of sex abuse and alien visitation/abduction experiences", *Perceptual and Motor Skills* **75**(1):259–266 (1992).

[2] Hedges, L.E. *Remembering, Repeating, and Working Through Childhood Trauma: The Psychodynamics of Recovered Memories, Multiple Personality, Ritual Abuse, Incest, Molestation, and Abduction*, Northvale, NJ: Jason Aronson (1994).

[3] Schmalz, M.S. and D.D. Dankel. "HEART — A model of emotion for natural language interpretation. 3. Mechanisms for Emotional Reverberation, Decay, and Modulation", in *Proceedings FLAIRS '96 Conference on Knowledge-Based Systems* (1996).

[4] Schmalz, M.S. and D.D. Dankel. "HEART — A model of emotion for natural language interpretation. 1. Background and model requirements.", in *Proceedings FLAIRS '95 Conference on Knowledge-Based Systems,* (1995).

[5] Schmalz, M.S. and D.D. Dankel. "HEART — A model of emotion for natural language interpretation. 2. Model specification and integration.", in *Proceedings FLAIRS '95 Conference on Knowledge-Based Systems,* (1995).

[6] Lahey, B.B. and A.E. Kazdin. *Advances in Clinical Child Psychology, Volume 12*, New York: Plenum (1989).

[7] Greif, G.L. and R.L. Hegar. "Parents whose children are abducted by the other parent: implications for treatment", *American Journal of Family Therapy* **19**(3):215–225 (1991).

[8] Dittburner, T.L. and M.A. Persinger. "Intensity of amnesia during hypnosis is positively correlated with estimated prevalence of sexual abuse and alien abductions: Implications for the False Memory Syndrome", *Perceptual and Motor Skills* **77**(3):895–898 (1993).

[9] Mack, J.E. *Abduction: Human Encounters with Aliens* (revised edition), New York: Ballantine (1994).

[10] Spanos, N.P., C.A. Burgess, and M.F. Burgess. "Past-life identities, UFO abductions, and satanic ritual abuse: The social construction of memories", *International Journal of Clinical and Experimental Hypnosis* **42**(4):433–446 (1994).

[11] Personal conversation with Walter Andrus, Director, MUFON (1995).

[12] Calvin, F.J. "Terrorism and Hostage Taking", in *Suicidology: Essays in Honor of Edwin S. Shneidman*, Northvale, NJ: Jason Aronson (1993).

[13] Miron, M.S. and A.P. Goldstein. *Hostage*, New York: Pergamon (1979).

[14] Hegar, R.L. and Greif, G.L. "How parentally abducted children fare: An interim report on families who recover their children", *Journal of Psychiatry and Law* **21**(3):373–383 (1993).

[15] Excerpted from references in Forst, M.L. and M.E. Blomquist. *Missing Children: Rhetoric and Reality*, New York: Lexington/Macmillan (1991).

[16] Hopkins, B. *Missing Time*, New York: Ballantine (1981).

[17] Powers, S.M. "Fantasy proneness, amnesia, and the UFO abduction phenomenon", *Dissociation: Progress in the Dissociative Disorders* **4**(1):46–54 (1991).

[18] Finkelhor, D., G.T. Hotaling, and A.J. Sedlak. "The abduction of children by strangers and nonfamily members: Estimating the incidence using multiple methods", *Journal of Interpersonal Violence* **7**(2):226–243 (1992).

[19] Burton, A. *Urban Terrorism*, New York: Free Press (1975).

[20] van Beek, P. "Reasoning about qualitative temporal information", *Artificial Intelligence* **58**:297–326 (1992).

[21] Bertrand, L.D., N.P. Spanos, and H.L. Radtke. "Contextual effects on priming during hypnotic amnesia", *Journal of Research in Personality* **24**(3):271–290 (1990).

But "propeller" is a verb!
Automatic tagging and noun/verb confusions

Lois Boggess and Lynellen D.S.P. Smith
Mississippi State University
lboggess@cs.msstate.edu perryl@cs.msstate.edu

Abstract

Most rule-based part-of-speech taggers assume that a novel vocabulary word is a noun, and this is probably the least damaging guess. However, when a novel main verb is mistagged as a noun, the whole sentence may be lost to a knowledge extraction system. When this occurs for several sentences in a row, a significant portion of the text may be lost. We report that an artificial neural network is able to take advantage of extended neighborhoods of words (six words to the left and right of a candidate word) to determine whether it is a noun, verb, adjective, or adverb, with excellent results on novel vocabulary.

1 INTRODUCTION

Recent years have seen increasing popularity of part-of-speech taggers, both stochastic [1], [5], [7] and rule-based [3], in the natural language processing community. These taggers claim success rates as high as 97%+ [4], can be trained to the sublanguages of various domains, and are able to produce reasonable hypotheses of the parts of speech of never-before-seen vocabulary. Consequently, they are a popular alternative to hand-crafted lexicons and machine-readable dictionaries in real-world tasks.

The value of trainable part-of-speech taggers is easily illustrated: in our application domain (millions of words from physical chemistry journal articles) we encountered the word "propeller" on several occasions. It was always a verb. (Apparently it is possible for small components of a molecule to propeller around an axis during a chemical process.) Yet the most recent edition of the American Heritage dictionary lists "propeller" as noun only. The difference between the semantics, the syntax, and the parts-of-speech in a general language such as common English, and the specific sublanguages of highly technical and literate subpopulations is widely known (Grishman and others have more than a decade of published research on the topic).

Proceedings of the 9th Florida Artificial Intelligence Research Symposium, 1996, by the Florida AI Research Society
0-9620-1738-8/96/511

Not surprisingly, then, these taggers are being used as the base step in parsers, and especially in the relatively shallow parsers gaining popularity now. When parsing is done for the purpose of information extraction, we find that some tagging errors are of little significance, while others are disastrous. In our experience, one of the most frustrating errors to which the widely respected Brill tagger is prone (and we would suspect the other taggers are, as well) is the mistagging of main verbs, when the main verb is novel to the tagger.

As the tools and techniques of natural language research disseminate into real-world applications, we believe that many groups will train their taggers and parsers on bodies of text from restricted domains. In our research, we trained on a body of 160,000 words, tagged and hand-corrected. We also have a large auxiliary lexicon, mostly nouns, of words known to be important in the domain, and a pre-processor that identifies most chemical formulae and other items specific to the domain of physical chemistry. Nevertheless, previously-unseen vocabulary occurs at an average rate of several words per sentence.

2 THE PROBLEM

Because nouns occur more frequently than any other part-of-speech category, most taggers have a heavy bias toward labeling an unfamiliar word a noun. Our experience suggests that this is usually the least damaging guess. To see why, consider that sentences tend to have many noun phrases (e.g., the first sentence of the preceding paragraph contains eight to eleven noun phrases, depending on how people count certain constructions). Adjectives, nouns, **and verbs** occur as modifiers in these noun phrases. When one of these modifiers is mislabeled as a noun, the damage done is minuscule, in practical terms of recovering information from the tagged text. The noun phrase in which the mistagged term occurs is undisturbed, and the term is perceived to be fulfilling the appropriate role (modification).

To emphasize an important point: *verbs occur more often as modifiers than as main verbs* in our large corpus of texts.

On examining the tagging errors that occur, our reaction has been that our rule-based tagger almost always mistags novel verbs. This is a mis-perception.

We calculated sample error rates and hand-checked the results, giving the tagger the benefit of the doubt whenever it applied a tag to a word that could plausibly have been labeled either adjective or participle. The tagger's success rate on novel verbs is approximately equal to its error rate on other parts of speech. *However, its success rate on novel verbs in main verb position is in the neighborhood of 50%.* This fact has major implications.

```
(verb_phrase ((w converts VBZ)))
(noun_phrase ((w part NN))
   (prep_phrase ((w of IN)
      (noun_phrase ((w the DT) (w surface NN))
         (prep_phrase ((w to IN)
            (noun_phrase
               (conjunct (w and CC)
                  (noun_phrase ((w AgSe NN)))
                     (noun_phrase ((w releases NNS))
                  )))))))
```

Example of main verb mistagged and resulting misparse of sentence fragment

Our hope was that occasionally losing a main verb would rarely have a major impact on the extraction of information from the text. [7] shows a success rate of 80% in extracting the same concepts from technical text as were extracted by expert document analysts, despite loss from tagging errors. This result supports our hypothesis that, in systems (like ours) based on shallow parsers, if most noun phrases are of good quality then the natural redundancy of language allows much of the target information to be extracted.

On the other hand, systems (like ours) trained on only one- or two-hundred-thousand words from the domain encounter novel vocabulary frequently. In most trial data sets that we examined, we easily found sentences in which none of the main verbs were tagged as such, and sequences in which up to three consecutive sentences had had their main verbs mistagged. The result was that a major block of information was lost from the body of an article. Sometimes the problem is worse than simple loss of information. A typical consequence of a shallow parsing of a sentence with a lost verb is that a prepositional phrase doesn't find the verb to which it should be attached, and instead attaches to a more distant noun phrase, resulting in a structure whose semantic content is far removed from the intention of the author.

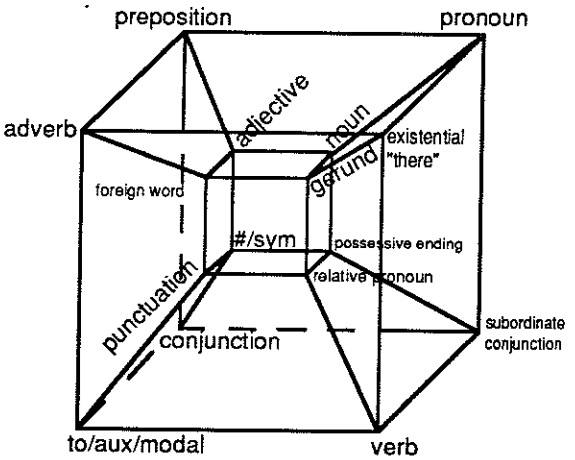
3 A GRAY CODE INSPIRED SOLUTION

One approach we are developing to address the problem of mistagging novel verbs in main verb position is to widen the tagging context window. Like others, we postpone the combinatorial explosion which results from examining the larger context by looking at tag sequences rather than lexical sequences. In the experiments reported below, we lower the dimensionality of the problem space further by mapping the Penn Treebank part-of-speech tags onto a four-bit Gray code.

0000	noun
0001	pronoun
0011	subordinate conjunction
0010	possessive ending
0110	relative pronoun
0100	gerund
0101	existential "there"
0111	verb
1111	to/aux/modal
1110	punctuation
1100	foreign word
1101	adverb
1001	preposition
1011	conjunction
1010	number/symbol
1000	det/adj/possessive pronoun

One possible Gray code for syntactic tags

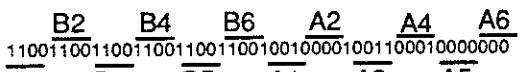
In the associated code space, the concept *noun* is a distance of 1 from related concepts such as *pronoun, adjective, gerund,* and *possessive ending.* The concept *verb* is maximally close to *existential "there"* and *to/aux/modal,* while the latter is maximally distant from the *noun* class, using the Manhattan distance on the corresponding 4-cube, which is easily calculated by counting the number of different bits. Part of speech tags encoded by a Gray code are easily input to an artificial neural network whose purpose is to automatically discover the patterns implicit in the tag sequences and disambiguate between the neighborhoods of open classes. (The "open classes" are nouns, adjectives, adverbs and verbs, in contrast to the relatively stable "closed" classes such as the prepositions, conjunctions, auxiliary verbs, pronouns, and so on.)



Gray Code Four Cube

We have conducted two experiments, giving Gray code tag input to standard back-propagation neural networks with one hidden layer. In each case, the network was fully connected, weights were initialized to a random value between -0.25 and +0.25, there was no momentum applied, and the learning rate was 0.35. The input data were neighborhoods consisting of six

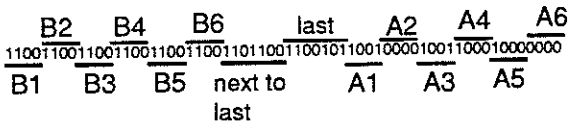
Gray coded tags on each side of every noun, adjective, adverb, or verb from our 160,000 word hand-corrected training corpus. Ten per cent of the corpus was held out and the other 90% was used for training the neural network. The training data were divided into nine files, each of which contained approximately 4500 training pairs. The held out test data also contained approximately 4500 testing pairs. The network was given the Gray coded tag sequences only - no lexical information. It had no access to a lexicon and hence had no distinctions between known and unknown words. Indeed, it had no concept of words at all. It merely chose what it thought the most likely Gray coded tag would be for a given combined left- and right- neighborhood comprised of 12 Gray code tokens, where the expected output was the Gray code for a noun, a verb, an adjective, or an adverb, as appropriate.



Format of Input to Neural Network

The network was correct about 75% of the time, for all four categories. On inspection, we realized that the most frequent context in which all four categories occurred was (of course!) in a noun phrase. Most positions of a noun phrase allow the appearance of adjectives, verbs, and modifying nouns. That is, there is a high inherent error in this task given to the neural network, and it may be nearly impossible to improve on the error rate for the initial problem posed. On the other hand, both the stochastic tagger that we have used in the past and Brill's rule-based tagger use information directly from the lexeme being tagged. To overcome the inherent error in the problem, we added the last two letters of each target word to the information given to the neural network.

In this second experiment, the input to the neural network was a Gray code sequence for the six tags before the target word, the seven-bit ASCII code for the last two letters of the target word, and then the Gray code sequence for the six tags following the target word. The expected output was again the Gray code corresponding to a noun, verb, adjective, or adverb tag. To increase the neural network's ability to classify the extra information, and to aid it in achieving a lower error rate, the number of units in the hidden layer was increased from the first experiment. Three configurations of the network were trained, to discover which had enough power to train quickly despite the extra computation needed due to the extra units in the hidden layer.



Format of Input to Neural Network

4 RESULTS

The 100 hidden unit network achieved 91% accuracy on the held-out testing data after 940 epochs of training, where an epoch was one pass through 11% of the training data (the training data was held in 9 files of the same size, so each file represents 11% of the total training data available). The 200 hidden unit network achieved 97% accuracy on the held out testing data after 1100 epochs of training, and the 300 hidden unit network achieved 97% accuracy after 590 epochs of training.

We are optimistic about the results obtained thus far for several reasons:

First, even without additional information about the last two letters of each target word, a neural network appears to be outperforming the rule-based tagger for correctly identifying unknown verbs in main-verb position. We surmise that the reason for this is suggested by a previous exploratory experiment. We color-coded the reduced tag set (allowing the colors for verb-ish tags to be in one color range and the colors for noun-ish tags to be in a contrasting color range) and displayed the results on a screen. There were clearly visible color patterns for neighborhoods of main verbs that were distinct from patterns for neighborhoods of nouns. Many of these patterns were longer than could be captured by a window of only two words, as used by most taggers today.

Second, the neural network approach is an automated, domain independent method of finding probable mistaggings. The neural net is labeling only open class words, in contrast to conventional taggers, some of whose success rates are for labeling the fixed sets of prepositions, pronouns, and other frequently occurring closed class words. The neural network neither uses nor needs a lexicon, so there is no distinction between previously known and unknown words. Finally, the network uses tags and two-letter suffixes as input, rather than whole words, so we are able to contain the combinatorial explosion which threatens when using a larger neighborhood.

5 RELATION TO OTHER WORK

In a paper presented to this conference two years ago [2], a system combining a neural network with a conventional probabilistic part of speech tagger was described. While that system did exhibit mildly better performance than a conventional tagger on unknown vocabulary, its error rate was nevertheless nearly five times that of the neural network described here. It should be pointed out that in the earlier system, all contextual information was delegated to the stochastic portion of the tagger, which used a window of at most three words.

Our experience may relate to a finding [10] that adding a hidden layer to a neural network did not improve performance in tagging text with part of speech. In the just-cited work, we believe that much

of the performance of the system is attributable to a very well designed suffix-handling algorithm and data structure. In that work, both a perceptron and a 50-hidden-unit network were trained and tested. No improvement was found with the addition of the hidden layer. The fact that our best results are found with much larger hidden layers suggests that there may simply not have been enough hidden units to realize a benefit in that particular effort.

6 FUTURE WORK

Our ultimate goal is not to replace our existing tagger or to out-perform it using a competing tool, but to supplement it. The existing tagger does comparatively well when tagging tokens in the scope of a noun phrase. The neural network is presented with its most challenging task (distinguishing among the possible tags for modifiers of nouns) in exactly that environment. On the other hand, the neural network approach seems to be reflecting our belief that a larger context will help identify main verbs - the serious difficulty that prompted our exploration of this problem in the first place. Our goal is to drastically reduce loss of main verbs by having a tool that can recognize the contexts in which it is appropriate to override the basic tagger and dictate that the token be labeled as a verb.

We also have plans to run further experiments with the neural network configuration. It has been suggested [9] that, instead of training the weight connections between the nodes, an evolutionary approach can help to overcome the problem of the network's getting caught in a local minimum, in addition to providing smaller training times. Fractal edge connections of a neural net are also reported [8] to provide improved performance in terms of reducing the overall error rate. We plan to experiment with both of these modifications to the standard back-propagation neural network.

This whole line of research originated in investigations into the self-similarity of natural language; we plan to continue exploring the utility of a Gray code for discovering this self-similarity. We anticipate that tools which automatically discover and exploit self-similarity will benefit application areas such as part-of-speech tagging, knowledge extraction, and data compression.

6 CONCLUSIONS

A conventional part-of-speech tagger which uses a limited context window has a high error rate when attempting to tag previously unseen vocabulary. When main verbs are mistagged, subsequent parsing and knowledge extraction efforts are often doomed to failure. In this paper we report encouraging results in the use of an artificial neural network for using extended neighborhoods to improve tagging of nouns, verbs, adjectives, and adverbs, based on their tag neighborhood. A neural network is well suited to finding and utilizing the complex patterns inherent in tag neighborhoods for suggesting the correct part-of-speech tag for open-class words in a real world corpus of scientific journal article text.

ACKNOWLEDGMENTS

This research was partially funded by the National Science Foundation and by ARPA through NSF grant #IRI-9314963. The Chemical Abstracts Service of the American Chemical Society graciously provided the journal articles comprising the physical chemistry domain corpus. We are indebted to Rajeev Agarwal for setting up our base-line tagging system, and to the members of the KUDZU research group for many ideas that have influenced our current research.

References

[1] Boggess, Lois, Rajeev Agarwal, and Ronald Davis. 1991. Disambiguation of prepositional phrases in automatically labeled technical text. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, Menlo Park: AAAI Press/The MIT Press, pp. 155-159.

[2] Boggess, J. E. and Lois Boggess. 1994. A hybrid probabilistic/connectionish approach to automatic text tagging. In *Proceedings of the Seventh Florida Artificial Intelligence Research Symposium*, Pensacola Beach, FL, pp. 147-51.

[3] Brill, Eric 1992. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, Association for Computational Linguistics, pp. 152-155.

[4] Brill, Eric 1994. Some advances in transformation-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence,* Menlo Park: AAAI Press/The MIT Press, pp. 722-727.

[5] Cutting, D., J. Kupiec, J. Pedersen, and P. Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, Association for Computational Linguistics.

[6] Hodges, Julia E., Shiyun Yie, Ray Reighart, and Lois Boggess. 1995. An automated system that assists in the generation of document indexes. Technical report #950712, Mississippi State University. http://www.cs.msstate.edu/PUBLICATIONS/technicalReports.html#1995

[7] Kupiec, Julian. 1992. Robust part of speech tagging using a hidden Markov model. *Computer Speech and Languague.* Volume 6, pp. 225-242.

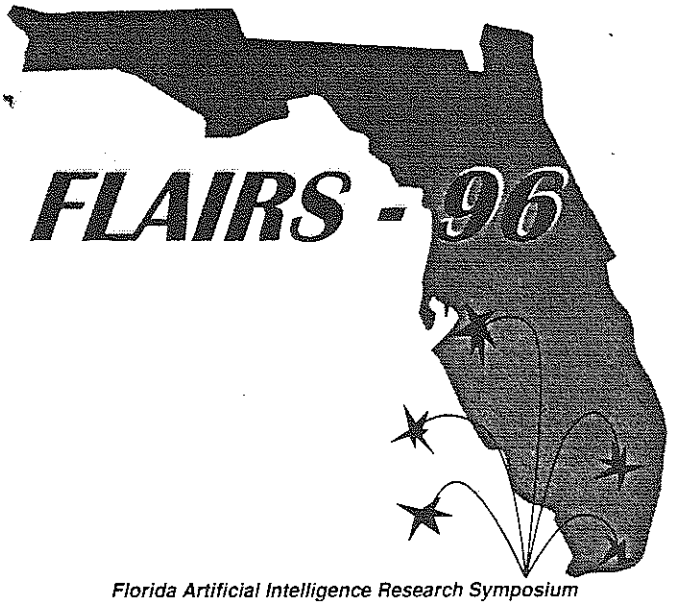
[8] Merrill, John W. L., and Robert F. Port. 1991. Fractally configured neural networks. *Neural Networks.* Volume 4, pp. 53-60.

[9] Porto, Vincent W., and David B. Fogel. 1995. Alternative neural network training methods. *IEEE Expert*. June, pp. 16-22.

[10] Schmid, Helmut. 1994. Part-of-speech tagging with neural networks. In *Proceedings of the 15th International Conference on Computational Linguisitics (COLING-94)*, Kyoto, Japan, pp. 172-76

Author Index

Keyword Index



Florida Artificial Intelligence Research Symposium