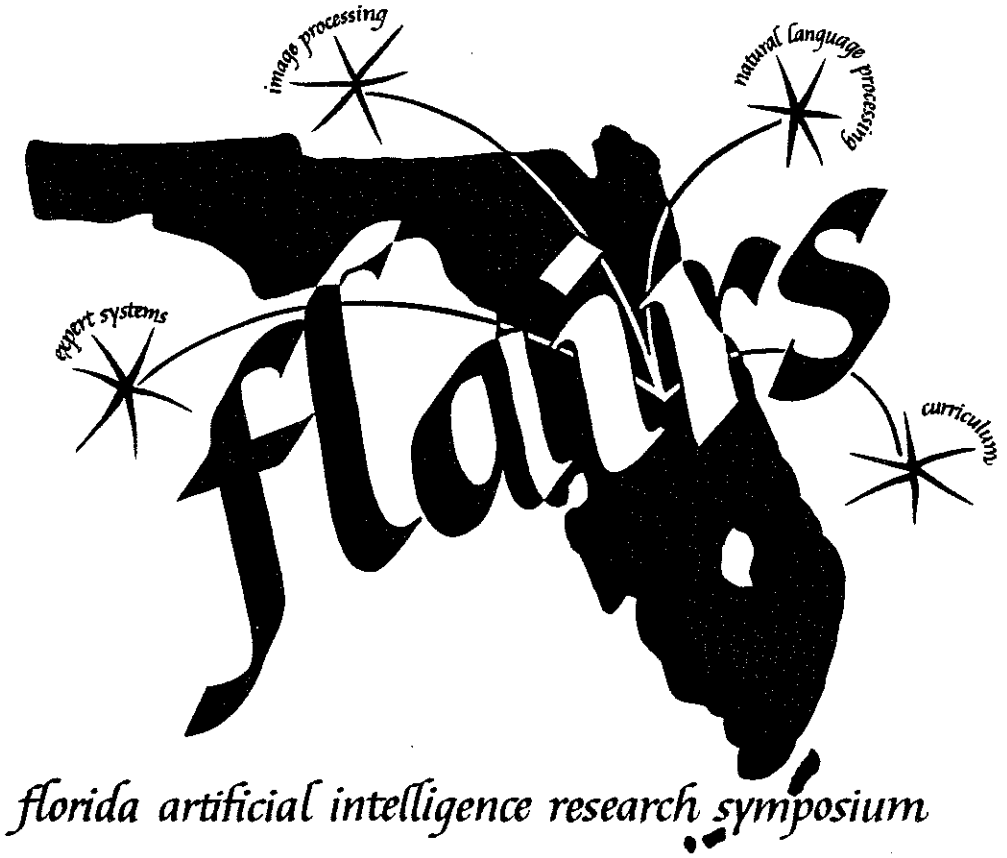
PROCEEDINGS OF THE SECOND FLORIDA ARTIFICIAL INTELLIGENCE RESEARCH SYMPOSIUM

Orlando, Florida
April 3-6, 1989

Mark B. Fishman
Editor

PROCEEDINGS OF THE 2ND FLORIDA ARTIFICIAL INTELLIGENCE RESEARCH SYMPOSIUM



Orlando, Florida
April 3-6, 1989

Mark B. Fishman
Editor

The Florida AI Research Symposium

P.O. Box 12560

St. Petersburg, Florida 33733

ISBN 0-9620-1731-0

Additional copies may be ordered prepaid from

Mark Fishman
P.O. Box 12560
St. Petersburg, FL 33733

Price: $25.00

FLAIRS-89
2nd Florida Artificial Intelligence
Research Symposium

Welcome to the (mirabile dictu!) SECOND meeting of FLAIRS! In the wake of last year's large and very enthusiastic turnout, we had considered renting a larger facility -- Iceland, for example -- but concluded eventually that the Holiday Inn in Orlando would continue to do nicely. We couldn't, in any case, figure out how to say, "The Neural Nets tutorial is in Ballroom B," in Icelandic.

For those of you attending for the first time, FLAIRS was organized last year by a group of AI researchers and professionals in academe and in industry, with the intent of establishing a permanent network of activity for the Florida AI community. The overwhelming success of last year's symposium attests to the considerable vitality of that community, and we hope this year to stimulate an even higher level of interest.

The theme of FLAIRS-89 is knowledge engineering, and in addition to a broad diversity of high-quality papers reflecting that focus, we are especially pleased this year to boast four speakers who are nationally known AI experts, and a full roster of tutorials on the various AI sub-disciplines. We are pleased also to be able to offer a full exhibit hall, and hasten to proclaim our appreciation to the very supportive exhibitors listed on the next page.

Thanks, finally, to our tireless Organizing Committee - the people who multitasked valiantly against the onslaught of time, the non-onslaught of money, and the generalized encroachments of entropy in all of its guises. And special thanks to you, the authors and attendees, whose participation has been the essential element to help make Florida an even more important AI center.

Mark Fishman
Proceedings Chair
for the
FLAIRS Organizing Committee 12

The FLAIRS Organizing Committee

General Chair: Avelino Gonzalez - UCF

Program Chairs: Doug Dankel, II - UF and Ken Ford - UWF

Promotion Chair: Jim Ragusa - UCF

Speaker Arrangements and ProceedingsChair: Rita Rodriguez - FIT

Proceedings Chair: Mark Fishman - Eckerd College

Financial Chair: Don Thomas - Cognitive Engineering International

Tutorials Chair: Dave Lawson - Stetson University

Industrial Liaison Chair: Phil Koltun - Harris Corporation

Panel Arrangements Chair: Dennis Murphy - Nova University

Awards Chair: Carlos Segami - UCF

Administrative Chair: Vince Amico - UCF

Abstracts of papers were carefully reviewed for presentation at FLAIRS-89 and inclusion in this set of proceedings. We thank our referees:

Frank Anger - Florida Institute of Technology

Patrick Bobbie - University of West Florida

Doug Dankel, II - University of Florida

Mark Fishman - Eckerd College

Ken Ford - University of West Florida

Fred Hoffman - Florida Atlantic University

Rita Rodriguez - Florida Institute of Technology

Carlos Segami - University of Central Florida

Special thanks to our sponsors and exhibitors

Conference Sponsors

Florida High Technology and Industry Council
University of Central Florida Division of Sponsored Research
Harris Corporation
Florida Institute of Technology
Eckerd College

Exhibitors

Addison-Wesley Press
Advanced Technologies, Inc.
Apollo Computers, Inc.
Gold Hill Computers
IBM
Intellicorp
Micro Devices
Symbolics
Texas Instruments
University of Central Florida Computer Engineering

...and to all of the following:

Dean Lloyd Chapin of Eckerd College, for continuing to tolerate and support the huge burden on my time, and for making Eckerd funds available.

Je-Nien Ernst, Gayle Green, Kristin Magnusen and Quinton Zondervan, Eckerd students who labored indefatigably and in the face of rubber cement fumes to complete much of the paste-up.

Roberta DiMouro, for fielding innumerable calls from authors, and stoically enduring the overhead of FLAIRS, while serving as secretary to a collegium of twenty faculty members.

Barbara McMahon of UCF Instructional Resources and Jacqui Permaul of UCF Printing Services, for work both on the announcement and the program.

...and to the deans, department chairs and corporate managers who allowed the steering committee members, forsaking much else, to devote their time and effort to the birthing of this conference.

Table of Contents

Session C2 - Neural Networks I

Chair: Dave Lawson

Session C3 - Applications

Chair: Bob Peart, UF

Session D1 - Knowledge Acquisition II

Chair: Janice Eisele, General Electric

Session D2 - Reasoning

Chair: Florin Zeviar, Boeing

Session E1 - Logic

Chair: Robert Osborne, Westinghouse

Session E2 - Robotics

Chair: Harley Myler, UCF

Session E3 - Real-Time

Chair: Abraham Kandel, FSU

Session F1 - Neural Networks II

Chair: E. K. Evinson, AT&T

Session G1 - Knowledge Representation

Chair: Mark M. Thomas, Automation Software Innovations

Session G2 - Inference Methods

Chair: Mark Sorrells, Symbolics

Session H1 - Knowledge Acquisition III

Chair: Robert Ahlers, Naval Training Systems Center

Session H2 - Natural Language

Chair: Stanley Green, McDonnell Douglas

Poster Session

Reasoning with Uncertainty in Knowledge-Based Systems

Ronald R. Yager

Machine Intelligence Institute
Iona College
New Rochelle, NY 10801

Abstract

A central concern for the development of knowledge-based systems is the construction of a mechanism to reason in the face of uncertain information. Uncertainty appears in many forms. A most common form of uncertainty is that of randomness. Ambiguity and fuzziness in language and concepts are another form of uncertainty. Central to this type of uncertainty is the notion of partial satisfaction. Lack of specificity is another type of uncertainty. An additional form of uncertainty is manifested by typicality. This appears in many cases as a form of nonmonotonicity.

In this talk we discuss a system for reasoning under uncertainty which provides an ability to represent and manipulate the various types of uncertainty described above. This system, which uses sets as its primary objects, can be seen as an extension of the theory of approximate reasoning. It includes probabilistic knowledge in manner closely related to the Dempster-Shafer theory of evidence. It includes a facility for reasoning with default rules with the additional power of being able to reason with partial satisfactions in default rules. A means is also provided for implementing logical connectives which lie between "and" and "or" via OWA operators. This ability allows us to have a simple means for representing linguistic quantifiers such as "most", "some", etc.

Biographical Sketch

Ronald R. Yager is Editor-In-Chief of the *International Journal of Intelligent Systems* and author of more than a hundred artificial intelligence journal papers. He is the Director of the Machine Intelligence Institute at Iona College in New Rochelle, New York.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

There Is More to Logic than Predicate Calculus

John F. Sowa

IBM Systems Research Institute
Thornwood, NY 10594

Abstract

Predicate calculus is the assembly language of knowledge representation. Like assembly language, it is low level and general. But also like assembly language, it is verbose, unreadable, and error prone. In artificial intelligence, the pure predicate calculus notation is used only in theoretical studies. Every system used for representing large knowledge bases either extends it or replaces it with another notation, such as frames, production rules, stylized English, or some sort of networks or diagrams. Even C. S. Peirce, the inventor of the predicate calculus, abandoned it in favor of graphs. All these considerations suggest that there is something fundamentally wrong or at least unnatural about the structure of predicate calculus as a knowledge representation language. This paper argues that conceptual graphs, an extension of Peirce's existential graphs, are more concise and natural as a system of logic for AI. The advantages are especially prominent in the mapping to and from natural languages, the treatment of context and discourse phenomena, and the representation of sets.

Biographical Sketch

John Sowa is a consulting staff member at the IBM Systems Research Institute. He has been working in the AI field since 1971, designing natural language and expert systems tools, teaching courses on AI and computational linguistics, and doing research on the theory of conceptual graphs. His book, Conceptual Structures, published by Addison-Wesley, presents conceptual graphs as a formal knowledge representation language that has been adopted by a number of research and development groups around the world. Besides conceptual graphs, he has also published articles on logic programming, knowledge engineering, and the integration of AI and database tools.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

Memory-Based Reasoning

David L. Waltz

Thinking Machines Corporation
Cambridge, MA 02142
and
Brandeis University
Waltham, MA 02254

Abstract

Memory-based reasoning (MBR) is a class of nearest-neighbor methods for producing expert system-like behavior. MBR methods are a natural match for massively parallel machines: In a typical implementation, a database of previous decisions is stored, one per processor, on a massively parallel machine. A new item is classified by finding its nearest neighbor in the database and either using the decision associated with this nearest neighbor or interpolating between the n nearest neighbors. Unlike traditional statistical methods, classifications are made on the fly. Memory-based reasoning has been applied to protein-structure prediction, medical diagnosis, word pronunciation and operating system troubleshooting; MBR variants have been successfully used in systems for document retrieval and object classification. Key MBR research areas include the similarity metrics used to match new items with old, and learning or preconditioning methods for generalizing the database, and system architectures that use MBR modules as components. MBR also bears some interesting relationships to connectionist and neural-net systems, and has been advanced as an alternative to heuristic search for the central paradigm of AI and cognitive science.

Biographical Sketch

David Waltz is the Director of Advanced Information Systems at Thinking Machines Corporation and a Professor of Computer Sciences at Brandeis University. He received a Ph.D. from MIT in electrical engineering and artificial intelligence in 1972, and also holds an M.S. and a B.S. from MIT, both in engineering. Until 1984, he was Professor of Electrical and Computer Engineering at the University of Illinois. He has served as editor of Cognitive Science and as AI editor for Communications of the ACM. Waltz's interests include text processing and document retrieval, massively parallel memory-based reasoning and connectionist systems, and constraint-propagation models for computer vision and other AI tasks.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

Parsing Pragmatics in Natural Language Dialogue

John W. Roach

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061

Abstract

Natural language parsing systems have concentrated on cognitive issues in understanding. Issues of status, politeness, affection (indeed, all emotions) have been ignored. Many hard problems, such as indirect requests and intentionality, have therefore been approached as a purely cognitive issue. To rely entirely, even primarily, on cognitive processing to solve the entire natural language parsing problem ignores both behavioral evidence that contradicts this approach and established theories of human-human communication.

In this talk, we shall introduce a new system based on human communication theory that emphasizes pragmatics as the correct approach to solving many difficult natural language issues. Our work posits that indirect questions, emphasis, focus and speakers' goals can largely be determined using structural means and uniform world knowledge of social situations. A system we have built (in Prolog) can correctly play the part of an airlines reservation agent in natural language dialogues with customers; the system has been thoroughly tested using actual dialogues recorded from the telephone. The system is able to handle indirect requests, changes of emphasis and focus, and determination of speakers' goals.

Our system is based on the human communication theory known as "metacommunication" in which every utterance is said to consist of both a semantic, content portion and a communication management portion. Communication management is used by a speaker to guide the listener's attention, indicate importance of topics and in general to maintain a shared communication context, which includes maintaining the proper social relationship with appropriate politeness, affection and status.

We have discovered that in request structures people encode metacommunication using the word order and sentence structure in which the request is phrased. There must be a reason, after all, for being able to form a request in thousands of different ways, all of which have the same semantic content.

Our system has demonstrated the richness of request forms by actually generating thousands of distinct forms for a request such as "what time is it?" in which each form differently encodes information about status, politeness, emphasis, etc.

DIALS was implemented as the front end of an airline reservation database. It is, however, independent of that task domain and can be used for any task-oriented dialogue problem. People have established a rich variety of pragmatic patterns for expressing a single contentful idea, and they will naturally want to use such a rich collection of patterns when talking (communicating) with computers at a human-computer interface. Metacommunication is therefore a key concept in building an interface that can maintain a shared communication context.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

Biographical Sketch

John W. Roach received his Ph.D. in computer science from the University of Texas, Austin, in June 1980. He joined the faculty of Computer Science at Virginia Tech where he is now Associate Professor. His research interests include counterfactual reasoning, logic programming, natural language processing, planning and vision.

Dr. Roach has recently won major international honors for his work in expert systems and in natural language processing. He won a $90,000 prize from Texas Instruments for application of Artificial Intelligence expert system technology to industrial automation. Two different papers on natural language have also won top honors at the International Communications Annual Conference.

Dr. Roach was general chairman of the IEEE Workshop on Principles of Knowledge-Based Systems and of the Second Conference on Applications of Artificial Intelligence, The Engineering of Knowledge-Based Systems. He was special guest editor for IEEE Expert, Volume 1, Number 3, Engineering Knowledge-Based Systems, and also serves on the editorial board of IEEE Expert.

Dr. Roach is a member of the American Association for Artificial Intelligence, the Association for Automated Reasoning, the Association for Computation Linguistics, the Association for Computing Machinery, the Cognitive Science Society, the Institute for Electrical and Electronics Engineers (Computer Society and Systems, Man and Cybernetics), and the Society of Photo-Optical Instrumentation Engineers.

AN APPROACH TO KNOWLEDGE ENGINEERING TO SUPPORT KNOWLEDGE-BASED SIMULATION OF PAYLOAD GROUND PROCESSING AT THE KENNEDY SPACE CENTER

by

Shawn McManus, Michael McDaniel
McDonnell Douglas Space Systems Company
Kennedy Space Center
P. O. Box 21233
Kennedy Space Center, FL 32815

ABSTRACT

Planning for processing payloads at the Kennedy Space Center (KSC) has always been a difficult and time-consuming task. With the advent of Space Station Freedom and its capability to support a myriad of complex payloads, the planning to support this ground processing maze involves thousands of man-hours of often tedious data manipulation. To provide the capability to analyze various processing schedules, McDonnell Douglas Space Systems Company-KSC is developing an object oriented knowledge-based simulation environment called the Advanced Generic Accommodations Planning Environment (AGAPE). Having nearly completed the baseline system, our emphasis in this paper is directed toward rule definition and its relation to model development and simulation. We focus specifically on the methodologies implemented during knowledge acquisition, organization, and representation within the AGAPE rule structure. An example model is provided to illustrate the concepts presented. Our approach demonstrates a framework for AGAPE rule development to assist expert system development.

I INTRODUCTION

Space station payload ground processing at KSC requires a great deal of planning and analysis, and AGAPE was designed primarily to support this activity. As the system has matured, its capabilities have become quite robust, making the system adaptable to modeling development activity in a wide variety of domains. To highlight and clarify the discussions in the ensuing sections, the major sections will feature a discourse of the relevant topics, followed by two examples from KSC space station ground processing. A future facility at KSC, the Space Station Processing Facility (SSPF) will provide an example of the facilities under consideration in our work, and installing a payload into a rack will benchmark a typical processing sequence.

Some local KSC definitions are in order:

payload - in the context of this paper, this will mean any hardware processed at Kennedy to become part of Space Station Freedom

experiment - any user (i.e. non-system) payload

NSTS - National Space Transportation System, the shuttle program

GSE - ground support equipment

APAE - Attached Payload Accommodation Equipment, the equipment that allows an attached payload to interface with station resources

rack - Structure used in space station modules for holding experiments, similar to instrumentation racks

II SYSTEM OVERVIEW

AGAPE is an knowledge based, object oriented simulation environment. The system's objects are built in a frame structure, a convenient method to allow storage of object attribute values in structures called *slots*. The object oriented approach to programming defines objects into two categories, *classes* and *instances*. Class objects may have any level of descendents (children, grandchildren), while instance objects can have no children. This parent-child relationship allows for *inheritance* of attributes and other structures to be discussed later. This hierarchy

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

constructs a tree-like *genealogy* of objects, as seen in Figure 1.

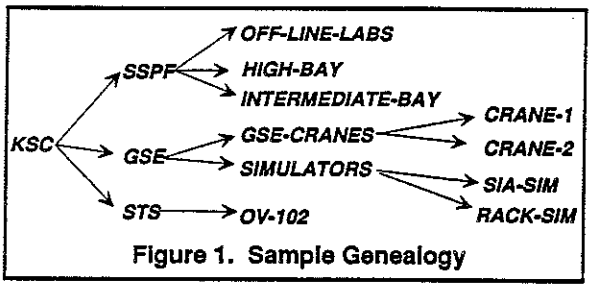


Figure 1. Sample Genealogy

The *knowledge base* resides in the HSKB, or *Hierarchical Segmented Knowledge Base* (Castillo, McRoberts and Sieck 1989). The hierarchy comes from the object oriented nature of the system. Small portions of the knowledge base, called *rule sets*, are attached to appropriate objects (the segmentation of the HSKB), and these rule sets can be inherited from parent to child, just as with attributes. This structure differs from the typical expert system knowledge base, where all rules reside in a single unit. The segmentation of the rule sets allows the simulation to reason over only those portions of the knowledge germane to the situation.

The *script-based* simulation utilized by AGAPE allows each object (or for similar processes, groups of objects) to contain its own activity capabilities and requirements. These scripts are built from three basic types of activities; *task* activities, such as servicing or testing, *transport* activities, to move an object from one location to another, and *installation/deintegration* activities, where an object installs itself in or removes itself from another object. The scripts also allow objects to define resources needed for a particular activity, such as technicians and lift-ing devices for a transport activity. These *resource requests* also provide a method for the script to interact with the HSKB, allowing certain attribute requirements to replace requests for specific objects. To move a 13,000 pound payload, for instance, the script may call for a specific crane, or it may require any lifting device with a capability of more than 6.5 tons.

This coupling of a rule-based system with discrete event simulation (see Figure 2) in a non-programmer environment is one of AGAPE's strengths. With the object oriented aspects of the HSKB blended into the object oriented simulation environment, some of the usual complexities associated with the ideas in Figure 2 are nicely resolved (Castillo et al 1989).

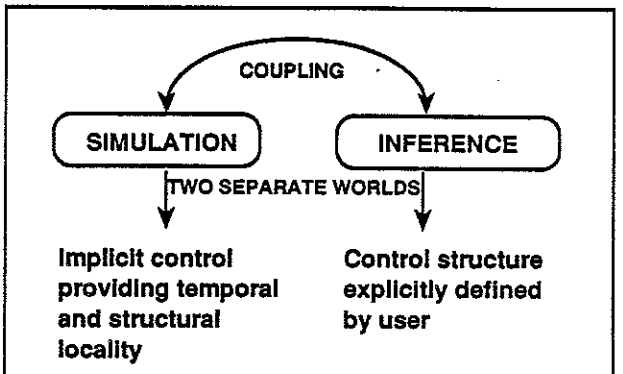


Figure 2. Integrating Simulation and Inference

III KNOWLEDGE ACQUISITION AT KSC

3.1 Ground Processing At KSC

Payload ground processing at KSC involves a series of complex, tightly controlled assembly and test procedures, as the *launch package* is assembled. (A launch package is the set of payloads for one mission, assembled as they would appear in the orbiter's payload bay.) As the payload arrives at the center, it is received and inspected for any shipping damage. The hardware is then taken to an assembly area for build-up to its on-orbit configuration. The payload is tested to assure its functionality, and then verified with the space station hardware it will interface with (rack, APAE, etc). All payloads are tested with space station systems to assure compatibility with such items as power distribution and the Data Management System (DMS). The payloads for a single mission are then assembled into a launch package, and interfaces with the orbiter are verified. The assembled launch package is finally installed into the orbiter and launched. The flow outlined typically lasts from three to eighteen months.

One of the major planning and analysis tasks being performed today at KSC involves the design and review of a new facility, the Space Station Processing Facility (SSPF). The reviews include standard items, such as power outlets and office sizes. Because of the limited resources to process the myriad of Freedom hardware, the review process also includes many utilization studies to assess the impacts of changes in the program and its material.

There are many thousands of single operations required to process any payload at KSC. To adequately model a given process, the proper modeling scope is required to define the precision of the simulation's products. The process

of preparing a flight rack to accept a payload can be laid out in its exact detail (hundreds of steps), the major processes can be defined (10-12 activities), or the overall process can be laid out in one to two steps. The more detailed the model, the smaller the scope of the simulation needed to provide an accurate and meaningful representation. If the modeler wishes to simulate the entire 20-mission build-up of the space station, for example, it is obvious to use the very highest level of overview (the fewest steps). This forces the simulation to concentrate on the overall utilization of facilities and equipment, the obvious point of such a large model.

3.2 Our Model - Flight OF-1

In order to more effectively delineate the processes occurring in the knowledge base development for AGAPE, one of the Freedom assembly flights, OF-1 (OutFitting flight 1, number 6 in the series), was chosen to act as a candidate mission. This flight information is currently taken from the August, 1988 Space Station Trial Payload Manifest (TPM), the most accurate set of flight information available for space station planning. This mission will allow a demonstration of the types and quantities of information necessary to support KSC process planning.

Flight OF-1 has many payloads of several types. The station payloads include a hand and eye wash station, a glovebox for utilization by experiments, and payload racks. The user payloads found on the manifest include some science payloads (e.g. SAAX 307X, life science 1.8m centrifuge), technology development payloads (e.g. TDMXF, fluid behavior experiment), and commercial endeavors (COMM 1243, Electroepitaxial crystal growth).

Although the various payloads employ the same basic interfaces to the space station, many require modified or unique attributes to their processing flows due to owners' preferences or singular payload characteristics. As found in the TPM, there are several items bound for the space station on each flight, all with some level of processing occurring at KSC. Multiply this example by the 20 assembly flights necessary to complete the station, and the enormity of the planning and scheduling at KSC becomes apparent.

IV KNOWLEDGE ACQUISITION

Numerous information sources exist at KSC for outlining *current* standard processing flows. Operations and Maintenance Instructions (OMIs) and Work Authorization Documents (WADs) are two of the sources available to the studious KSC knowledge engineer. The difficulty in determining Freedom payload processing characteristics stems from the high rate of change of space station resource information. The on-orbit resource types and locations differ almost from day to day. The challenge for the modeler is to find the information, as it exists at any given time, and compile it into a meaningful and realistic model of space station processing.

4.1 Acquisition.

Because the models being developed represent a program still in its infancy, many of the documents needed to complete the model have not been created. These new space station documents, however, are nearly all based on one or more *existing* documents used to support NSTS processing. For those items in the model requiring a nonexistent document, relevant current documents can be obtained and used as a basis. Generally, once the space station equivalent of the document has received final approval, the changes necessary to update the 'old' model are relatively minor, and quickly implemented and verified. The benefits of developing a model in an object-oriented environment become apparent at this time, as one change propagates through many descendents, making the change almost instantaneous.

At the time of this writing, the SSPF is undergoing tertiary review of its design. As such, specific items like equipment locations and utility port sizes and types are in a nearly constant state of flux. This makes the modeler's job rather difficult, because attribute values are constantly in need of revision. The basic layout of the building, on the other hand, has remained nearly constant over the past few months.

The data for the SSPF comes from two main types of sources, written and personnel. Documents are released in increments as the review process develops, so changes are manifested only when the next document revision is released. Data from the personnel contacts change almost daily while the review process continues. This makes for frustration as a knowledge engineer, since values and concepts are modified from minute to minute, depending on the person being interviewed. Our experience shows that using values from the documentation leads to the fewest problems. Generally, the people involved with the design reviews know when a change in the documents is about to occur, but these alterations should only

be incorporated after they have been approved by the appropriate personnel.

Ground processing data and procedures are being formulated at this time. All these documents depend a great deal on the actual configuration of space station hardware, which remains in a state of constant change. The data presented is based on the most recent versions of the information.

4.2 Organization.

Because of the characteristics of the HSKB utilized by AGAPE, the organization of data easily falls into a familial grouping, much in the same manner as object-oriented programming. For the most part, rules necessary to accomplish a simulation follow the payload type groups: Payloads in general have different scripts than GSE, rack payloads use different rules than attached unpressurized payloads, etc. Similarities in certain aspects of processing allow for use of the same rules, provided that the rules were written in a generic fashion.

4.3 Representation.

The grouping of rules into rule sets is actually controlled in a large manner by the architecture of the system. Unlike traditional expert systems, the reasoning does not involve the entire knowledge base. Instead, as shown in Figure 3, the simulation reasons over only that part of the knowledge attached to the scripts under consideration. This automatic segmentation of the knowledge occurs in such a way that the modeler often fails to notice.

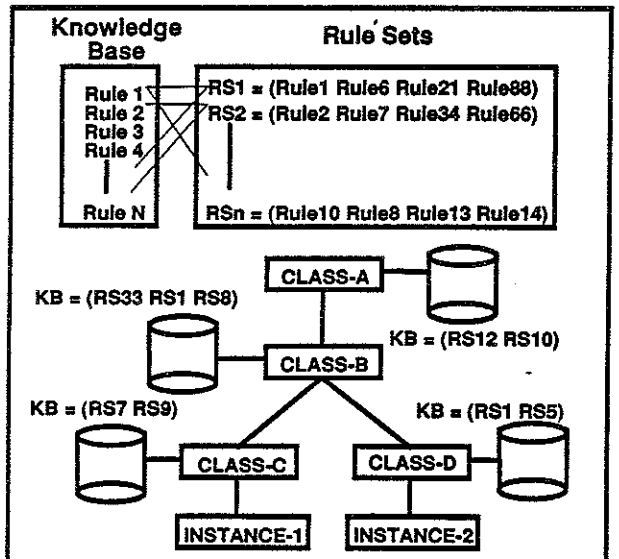


Figure 3.
Hierarchical Segmented Knowledge Bases

One challenge in rule development is to "step back" from the process being modeled to produce a rule applicable to several procedures. This is especially true of resource request rules. Several types of payloads, for example, require a certain type of technician to perform certain tasks, such as alignment. Producing a generic rule to acquire this technician means one rule is mapped into all rule sets callin for this operation. This saves considerable memory, since the rule exists in one location and mapped to the rule sets requiring it.

As an example of the representation of the knowledge gathered for the models at KSC, a rule is needed to acquire a crane in the SSPF to relocate a payload as it moves through its processing activities. This rule looks like:

```
(define-hskb-rule P1
   :lhs (ACQUIRE-CRANE ?PAYLOAD ?CRANE)
   :rhs ((BIND-IN-LIST ?CRANE (ASK 'GSE-
                CRANES CHILDREN))
         (> (ASK '?CRANE REQUEST 'LIFT-
                CAPACITY 'IS)
            (ASK '?PAYLOAD REQUEST 'MASS
                'IS))
         (RETURN-VALUE ?CRANE))
```

This rule finds a crane whose lift capacity is simply greater than the mass of the payload. With the backward chaining process used in reasoning, the left-hand side (:lhs) or *consequent* of the rule is true if the right-hand side (:rhs) or *antecedent* is satisfied. All the statements in the rhs of the rule must be satisfied for that side to be true. In the case of this rule, a crane object (variable ?crane) is capable of lifting a payload if the rhs is complete. The BIND-IN-LIST command assigns the ?crane to each of the children (first-order descendents) of the class GSE-CRANES. Each of the cranes is queried to determine the lift capacity of the device *(ASK '?CRANE REQUEST 'LIFT-CAPACITY 'IS)*. This is compared to the mass of the payload *(ASK '?PAYLOAD REQUEST 'MASS 'IS)* to assure a simply greater-than relation. The first crane to satisfy these requirements is returned to the caller *(RETURN-VALUE ?CRANE)*.

This rule could be expanded to include function calls or calls to other rules, to assure the accuracy of the model. By replacing the greater-than *(>)* line above with:

```
(BIND ?X (ASK '?CRANE REQUEST 'LIFT-
          CAPACITY 'IS)
(CRANE-CAPACITY (ASK '?PAYLOAD REQUEST
          'MASS 'IS) '?X)
```

This section would call a function or a rule *CRANE-CAPACITY* to calculate the proper amount of over-capacity for a crane to safely lift a payload. For example, if KSC requires the lifting device to exceed the payload's mass by 40%, or use a more complicated formula, or tabular values, the rule could accommodate the requirements.

Expanding the scope of the above rule to choose among several different devices requires only simple expansion of the above concept. There is no 'or' predicate supported by the HSKB rule syntax at this time; a slightly different approach is required to execute an 'or' condition in rule firing. Several rules in a single rule set with identical consequents replace a traditional 'or' structure. Choosing among several payload moving devices, such as cranes, fork lifts, and air-bearing pallets, would use 3 rules with the same *:lhs* attribute. The order of the rules in the rule set governs the firing order. The rules set might look like:

```
(DEFINE-HSKB-RULE 'MOVE-PAYLOAD
      :lhs (MOVE-PAYLOAD ?PAYLOAD)
      :rhs (ACQUIRE-MOVING-DEVICE
            ?PAYLOAD)

(DEFINE-HSKB-RULE 'ACQUIRE-CRANE
      :lhs (ACQUIRE-MOVING-DEVICE
            ?PAYLOAD)
      :rhs (...)

(DEFINE-HSKB-RULE 'ACQUIRE-PALLET
      :lhs (ACQUIRE-MOVING-DEVICE
            ?PAYLOAD)
      :rhs (...)

(DEFINE-HSKB-RULE 'ACQUIRE-FORK-LIFT
      :lhs (ACQUIRE-MOVING-DEVICE
            ?PAYLOAD)
      :rhs (...)
```

The first rule in the set calls for a rule with the pattern *:lhs (ACQUIRE-MOVING-DEVICE ?PAYLOAD)*. The three rules matching that pattern are tried in order, the final device coming from the first rule to match its antecedent. In this rule set, a crane is more desirable than a pallet, with a fork-lift being the least desired.

These rules can be utilized at many points in the simulation. Any time in a script that a certain rule set is needed, an HSKB message can be attached to activate rule usage. To get a crane to relocate a payload within the SSPF, for instance, a message could be used. The text of the message would appear as:

```
(DEFINE-HSKB-MESSAGE 'PAYLOAD 'MESSAGE26
      ()
      (ACQUIRE-CRANE ?SELF ?CRANE))
```

The rule *ACQUIRE-CRANE* is passed the values SELF (the payload calling the message) and CRANE, which will contain the crane returned by the rules. This will allow the script to acquire a crane to perform its current activity.

V CONCLUSION AND FUTURE RESEARCH

As the models under development at KSC become more robust and flexible, many varied processes could be simulated. The current processing of STS payloads could be studied, along with the processing of the shuttles themselves. KSC's budgetary cycle could also be modeled. Impacts of future NASA programs, such as lunar base concepts and planetary missions, could be assessed to determine long-range goals.

With the user-oriented capabilities of the AGAPE system, coupled with its robustness, focus the process of knowledge engineering into a single, continuous technique. Allowing the models to be developed *directly* by the people performing the payload processing brings the actual knowledge one step closer to the simulation. This system could assist many studies at KSC, not to mention processing scenarios at other NASA centers.

REFERENCES

Castillo, D., Ihrie, D., McDaniel, M., Tilley, R. 1987. "An AI Approach for Scheduling Space-Station Payloads at Kennedy Space Center." In *Proceedings for Third Anual Conference on Artificial Intelligence for Space Applications* (Huntsville, AL, Nov, 1987), pp. 361-370.

Castillo, D., McDaniel, M., Sieck, B., Tilley, R. 1988. "Coupling Artificial Intelligence and Simulation for Analyzing Payload Ground Operations at Kennedy Space Center." In *Proceedings for First Florida Artificial Intelligence Research Symposium* (Orlando, FL, May 4-6, 1988).

Castillo, D., McRoberts, M., Sieck, B. 1988. "Embedded Expert Systems Improve Model Intelligence in Simulation Experiments." In *Proceedings of Summer Simulation Conference* (Seattle, WA, July 24-26).

Castillo, D., McRoberts, M., Sieck, B. 1989. "HSKB: An Architecture for Embedded Reasoning in Simulation Systems." In *Proceedings of Multi-Simulation Conference* (San Diego, CA, Jan. 1989).

AUTOMATED KNOWLEDGE GENERATION FROM INCOMPLETE CAD DATA: RESEARCH RESULTS

Harley R. Myler Avelino J. Gonzalez
Massood Towhidnejad Frederic D. McKenzie Robin Rouch Kladke

University of Central Florida
Dept. of Computer Engineering
Orlando, FL 32816

ABSTRACT

The task of knowledge acquisition is often prohibitive in terms of time and expense, hindering the development of knowledge-based applications. The ongoing Automated Knowledge Generation (AKG) research at the University of Central Florida is addressing this problem by attempting to create a complete knowledge base directly and automatically from computer-resident data sources.

The primary focus of the first year of AKG research [Gonzalez et al. 88, 89] has been the development of a prototype system capable of accessing Intergraph CAD (ICAD) database files and producing a complete frame-based knowledge base compatible with NASA's Knowledge-based Autonomous Test Engineer (KATE) diagnostic system. A secondary focus has been to maintain generality within the system which would allow for easy adaptation to other data sources and diagnostic environments.

1.0 INTRODUCTION

The AKG system extracts information from process control systems in much the same way that engineers do. When an engineer views a CAD representation of a process, he/she immediately forms an abstract mental representation of the physical system. This initial impression is heavily dependent on perceived interconnectivity of components and a priori categorization of component functionality. The accuracy of the engineer's interpretation depends on his/her knowledge, experience, and ability to reason through inconsistencies [Gonzalez et al. 88]

The Prototype AKG system is divided into seven modules:

1. ACCESS: remote data communication module
2. SPAWN: component flavor generator
3. CONSTRAINT GENERATOR: component connectivity determinant
4. BUILDER: knowledge base frame builder
5. RESOLVER: incomplete/inconsistent data resolver
6. COMPONENT AND PROCESS DATABASE: internal data source
7. USER INTERFACE: graphic window interface to AKG

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

1.1 Object-Oriented Approach

An Object-Oriented Programming (OOP) approach under the Symbolics 3650 LISP machine environment has been taken in the development of AKG. Each component from the CAD database is represented as a flavor object within AKG. Such an approach is felt to accurately model the physical system by representing components as an organized set of discrete objects capable of communication with external processes. In addition, OOP encourage modularity of design, making development, modification and enhancement of the system much simpler.

1.2 The AKG Process

AKG performs the task of knowledge extraction by first educating itself on the system's composition. A CAD-generated description of the target system is obtained through the ACCESS module. This module communicates with the computer hosting the CAD system and downloads two CAD database files, COMPOC.DAT and TOFROMC.DAT. In order to accomplish this, ACCESS uses a customized file which contains the unique communication configurations required by the host as well as appropriate database query instructions needed to format the two files. The COMPOC.DAT file contains component details made up of a unique identifier, nomenclature, and possibly other descriptive information such as operating range and units. The TOFROMC.DAT file contains structure data which describes the component interconnectivity of the system being modelled. The SPAWN module then uses information from the COMPOC.DAT file to create unique component objects within the AKG environment and the CONSTRAINT GENERATOR module sends connectivity information to each of these component objects. The connectivity

structure imposed represents an initial constraint set on the system.
Once the CONSTRAINT GENERATOR establishes the initial component connectivities, AKG accesses the COMPONENT AND PROCESS KNOWLEDGE BASE to capture additional information. The COMPONENT AND PROCESS KNOWLEDGE BASE is currently in the conceptual stage. Current work is focused on structure and storage issues. An effort is being made to design a structure which is optimal in terms of speed and convenience while remaining adaptable to future implementation in a parallel environment. The anticipated structure includes various levels of knowledge and abstraction with like objects occupying the same general area within a level. Process knowledge relating to a particular type of object will be stored in the structure of that object, while knowledge that is more general will be stored in a separate, related database in a similar, less complex structure. Given the large expected size of the knowledge base, such a structure which minimizes search space is desirable.

The objects in the component database will be stored as text in a Lisp structure. When accessed, specific objects, which are assumed related to the type of process being represented, will be copied to a current-process database to facilitate subsequent access. Since process control systems tend to use similar components throughout, the caching of component objects into the process-related database will help to optimize additional searches. Within the process-related database, objects can be changed, updated, and even spawned into the AKG object cluster. This proposed approach to the COMPONENT AND PROCESS KNOWLEDGE BASE will provide the type of accessibility and interrelatedness needed for rapid component resolution.

Based on the information gathered from the COMPONENT AND PROCESS KNOWLEDGE BASE, a confidence factor is generated for each object in the system. This value represents AKG's level of confidence that a component has been labelled correctly with respect to its function. A global threshold is established by the user which, when exceeded, flags a component as ready for conversion into a knowledge-base frame. If a component cannot be resolved (i.e., its confidence factor does not reach the minimum threshold), due to lack of information or conflict, the RESOLVER module is called to deduce the correct identification from component data stored in AKG's internal knowledge base.

The development of the RESOLVER has moved beyond the conceptual stage into design. The RESOLVER is the primary module which handles confidence propagation. There are two confidence factors related to a component object: path consistency and component consistency. The path consistency is a measure of the possibility that the connections between a component and its upstream and downstream components are valid. Path consistency acts as a weighing factor for a component. This weighing factor affects the amount of influence that this component's confidence factor may have on the confidence factors of its upstream and downstream components. The component consistency represents a measure of that certainty that a component has been completely identified. This factor includes not only functional identification of the component, but also the influence of path consistency.

Once the RESOLVER is called, all the components in the system are examined and the components with the highest confidence factors are marked. Based on the information in these marked components (i.e., their path consistency and component consistency values), the propagation of confidence proceeds beginning with neighboring components. The propagation of confidence factors is global in the system and continues until all the components' confidence factors change less than some preassigned rate of convergence. At that time, the system's confidence factor is considered settled. The RESOLVER then scans all the components in the system and flags the components with confidence factors below the user-defined threshold. As a last resort, the RESOLVER asks the user to supply new information and confidence factors for these flagged components. This resolution process repeats until all the components' confidence factors exceed the threshold value.

Confidence levels are updated through a technique developed for image understanding called **relaxation labeling.** This process is a subset of constraint labeling, in which a system of elements with some arbitrary connectivity must be labeled according to a predetermined criterion. The criterion is selected so that a labeled component with a high degree of confidence can influence a nearby component with lesser confidence. The process of iterating across the network of component nodes and reiterating within clusters of nodes is referred to as relaxation labeling.

AKG uses the criterion that process and control system components must have similar, and sometimes identical, properties. For example, one never connects a NAND gate to a pressure valve. Likewise, a valve rated at 200 PSI would not (properly) be connected to a pressure pump capable of 800 PSI. The COMPONENT AND PROCESS KNOWLEDGE BASE of AKG provides these properties as supplement to the CAD data. This knowledge base contains general domain knowledge concerning component details and system aspects of process control. Such information will not only include standard values for tolerance, delay and transfer function (status) for each generic component represented, but also will include information indicating which components may be validly connected. Each of these auxiliary data sources is capable of growth; as the AKG system acquires more information about the

components and control mechanisms, it becomes more capable of resolving conflict without resorting to human interface.

AKG compares, using the relaxation algorithm and the COMPONENT AND PROCESS KNOWLEDGE BASE, the validity of the system component connections. When a component is flagged as valid, AKG is then able to assign a function to it that is consistent with the target reasoning system. Such an intelligent approach to conflict resolution raises the AKG system well above the capability level of a simple translator.

2.0 TRANSLATION VERSUS INTELLIGENT INTERPRETATION

The AKG prototype took as its testbed a KATE demonstration circuit for purging pneumatic systems called the Purge Demo. The KATE knowledge base for this system had been manually constructed by NASA personnel. A test of the AKG prototype was to autonomously produce a knowledge base which would closely approximate its human-generated counterpart. The primary goal of the first year of research was to replicate the results obtained by Thomas [87] in his knowledge generation research on the Purge Demo carried out in the Summer of 1987. Thomas's work indicated that simple translation is not sufficient for the resolution of CAD data into a knowledge base.

The results of both studies are represented in Table 1. A description of the KATE slots depicted in the table may be found elsewhere [Cornell 87, Gonzalez et al. 88]. As shown in Table 1, Thomas's translation approach was found to be unable to provide any values for some KATE slots and predicted a relatively low potential capability to fill others. In each of these cases the AKG intelligent interpretation approach is found to be superior. The component information of the COMPONENT AND PROCESS KNOWLEDGE BASE coupled with intelligent parsing will enable slots AN-ELEMENT-OF (AEO), TOLERANCE, DELAY, and STATUS (transfer function) to be filled at least 75% of time. The process information coupled with the TO-FROM list will allow identification of 90% to 100% of the SOURCE-PATH, IN-PATH-OF, SOURCE, and SINK slots. The resolution process, unavailable to a translator, will depend heavily on relaxation labeling.

The AKG system provides many advantages over a simpler translation approach. A knowledge base translator is capable only of uncritically reformatting information explicit within its input data. An intelligent interpreter, however, is able to extend and correct input by inferring missing values and resolving conflicts. This ability is vital if a conversion system such as AKG is to rise above the level of a user-friendly querying device.

TABLE 1. COMPARISON OF RESULTS.

	Translation Results			AKG Results		
Slots	Filled Auto.	Percent	Est. of Pot. Cap.	Filled Auto.	Percent	Est. of Pot. Cap.
aio	52/52	100%	100%	26/26*	100%	100%
aeo	NA	NA	NA	14/14*	100%	> 75%
nomencl.	0/52	0%	100%	13/13*	100%	100%
source-path	8/52	15%	50%	25/25#	100%	> 90%
in-path	52/52	100%	> 90%	35/35	100%	> 90%
source	2/2	100%	> 80%	NA	NA	100%
tolerance	NA	NA	NA	2/2*	100%	75%
delay	0/3	0%	0%	3/3*	100%	75%
status	0/52	0%	50%	6/6*	100%	75%
units	23/23	100%	100%	23/23	100%	100%
range	16/16	100%	100%	15/15	100%	100%
sinks	2/2	100%	80%	NA	NA	100%

NOTES:

*: Filled with the help of the component database

#: Need special operators to get this result

2.1 The Environmental Control System Testbed

The concrete differences between translation and intelligent interpretation were accentuated when the AKG system moved beyond a demonstration circuit and was given a real physical system as input. The current AKG input is the Environmental Control System (ECS) of the Orbiter Maintenance and Refurbishment Facility (OMRF). Unlike the Purge Demo, the ECS CAD representation is not idealized. It contains a number of inconsistencies and CAD misrepresentations which a translator would be unable to handle. The power of AKG lies in its ability to resolve problems of this nature.

2.2 The Problems Encountered with CAD data

It has always been assumed that the CAD structure information is correct and the conversion process is limited only by the availability of functional information. This, however, is not the case. The CAD system structure is dependent on the techniques used by the CAD draftsperson to connect components. Often the database interpretation does not reflect the information in the graphic output.

The primary source of CAD misinterpretation problems encountered by the AKG researchers is the manner in which logical and physical connections are mixed and placed. Since the conventions followed in carrying out these tasks are wholly dependent on the

draftsperson, the CAD interpretation of a system containing these elements is unpredictable.

One example of this is shown in Figure 1. A single output feeding multiple inputs, though appearing correct graphically, may not be interpreted correctly in the database.

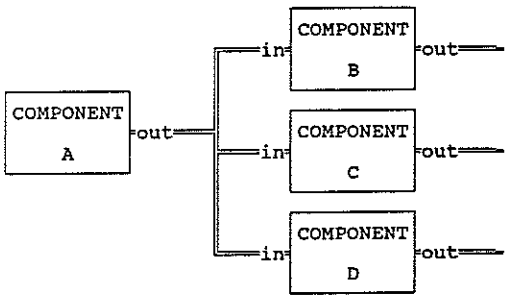


Figure 1. Correct interpretation of single output feeding multiple inputs.

Instead of correctly reflecting this information functionally in the database, cases have been found in which the to-from connections are misinterpreted as shown graphically in Figure 2. Instead of representing A as the only input to B, C, and D, components A, C, and D are all interpreted as feeding input to B. and components C and D are incorrectly interpreted as having no inputs and multiple outputs. This problem can be directly traced to the manner in which physical and logical connections are placed in the CAD drawing.

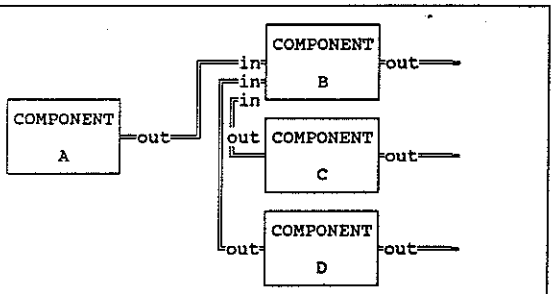


Figure 2. Incorrect (actual) interpretation of single output feeding multiple inputs.

This inconsistency in indicating flow leads to a second, related problem. The AKG system traverses the to-from list to identify starting and ending components within the system. Given the interpretation of Figure 2, components A, C, and D are all considered starting points of flow since none have upstream components attached.

A translator would be unable to recognize that the problems outlined above even exist. The CAD data would simply be accepted as valid and the error in interpretation would be propagated throughout the knowledge base, producing nonsense. AKG, on the other hand, should be able to handle these conflicts through the RESOLVER module. With the aid of data from the COMPONENT AND PROCESS KNOWLEDGE BASE, nonsensical connections will be recognized by the system and flagged with low confidence.

A further problem may arise when a measurement device taps into a connection line between components (see Figure 3). In such a situation, the measurement device should be interpreted as an output node or endpoint.

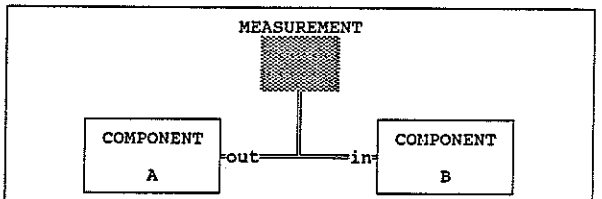


Figure 3. Correct interpretation of a measurement tapping into a connection.

Because of the CAD system's inability to differentiate measurement devices from normal components, the interpretation shown in Figure 4 may result. In such an case, component B is interpreted as having two inputs: component A and Measurement. Only one of these inputs is valid, however. AKG must resolve this conflict by determining the true input and categorizing other inputs as commands or measurements.

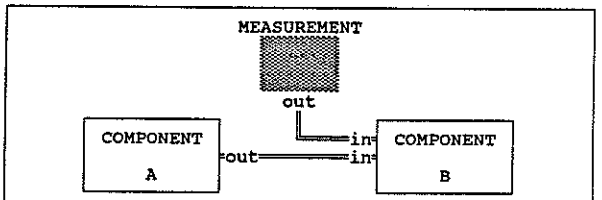


Figure 4. Incorrect interpretation of a measurement tapping into a connection.

The problem of functionally identifying components would also be overlooked by a translator, however, several approaches are currently being examined by AKG to resolve it. One involves the intelligent parsing of available information for clues as to the component's identification. A second entails obtaining additional clues by examining the functionality information of neighboring components. Such approaches would be adequate given that the information utilized is

accurate and available. A more comprehensive solution to the problem would involve icon recognition of unresolved components. All three approaches in conjunction with the relaxation labeling previously mentioned should be capable of total resolution.

The problem of misinterpreted flow is compounded when the draftsperson places connections in the incorrect order. If the connection between components A and B above was made by selecting B as the starting point and selecting A as the ending point, component A would be seen as downstream of B, regardless of other connections already made. If B was meant to be an ending component of the system, it would not be recognized as such. Component A would be incorrectly interpreted as the end component.

This problem is beyond the ability of a translator to solve. A translator would assume that a connection drawn from one component to another represents correct flow. If, however, the connection were drawn in the incorrect order, an unresolvable conflict would arise later in the analysis. An intelligent interpreter would be able to overcome this conflict by overriding conflicting facts within the system by using information supplied by the COMPONENT AND PROCESS KNOWLEDGE BASE.

Because of these problems, and those not yet encountered, it will be necessary for the AKG system to pre-parse the connectivity information in order to validate the structure coming from CAD. This resolution process will be incorporated into the CONSTRAINT GENERATOR module.

3.0 CURRENT AND FUTURE DIRECTIONS IN AKG

In addition to the work being carried out on the COMPONENT AND PROCESS KNOWLEDGE BASE and RESOLVER discussed earlier, the AKG is in transition to a new KATE frame representation and is examining approaches to the component identification problem.

3.1 The New KATE Frame Structure

Since the development of the prototype system and its testing on the Purge Demo, KATE has undergone some changes in frame structure. These changes have largely been made to incorporate object-oriented programming into the structure of KATE.
The new frames currently fall into two major groups: generic (upper-,mid-level) and instance (low-level) frames. Generic frames contain information common to a group of components. Instance frames contain information regarding specific components.

Because of the modular and generic nature of AKG, transition to the new structure required very little effort. The following is a list of the frame slots currently supported by KATE and AKG:

Slots found in generic frames:
1. KINDS: higher order classification of objects from which qualities are inherited.
2. AKO (A Kind Of): inheritance slot linking object with classification.
3. INSTANCES: functional relationship between inputs and outputs.
4. CURRENT-VALUE: the present value of a measurable object.
5. UNITS - the operating units of an object's value.

Slots found in generic frames which may be overridden by instance frames:
6. PARAMETERS: scaling factor.
7. DELAY: maximum allowable time, in intervals of 1/60 of a second, between an input change and an output effect.
8. TOLERANCE: maximum allowable deviation from the expected value.
9. RANGE: the anticipated operating range of an object.

Slots common to both generic and instance Frames:
10. NOMENCLATURE: the printed description of an object for use in reports and screen displays.
11. INPUTS: downstream objects which have the current object as input.
12. OUTPUTS: upstream objects which serve as input to the current object.

Instance frames:
13. AIO (An Individual Of): inheritance slot linking an instance with its higher-order classification.

The following is an example of a set of frames created with the new slots:

```
;;; generic level frames
(DEFRAME ANALOG OBJECT
  (NOMENCLATURE "an object with analog
       input")
  (AKO THING)
  (KINDS MULTIPLIER))

(DEFRAME MULTIPLIER
  (NOMENCLATURE "a multiplier")
  (AKO ANALOG-OBJECT)
  (INSTANCES MULT1)
  (INPUTS (IN1))
  (OUTPUTS (OUT1) (OUT2))
  (OUTPUT-FUNCTIONS (OUT1 (* IN1 SCALAR1))
       (OUT2 (* IN1 SCALAR2)))
  (PARAMETERS (SCALAR1 10.0) (SCALAR2 5.0))
  (DELAY (OUT1 60) (OUT2 60))
  (TOLERANCE (OUT1 .1) (OUT2 .2))
  (UNITS volts))

;;; instance level frames
(DEFRAME MULT1
  (NOMENCLATURE "a multiplier")
  (AIO MULTIPLIER)
  (INPUTS (IN1 (C1 OUT1)))
  (OUTPUTS (OUT1 (M1 IN1)) (OUT2 (M2 IN1)))
  (PARAMETERS (SCALAR1 15.0)))
```

3.2 The Intelligent Parser

The first module to be implemented for aiding in the identification of components is that of an intelligent parser. The parser will glean significant information from the nomenclature of a component which will aid in determining a key into the COMPONENT AND PROCESS KNOWLEDGE BASE. For example, there are many potential types of valves in a process control system (gate, ball, butterfly, etc.). In order to key into the specific information contained in the COMPONENT AND PROCESS KNOWLEDGE BASE, AKG must accurately recognize the component being examined. The parser will analyze the descriptive information provided through the CAD database in order to facilitate this recognition process.

3.3 An Icon Recognizer

If descriptive information is not available for a component or if the information is too general (e.g., a butterfly valve is described only as a valve), another method must be available to resolve the component's identity. An obvious source for this information is the user, but in order to maximize the autonomy of AKG, another option is sought. The approach currently being considered is icon recognition. A recognizer would be capable of examining the CAD graphic directly using image processing techniques and, given that components within the CAD drawing conform to a prescribed standard, determine the identity of a component for which little or no textual information is known.

4.0 CONCLUSIONS

This paper has discussed the basic differences between what is referred to as a translator and an intelligent interpreter, such as AKG. The results of generating knowledge from a real process control system was found to not only confirm the viability of the AKG system, but also point out situations where intelligent interpretation must compensate for inconsistencies in CAD database representations.

The current work of the AKG researchers is moving the project toward the realization of a nearly autonomous knowledge generator. By accessing knowledge which is supplemental to the CAD database, AKG is provided with the majority of the basic knowledge necessary to proceed with the resolution process. The upcoming year of research will focus attention on this problem of conflict resolution.

ACKNOWLEDGEMENTS

The AKG research project is supported by NASA, Kennedy Space under contract NAG-10-0043

REFERENCES

[Cornell 87] M. Cornell, "The KATE Shell - An implementation of Model-Based Control, Monitor and Diagnosis", Proceedings of the First Workshop on Space Operations Automation and Robotics, Houston, Texas, 1987.

[Gonzalez 88] A. J. Gonzalez, H. R. Myler, B. C. Owen, and M. Towhidnejad, "Automated Generation of Knowledge from CAD Design Data Bases", Proceedings of the First Florida Artificial Intelligence Research Symposium, Orlando, Florida, 1988.

[Gonzalez 89] A. J. Gonzalez, H. R. Myler, and M. Towhidnejad, "Automated Knowledge Generation in Support of Shuttle Ground Operations", Proceedings from the Artificial Intelligence in Government Conference, Washington, D.C., 1989.

[Thomas 87] S. J. Thomas, "Automated Construction of a Knowledge Base from Computer Aided Design Data", NASA Internal technical report, Artificial Intelligence Section, 1987.

AN APPROACH TO THE AUTOMATED ELICITATION OF SOFTWARE REQUIREMENTS

Patrick O. Bobbie / Kenneth M. Ford / Edward G. Rodgers

Division of Computer Science
University of West Florida
Pensacola, Florida 32514

ABSTRACT

This paper is on a research effort aimed at the application of knowledge acquisition techniques to the problem of eliciting an adequate set of software requirements. Software design and construction is usually carried out in strict conformance with a development life cycle. Although this life cycle comes in a variety of models, several models begin with the requirements specification phase. It is at this crucial phase that the user attempts to communicate to the software engineers what the system must do when installed. Thus, the quality and ease of production of a software requirements specification document is generally a function of the ability of the software engineer and user to communicate. Unfortunately, most software engineers have little or no experience in formal methods for eliciting descriptive knowledge from users. The authors feel that the difficulty of eliciting a useful set of system requirements from a software user is fundamentally the same problem as that of eliciting expertise from a domain expert for the construction of an expert system. In this paper, an effort aimed at developing and unifying the necessary theoretical and practical models for the semi-automated elicitation of software requirements from repertory grid data is discussed.

INTRODUCTION

The repertory grid model (Kelly, 1955) serves as a basis for eliciting knowledge from a software user. Further analysis of elicited knowledge results in a requirements document. The repertory grid model has three major components -- the *constructs*, *elements*, and preference *ratings*. The constructs form a cognitive dimension with each construct representing a binary perception of a given element. Thus, a cognitive dimension is bipolar and indicates the two extremes (left and right-hand poles) of some perception. Constructs are contextually dependent and the semantics thereof could be variously defined or rated. The elements are drawn from a common domain or class and are characterized by common events. The elements are either real or abstract, but representative of a given system or context. The ratings are subjective quantities chosen by the user from the ranges of the 'bi-polar' constructs to characterize some preferential perception of a given element.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

SYSTEM MODELING

We model the requirements phase of the software development cycle along the following dimensions (Roman, 1985).

- The functional problem domain (FPD).
- The non-functional problem domain (NFPD).
- The functional problem environment (FPE).
- The non-functional problem environment (NFPE).
- The causal domain events from the environment (E).

The FPD dimension constitutes the domains from which *system* elements are drawn. The FPE dimension constitutes the elements of the "world" that directly affect the FPD. The NFPD dimension characterizes the constructs or the qualitative attributes of FPD. The NFPE dimension represents the constructs of the "world" or environment in which the system is to operate. In addition, the E dimension intersects with the other four dimensions in that it represents the temporal events that cause changes in the system's behavior.

INTERACTIVE KNOWLEDGE ELICITATION

The paradigm which unifies the repertory grid and the software requirements models is the fourth-generation paradigm, *spreadsheet*. The spreadsheet paradigm is an efficient and convenient grid elicitation technique. The spreadsheet visual interface enhances the user's ability to recognize and offer distinctions between the elements on the basis of the constructs. Furthermore, spreadsheets offer a flexible user-interface, and aid users in visualizing several of the repertory grid relations at once.

Software users (i.e., clients) adapt, with relative ease, to the use of spreadsheets, thereby reducing complete reliance on the software engineer. With some assistance from the software engineer, a user initially enters the values for the constructs and elements directly into the repertory grid. The user then fills in the ratings of the elements for each construct. At any time during the knowledge elicitation process, the user may choose to appraise the requirements produced so far, and if necessary, readjust the ratings in the grid. The ratings are multi-valued, ranging from the left-hand pole to the right-hand pole of the constructs spectra.

CONSTRUCTION & ANALYSIS OF REPERTORY GRIDS

Construction

In binary rating systems, the assignment of a '1' means that the rating of a given construct is 'true' for the element. Conversely, the assignment of a '0' indicates the 'false' value for the element. Consequently, such binary grids are viewed as matrices of Boolean values. As a result, the entailments or specification rules derived from the repertory grid data are based on classical two-valued logic (Adams-Webber, 1979).

However, in the case of multi-valued grids (e.g., Figure 1) each repertory grid may also be considered as a set of element-measurement vectors. Thus, we define a mapping function f which models an element-measurement vector as follows:

$$f[C, M_V(X) = \alpha] = M_V(X), \text{ for all } C$$

where $M_V(X) = \alpha$, is a measure of an element X on the interval of 'bipolared' construct C. Each measure of X is relative to other elements' measures and lies between the left- and right-poles of construct C. An element X may have different measures across all constructs. For example, an element X_1 may have an element-measurement vector

$$M_V(X_1) = \{(X_1, 1.0), (X_1, 0.25), ..., (X_1, 0.5)\},$$

where each entry represents different construct ratings. Similarly, we define a mapping function g which models a construct-measurement vector as follows.

$$g[X, M_H(C) = \alpha] = M_H(C), \text{ for all } X$$

where $M_H(C) = \alpha$, is a rating on the C's interval for an element X. The construct C may have different values across all the elements. For example, a construct C_1 may have a construct-measurement vector:

$$M_H(C_1) = \{(X_1, 1.0), (X_2, 0.75), ..., (X_N, 0.0)\},$$

where each entry represents construct C_1 ratings for each of the N elements. The range of rating values determines the numerical values available for representing the position of an element on a construct's interval. For example, the values (0.0, 0.25, 0.5, 0.75, 1.0) result in five numerical values. The repertory grid data is, therefore, either a set of element-measurement column vectors or construct-measurement row vectors. Also, the grid may be viewed as a two-dimensional projection, referred to as an α-plane matrix (Ford & Chang, 1989). In the above example with five numerical values, the grid is referred to as a 5-plane matrix.

The resulting α-plane matrix is multi-valued and the entries indicate the ratings or α-values. The α-plane can be mapped into a set of α-plane binary (or two-valued) matrices according to the following definition:

$$\alpha\text{-plane}(C_i, X_j) = \begin{cases} 1 \text{ if } f(C_i, m_v(X_j)) = \alpha \\ 0 \text{ if } f(C_i, m_v(X_j)) \neq \alpha \end{cases}$$

(Note: α-plane multi-valued matrices can also be mapped using the function g).

Analysis

Given some heuristic rules governing permissible combinations of α-planes, the set of all possible generalizations derivable from the repertory grid data is called the hypothetical set G (Ford & Chang, 1989). A generalization taken from the set G may then be evaluated by some rule(s) of confirmation (Ford & Chang, 1989) to derive the degree of support provided by the evidence to the generalization.

A CASE STUDY

The Problem

The case study is a pricing system for resource allocation in a public enterprise economy (Eckert & Leftwich, 1988). The task is to develop a collection of programs for automatic shifting and reallocation of resources from one area to another to yield an optimal value of marginal product (VMP). The most crucial aspect of the task is the ability to capture the user requirements in a completely descriptive manner without loss of user goals. For the pricing system, the goal is the capability for regenerating specifications based on the dynamics of market values. Hence, the specifications document emphasizes the non-procedural aspects that determine the VMP.

The Model

We map the requirements structure of the pricing system onto the five dimensions of the requirements phase of the development cycle. First, the user describes the major components of the pricing system. Secondly, each component is further described to delineate the specific elements of primary interest. A typical pricing system comprising the identified components and a set of constructs which affect the functional behavior of the elements is presented below.

FPD Elements	FPE Elements
1. Office Personnel	1. Education
2. College Graduates	2. Construction Firms
3. Lawyers	3. Machine Shops
4. Contractors	4. Government
5. Engineers	5. Retail Stores
6. Physicians	6. Agriculture
7. Teachers	7. Manufacturing
8. Scientists	8. Hospitals
9. Farmers	9. Financial Institutions
10. Managers	10. Economic Institutions
11. Economists	11. Legal Institutions

The following is a set of common constructs (NFPD and NFPE) that prescribe the qualitative attributes of the elements. The constructs motivate changes in an economy and directly impact on the dynamics of the pricing system. Each construct is defined with its left- and right-hand poles.

1. Mobility (vertical..horizontal)
2. Quantity (high..low)
3. Transferability (desired..not desired)
4. Efficiency (high..low)
5. Availability (high..low)
6. Timeliness (long..short)
7. Productivity (high..low)
8. Pricing (fixed..variable)
9. Cost (high..low)
10. Marginal Value (high..low)

Events E that precipitate the adjustment of the economic parameters are (1) periodic productivity assessment and (2) beginning of year budget assessment. Each event occurrence results in regeneration of specification rules to reflect normalized market values.

Mapping The Model Into Repertory Grids

We generate a repertory grid matrix to model the multi-valued ratings. The user fills in the grid based on a confidence constant interval (α-upper .. α-lower), where α-upper and α-lower are the left- and right-hand poles of the constructs respectively. Because of space limitation, we present in Figure 1 the grid data for the FPE elements and the constructs. The goal is to precisely specify the economic parameters that precipitate reallocation of human resources in the public economy.

Element Index:

1 - Education	5 - Retail Stores	9 - Financial Institutions
2 - Construction Firms	6 - Agriculture	10 - Economic Institutions
3 - Machine Shops	7 - Manufacturing	11 - Legal Institutions
4 - Government	8 - Hospitals	

Rate elements on a scale from 1 -> 3, a '1' is at the LHP, a '3' is at the RHP, while the rest fall between the poles.			Elements										
			1	2	3	4	5	6	7	8	9	10	11
C 1	vert. mobility	horizon. mobility	2	1	3	2	2	3	2	1	3	1	1
C 2	high quantity	low quantity	3	2	2	3	1	1	3	2	2	1	1
C 3	desired trans.	no desired trans.	2	2	3	2	1	1	2	3	2	1	2
C 4	high eff.	low efficiency	1	2	2	3	1	2	3	2	2	1	2
C 5	high avail.	low availability	2	2	3	2	1	2	1	2	1	2	3
C 6	long time.	short timeliness	2	3	2	3	2	1	1	2	1	3	2
C 7	high product.	low productivity	2	3	2	1	1	3	2	1	2	1	2
C 8	fixed pricing	variable pricing	1	2	2	3	3	2	3	1	3	1	2
C 9	high cost	low cost	2	2	3	2	1	2	3	2	1	2	3
C10	hi margin. val.	low margin. value	2	2	1	2	1	2	3	2	1	2	3

Figure 1: A sample repertory grid of economic parameters.

Figure 2 represents the collection of α-planes derived from the repertory grid in Figure 1.

	1.0	.50	0.0
	1 2 3 4 5 6 7 8 9 10 11	1 2 3 4 5 6 7 8 9 10 11	1 2 3 4 5 6 7 8 9 10 11
1 Mobility	0 0 1 0 0 1 0 0 1 0 0	1 0 0 1 1 0 1 0 0 0 0	0 1 0 0 0 0 0 1 0 1 1
2 Quantity	1 0 0 1 0 0 1 0 0 0 0	0 1 1 0 0 0 0 1 1 0 0	0 0 0 0 1 1 0 0 0 1 1
3 Transfer.	0 0 1 0 0 0 0 1 0 0 0	1 1 0 1 0 0 1 0 1 0 1	0 0 0 0 1 1 0 0 0 1 0
4 Efficiency	0 0 0 1 0 0 1 0 0 0 0	0 1 1 0 0 1 0 1 1 0 1	1 0 0 0 1 0 0 0 0 1 0
5 Availability	0 0 1 0 0 0 0 0 0 0 1	1 1 0 1 0 1 0 1 0 1 0	0 0 0 0 1 0 1 0 1 0 0
6 Timeliness	0 1 0 1 0 0 0 0 0 1 0	1 0 1 0 1 0 0 1 0 0 1	0 0 0 0 0 0 0 0 1 0 0
7 Productivity	0 1 0 0 0 1 0 0 0 0 0	1 0 1 0 0 0 1 0 1 0 1	0 0 0 1 1 0 0 1 0 1 0
8 Pricing	0 0 0 1 1 0 1 0 1 0 0	0 1 1 0 0 1 0 0 0 0 1	1 0 0 0 0 0 0 1 0 1 0
9 Cost	0 0 1 0 0 0 1 0 0 0 1	1 1 0 1 0 1 0 1 0 1 0	0 0 0 0 1 0 0 0 1 0 0
10 Margin. Val	0 0 0 0 0 0 1 0 0 0 1	1 1 0 1 0 1 0 1 0 1 0	0 0 1 0 1 0 0 0 1 0 0

Figure 2: Alpha-planes derived from the repertory grid data in Figure 1.

Production of Requirement Rules From Repertory Grids

The system can quickly produce a set of "requirement" rules from the repertory grid data. Consider a generalization taken from the set of G such as,

"Hi-quality entails low-efficiency" *(.67).*

A numerical measure of the degree of confirmation of such an entailment relation may be had by dividing the confirmatory evidence by the total relevant evidence. In the example above, the entailment is confirmed by the evidence to a strength of .67. A detailed explanation and justification for the particular logic of confirmation is given elsewhere (Ford, 1989).

CONCLUSION

The major issue addressed in this paper is the use of the repertory grid model to provide a framework for eliciting non-procedural requirements data. The concept of the α-plane is discussed as a binary decomposition of repertory grid data. The decomposition process, coupled with the capability for realizing construct extension through combinations, permits the derivation of specification rules. Consequently, these rules represent the requirements specification against which further design and construction of the target software is checked for correctness. We demonstrated these theoretical frameworks through an example problem.

REFERENCES

Adams-Webber, J. R. 1979. *Personal Construct Theory: Concepts and Applications.* New York: Wiley.

Eckert, R. D. and R. H. Leftwich. 1988. *The Price System and Resource Allocation.* New York: Dryden Press.

Ford, K. M. and P. J. Chang. 1989. An approach to automated knowledge acquisition founded on personal construct theory. In: *Advances in Artificial Intelligence Research*, M. Fishman (Ed.). Greenwich, CT: JAI Press.

Kelly, G. A. 1955. *The psychology of personal constructs.* New York: Norton Publishers.

Roman, G-C. 1985. "A Taxonomy of Current Issues in Requirements Engineering." *IEEE Computer* 18, no. 4 (Apr.): 14-23.

A Parallel Implementation of Depth First Search

Roger E. Eggen
University of North Florida

abstract

This papers presents some of the ramifications and considerations that occur when parallel implementations of well known problem solving techniques in Artificial Intelligence are considered. Search is used to solve a wide variety of problems in AI. This paper considers some of the details and how the search process can change when implemented in parallel. A set of C macros are introduced as aids for controlling parallelism on loosely coupled computers.

INTRODUCTION

Several well known techniques for solving problems change when a parallel implementation is considered. This paper presents several considerations concerning the implementation of parallel depth first search that is not modified by any heuristics. Parallelism forces the programmer to reconsider some of the standard problem solving techniques used in AI. Since several processors are solving the problem concurrently, some of the typical pitfalls do not occur. An example occurs when the search process is not able to proceed to a leaf due to the depth of a branch of the tree. A process may still get stranded in a deep branch but other processes will continue to search the remaining portions of the tree. This paper will discuss the following considerations: the configuration of a loosely coupled parallel computer used in the implementation, the type of problems solved, communication between processors, side effects of the searching technique and the tools used to implement the parallel search. Parallelism provides problem solvers with not only improvements in performance but allows problems to be solved that previously required unrealistic computer times. For example, if the solution lies on the right side of a search tree, while the search process begins on the left side, an inordinate amount of computer time may be required to allow the search to proceed far enough to reach the right side of the tree. By assigning several processes to search the tree, a process can begin searching the right side immediately. Thus, search problems that contain very deep branches can now be solved in reasonable computer times. Note, however, that problems which are in class NP remain in class NP. Parallelism only reduces the amount of overall computer time necessary for problems that are solvable, parallelism cannot solve problems that are known to be unsolvable.

PROBLEM CONSIDERATION

Through parallelism a process is assigned to each branch of the tree. If sufficient processors are available, the depth first search technique becomes a combination of breadth first and depth first techniques since each branch of the tree is searched concurrently. Each

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

first is accomplished by using several processes to search the tree.

The organization used is to assign one process to solve the problem initially. When it is realized that the solution does not exist at this level (logically the root of the tree), subproblems are defined to pursue alternative or successive routes to the solution. Each subproblem is considered to be at the second level of the tree. The subproblem includes the partial solution found at the root and continues pursuit to the complete solution by following the alternatives that exist at the second level. When further choices exist the problem will again be subdivided, allowing more processes to be allocated to traverse new subtrees causing more parallelism to occur. This problem outline solves a variety of problems in AI since the amount of parallelism used is not defined initially, but rather is problem dependent. Therefore, problems with varying degrees of parallelism fall into this paradigm. For example, if we consider the search tree for the well known game of tic-tac-toe, the initial configuration has no x's or o's. This is the state of the current solution. The next move places an x in one of the 3 unique locations and each of these partial solutions is passed to a slave process that individually continues the search. Thus, the traditional depth first search becomes a depth first and breadth first search when parallelized since each branch is pursued concurrently.

PARALLEL COMPUTER CONSIDERATION

The loosely coupled computer is configured as a network of Sun workstations, however, the parallel paradigm can be implemented on either loosely coupled or tightly coupled computers. The computers can themselves be a parallel machine like the Sequen, a network of several single processing machines like the Suns or a combination of both single and multiprocessor machines. Problems are organized with a master and several slaves. The master serves as a controller for the problem, passing work to the slaves, performing I/O, creating the slave processes, and ensuring that all slaves have completed their work before stopping the program. The Sequent has been used as the master and each Sun as a slave in performing parallel execution. The tools controlling the parallelism cause the actual computer organization to become transparent. That is, the tools can be used on tightly coupled computers as well as loosely coupled computers. Any processor on any machine can serve as the master while any other processors can serve as a slaves.

The program consists of 3 physically different programs. A C program defining the role of the master, a C program defining the role of the slaves and a C program defining what processors are available comprise the complete package. Each slave can execute the same program or have different programs defined. The implementation is for a MIMD machine. All of the C programs are simply augmented with macros to control the parallelism. One can think of the programming language as containing a set of new instructions or built in functions, which are actually implemented as C macros, along with the standard C programming instructions.

Parallel Programming Tools

The tools consist of a set of macros written in C. The solution to the problem, search in this case, is written in C using the tools to control parallelism and communication between processes. There are a set of macros to implement the send and receive paradigm necessary for loosely coupled processors. A second set of macros implement parallelism on a tightly coupled computer like the Sequent or the Encore. Different memory models are supported with the macros. Some of the macros are used in both types of paradigms. Reference (Lusk 1987) defines the tools as several C macros that control parallelism and assist the programmer in construction of the parallel program by removing low level details of creating slave processes and controlling the synchronization of those processes. The tools are designed

to run on any UNIX based computer using the standard M4 macro processor.

The following outline shows a common program organization for the master to implement parallel programs using the message passing paradigm. The program follows the basic design of any C program with storage being declared first. Part of the storage requirement is a declaration of the types of messages to be passed. These define the responses to be made from the slaves, how subproblems are passed to the slaves, and control signals that tell if a subproblem is being sent or if the program is finished. C is extended to include a data type of PROC_ID that stores the process id's of the slaves. Global memory is allocated that allows the master to maintain data necessary to monitor the slaves. A REMOTE_CREATE macro creates the slave processes. Macros exist so that the master can keep track of slave processes that are working and those requesting new problems through their process id's. The master performs the service of a monitor common to many operating systems and discussed in typical operating system books. The master defines what a problem is, passes each problem to the slaves and determines if the program is finished. Upon completion, the master issues an END_SIGNAL to inform the slave processes to kill themselves off. The master performs all I/O operations as necessary for the problem being solved. Problem solutions can be organized as shown in figure 1. Each slave communicates with the successive slave with the final slave sending its message back to the master or the communication can be established as shown in figure 2 with slaves only sending messages to the master. The SENDR macro is used to send messages. SENDR contains parameters indicating the id of the receiver, the message type being sent and the pointer to the message. The macro expansion contains all code to cause the actual message transfer to occur. The RECEIVE macro allows a message to be received.

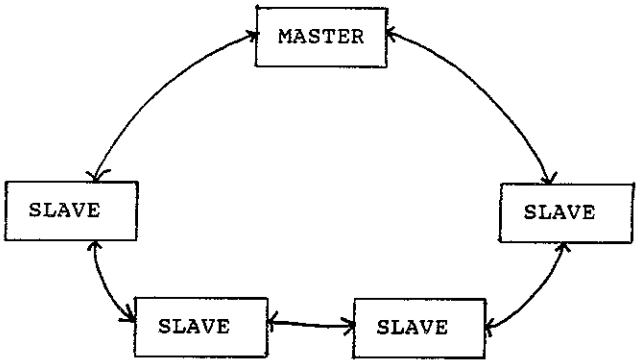
The slave program defines the same message types as the master, which is necessary so that the slave can receive the structure of the message sent by the master and vice versa. The slave receives a subproblem via the RECEIVE macro just as the master receives messages from the slave using the RECEIVE. Messages are returned to the master in the same manner as messages are sent from the master, using the SENDR macro. The structure of the slave program is that of a typical C program with message types declared, followed by the logic of the program that determines the function of the slave.

The third program necessary to implement send and receive on loosely coupled computers defines a PROCESS_GROUP. The process group defines the group of processors that will be involved in this problem solution.

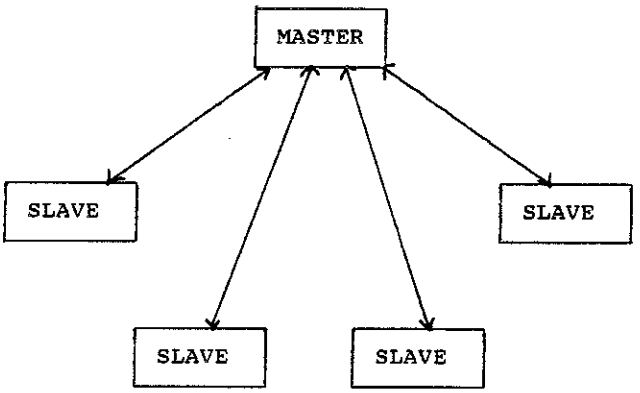
While the tools exist to assist the programmer in developing a parallel solution to problems, the programmer is responsible for all logic required to solve the problem. If one process finds a solution or is capable of pruning some of the search tree, one of two events can occur. That slave can notify the master that will send a message to the other slaves to modify their search space or the slave can communicate directly with other slaves, as determined by the logic of the program constructed by the programmer. The logic of the problem in the program will define how this is to be accomplished. The C macros only provide a mechanism for passing messages.

The C macro tools are constructed so that they can be used on single processor machines as well as multiprocessing machines. The slave processes will be time sliced so that the program will run just as though it was on a parallel computer. Of course, the speed ups will not be realized but the logic of constructing parallel programs still exists.

A set of macros exist for tightly coupled computers as well as the message passing paradigm. Since message passing can be simulated on shared memory machines, only those macros are discussed in this paper. The macros are designed to be highly portable in order to run on any memory model. Memory models are distinguished by the way shared memory is handled. When a data item is declared



SLAVES COMMUNICATE WITH EACH OTHER
FIGURE 1



SLAVES COMMUNICATE WITH ONLY THE MASTER
FIGURE 2

global, the memory model exemplified by the Sequent provides a copy of the data item given to the slaves. The Cray, for example, lets the slave access exactly the same data item. This is a problem only in the shared memory paradigm or the simulated message passing on the shared memory machines, since the message passing paradigm is implemented on distinct computers with the information passed from one machine to another. Within this paradigm, the kind of memory model is not a factor. The macros are designed to accommodate all memory models so the programmer need not be concerned about the model currently being used.

CONCLUSION

This paper points out that traditional AI problem solving techniques change when parallel implementations are considered. Real speedups occur when problems are solved in parallel and some problems that cannot be solved in reasonable computer times due to the nature of the search space can be solved when parallel solutions occur, since processors will pursue different parts of the search tree. Of course, single processor machines can arrive at similar conclusions if analysis of the search tree can determine what portion of the tree should be searched first. Parallelism can only reduce computer execution times, it cannot make a problem that is unsolvable by single processor machines suddenly solvable by parallel machines. The set of tools in the form of C macros allow the user to construct and implement parallel programs at a higher level than traditionally allowed by parallel systems.

REFERENCES

Luger, G.F., Stubblefield, W.A., Artificial Intelligence and the Design of Expert Systems, Benjamin Cummings, 1989.

Hoare, C.A.R., "Monitors: An Operating System Structuring Concept", Communications of the ACM, pp. 549-557, Oct. 1974.

Lusk, E.L., Overbeek, R.A., et al, Portable Programs for Parallel Processors, Holt, Rinehart, and Winston, 1987.

Datapac: A Parallel Reasoning Forward Chained System

Onzik Kim and Lawrence O. Hall
Department of Computer Science and Engineering
University of South Florida
Tampa, Fl. 33620
Ph. 813-974-4195; e-mail(csnet):hall@usf.edu

ABSTRACT

Parallelisms on various scales can bring a considerable speed-up in a rule-based reasoning system. The Datapac system is a parallel hierarchical forward chaining inference system to maximize the inferencing speed in multiprocessor environment by employing the parallelisms in a full range, which includes parallelism between different rule inferences as well as parallelism within a rule inference. The hierarchical forward chaining inference system is the rule based system where the rules are arranged in hierarchy according to the precedence relationship so that each rule is examined only once for firing. Rule level parallelism allows every rule to try to start its inferencing process as soon as it satisfies the precedence conditions. Thus, control of the inference flows from the top of the rule system to the bottom. This processing system, which could be modeled (and has been run) on a multiple instruction multiple data (MIMD) machine, has been simulated by Multilisp in this research. Performance evaluations demonstrate that this hierarchical forward chaining system (Datapac) provides a significant speed increase in simulated environment and real gains in multiprocessor environment.

INTRODUCTION.

Rule based forward chaining inference systems are one of the basic methodologies widely applied to the field of AI and Expert systems. There have been many kinds of efforts to improve these forward chaining systems in terms of speedup as well as various problem solving abilities. Many of these have involved developing parallel versions of OPS5 [FG84, HS86, Ok84]. We take a very different approach.

It is known that when a system works on the problem with a large knowledge base of rules and facts, lots of inferencing, which implies time consuming pattern matching and searching operations of a large knowledge base is required. It can result in explosive degenerations of inferencing performance in terms of speed and efficiency. This difficulty is one which this work attempts to address in terms of the limits of and the advantages of parallelism applied to the problem.

The Datapac system is a parallel hierarchical forward chaining inferencing system to maximize the speed-up in multiprocessor environment by allowing every rule to have a processor from the very start and by not allowing premature inferencing. The system finds implicit structures in a rule set, gives explicit structure to it partitioning it into hierarchically related subsets of rules and processes it in parallel with multiple processors.

In the preprocessing stage of the system, the rules are rearranged in a hierarchy according to the preceder-successor relationship between rules. If a rule's conclusion appears in another rule's premise part, the former is a preceder of the latter and the latter is said to be the successor of the former.

The *level* of each rule is an integral kind of representation of this inter-relationship between individual rules. The control of inferencing flows from the top to the bottom level in the rule hierarchy. It does not necessarily mean that rules are processed level by level collectively. This preprocessing of hierarchical rearrangement, together with other proper levels of parallelism provides subsequent aid to achieve the fully underlying rule level parallelism.

Initially, an unlimited number of processors is assumed to be available for our system. All the rules in the system are given one processor respectively and start inferencing at once. What controls the reasonable transition of inferencing is the preceder-successor relationships between processors. A rule processor won't try matching its premise to the working memory for rule firing until its every preceding rule's firing is completed.

The working memory(WMP) contains all the facts which are either initially given or derived from previous

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

inferencing. Each currently active rule has its antecedent matched against working memory, if it is to be ever fired.

After a rule is fired it sends the result to its succeeding rule processors immediately, enabling them to start inferencing. This processing system has been simulated by CSIM Multilisp in which the C-based simulation language CSIM [Sh86] was integrated into Multilisp at MCC [KM86]. Multilisp is a dialect of Lisp which provides some parallel structures mainly related to the spawning of new tasks. Multilisp's principal construct for both creating, delaying, deleting and synchronizing tasks is the *future*. The construct (future X) immediately returns a future value spawning a task for evaluation of X, to replace the future value to the actual value. The future is said to be initially undetermined; it becomes determined when it's value has been computed. An operation that needs to know the value of an undetermined future is suspended from its execution until the future is determined. But many operation such as assignment and parameter passing do not need to know anything about the values of their operand and may be performed on an undetermined future. Theoretically, the use of future naturally provides an unrestricted degree of parallelism to the systems.

Through the testings, this hierarchical forward chaining system (Datapac) shows a considerable improvement in speedup. For example, in a rule base where 15 non-variable rules on average reside in each level, the speedup ratio of parallel run to sequential run was over 9. For a variable rule system of 100 rules, where 20 variable or constant rules on average reside in each level, the speedup ratio was over 10.

PARALLEL HIERARCHICAL FORWARD CHAINING.

The knowledge base of this rule-based system consists of facts (propositional assertions) and rules (productions). The rule has the form (L R P C) in which premise P and conclusion C are augmented by the rule level number L and rule serial number R. P is not necessarily a single clause or predicate. It can be any combination of conjunction and disjunction of clauses or predicates in any depth. C can be either single clause or predicate. In addition to the LHS and RHS each rule of Datapac is given a serial rule number and level number which is assigned '?' initially and corresponding level number after preprocessing is done. The following is an example to illustrate the general form of knowledge representation in Datapac.

```
(1 3 (rand (ungulate) (has-long-neck)
     (dark-spots) (has-long-legs))
     (giraffe))
(2 4 (rand (bird) (not-fly) (swim)
     (black-and-white)) (penguin))
```

In this forward chaining inference scheme, the rules are rearranged in a hierarchy. There are two types of information which are extracted from a given rule set and used to reshape that given rule set with explicit structure.

These are the *level* of each rule and the preceder-successor relationship between rules. The former is a global and integrated representation of rule hierarchy and, thus, useful for centralized control of the inference process where the number of processors are limited, while the latter is a representation of hierarchy between individual rules and even between clauses of rules, hence it can be a powerful tool for distributed control of the inference process.

The top level consists of those rules which are led to by no other rules in the sense that no clauses in their antecedents are conclusions in any of the other rules in the system. Each other processing level will consist of a set of rules which have conclusions from the preceding level as clauses in their antecedents. The level which provides the answers to the system will consist of a set of rules whose conclusions are not to be found in any other rules antecedent.

The preceding rule set of a rule is the set of antecedent rules whose conclusion is directly linked to any premise of a rule. The successing rule set of a rule is the set of rules which are directly affected by the truth of a rule's conclusion.

Example.
(? 4 (OR (s3) (s2)) (s1))
(? 5 (s4) (s5))
(? 14 (AND (s5) (s1) (s9))

Rule 14 needs the result of rule 4 and rule 5's inference. So, there's precedence relationship of (4, 5 < 14) while there's no relationship between rule 4 and rule 5.

A successor rule can start its inferencing process only after its preceder rule's inferencing is done. Parallel Hierarchical Forward Chaining means parallelism is open to every rule to have a processor with a constraint in that parallelism which comes from the preceder-successor relationship between rules.

Subrule parallelism has less meaning in this forward chaining system compared to the Backpac, backward chained knowledge system developed at USF [Hl88]. In the Backpac system, a rule could be represented as a cluster and subrule parallelism brings other rule processors spawning which can bring again other subrule parallelisms. So subrule parallelism actually embraces enormous amount of parallelism. In the Datapac, on the while, subrule parallelism means only the several clause processors dedicated and limited to the job of checking the truth of each premise clause, each of which is taking charge of matching one clause against working memory.

The Start Up (and distributed control).

All the rules in the system are given one proces-

sor respectively and start inferencing at once. But what controls the reasonable transition of inferencing is the preceder-successor relationships between processors. A rule processor can not try matching its premise to the working memory for rule firing until its every preceding rule firing is completed. After a rule is fired it sends the result immediately to its succeeding rule processors' working memories, immediately enabling them to start inferencing. Especially, the set of all initial and derived facts from individual rule processors are collected and posted in a global working memory named the post office working memory.

THE IMPLEMENTATION.

Largely, this program consists of preprocessing part and inferencing part. Figure 1 shows the whole system scheme.

The preprocessing part takes charge of reorganization of data to be processed. The term *data* means a set of rules and a set of facts. But here especially the set of rules is considered to be reorganized for the system's efficiency and speedup. To do this, the role of this part is to find the precedence relationship of each rule from an input rule set.

Moreover, the preprocessing part partitions the rule sets into many differential subrule sets according to this precedence.

preprocessing
Find levels of each rules(routine FIND LEVEL)
Cluster rules according to the level(routine EQISET)
Find the preceding rule set of each rule(routine PRECEDER)
Find the succeeding set of each rule(routine SUCCESSOR)

inferencing
Initialize the inference engine(routine INITI1, INITI2)
Dispatch the task of inferencing to the inference engine (routine REORDER-SET)
Do the rule level inferencing (routine R-ENGIN)

Figure 1: Processing Structure.

FIND-LEVEL successively calls itself to find out the level of its antecedent rules. The idea is that any rule which is proved to have a preceding rule is in a level which is one step higher than that of the highest level of its preceding rules. The number of tasks spawned is increases quickly to cover the whole rule space.

After every rule has its value of level, they are partitioned into subsets in which every rule has the same value of level. It is nothing but a filtering of the whole rule set with the level number.

These preceding rule sets and succeeding rule sets, which can be collected simultaneously during FIND-LEVEL execution, will play an important roles in the communication between rule nodes. Some rule processors are waiting for signals (saying ok for a start from preceding node.) On the while, rule processors are sending signals to the succeeding nodes which are waiting for that signal.

Once reformation of the input rule set is done, they can be a reference for any processing of arbitrary input facts. In other words, the processed rule sets partitioned still retains the same function although arbitrary sets of facts may come into the inferencing routine.

For actual real-time inferencing, working memories with given facts are assigned to each rule processor. Rule id numbers are rearranged to be dispatched to each processor in the order of precedence, which means that no rule comes after its succeeding rule. (arbitrary order between 2 rules with no precedence does not matter).

Every rule is assigned a processor initially, that is to say a future value is assigned to its flag spawning an inferencing process. The values' future will become 1 after corresponding rule inferencing is completed. After this, R-ENGIN which is a de facto inference engine is called. R-ENGIN first checks the conditions to start inferencing, then it checks the condition of rules and fires the rules accordingly.

The premise processor deals with the and-or combination of propositions, while the duty of finding out whether an individual proposition is true or not (according to the WM contents)is the charge of individual clause matcher. The clause matcher tries to match an individual predicate to any items in its own working memory. The premise processor then makes the final decision about rule firing by synthesising the returned truth values of each predicate along with its logical connectives.

For variable rule inferencing MULTI-MATCH works on the premise of variable rules to try to match any complicated variable predicate to its working memory and produces a set of possible variable bindings. In this variable premise matcher, subrule level parallelism is not like that of non-variable rules because of the complexity and a need of consistency in variable rule binding refrain the premise matcher from being forked into clause matchers. Instead, it has parallelism between different subfunctions for the part of the match process which otherwise reveals a bottleneck of the processing. Then, BUILD-CLAUSE uses the derived variable bindings together with the conclusion of the rule to produce new facts with instantiations of variables in the form of conclusion predicates.

In case that the decision is fire the rule, the newly derived facts are added to the working memories of each succeeding node and a global working memory named post office working memory.

RESULTS.

This processing system run has been simulated by CSIM Multilisp for performance testing. In our simulations, it is assumed that an unlimited number of processors are available. The results are reported in terms of speed-up and the maximum number of tasks which ran in the parallel trials. The speed-up is measured by the best sequential version versus the best parallel version. The times are divided thereby providing the reported number.

Some artificial and real domains have been applied to the system. The results show that this hierarchical forward chaining system (Datapac) can provide a considerable improvement in terms of speedup. For example, for a 90 rule set without variables (constant) where 16 rules on average are in each level, the speedup ratio of parallel run to sequential run was over 9 with a maximum of 22 processors used. For a 100 variable rule system inferencing where 20 variable or constant rules on average reside in each level, the speed-up ratio of parallel run to sequential run was over 10 with a maximum of 180 processors used. A rule base consisting of 25 variable rules gave a speed-up of about 5 times with a maximum of 30 processors.

A knowledge base which consists of 64 rules and can diagnose fevers was a real example which has been used in our system. It showed a simulated speed-up of about 12 times with a maximum of 43 processors.

SIMULATION VS. AN ACTUAL ENVIRONMENT.

The results above indicate that the current version of Datapac has already been developed to the point that its speedup goes over the postulated maximum of rule level parallelism (which is equal to the average number of rules in a level).

In this system, only one step deeper level from the rule parallelism is employed. Going further into deeper levels did not compensate the cost of overhead of task spawning and shows no improvement in performance. Generally, for finer grained parallelism other than the rule level's, the synchronization method used is more likely to be a communication system between nodes, where its communication overhead becomes one of the critical problems.

System performance and the amount of speed-up are also sensitive to both the sample and machine environment. The performance of Datapac in the CSIM Multilisp is related to the way the rules are related and how large the rule set (how wide and how deep) is, according to our experiments. In its current state, matching and instantiation in variable rule processing takes relatively large amount of time in processing of mixed (constant and variable) rule inferencing. That is the way in which variable rule inferencing shows a slight slow down in speed-up.

Naturally, there are differences between model and reality. The first concern of this research is to develop the optimal parallel inferencing scheme free from environmental constraints by keeping the system as general as possible. But it could not be totally free from the conditioning of the developing bed. One of changes in original scheme, in that point, is a global storage area named post office working memory(WMP). Each rule's working memory only contains minimal information needed for its own inferencing. This saves communications during the execution of the system. In the current version, WMP passively collects the newly driven data rather than going and taking them for a very large knowledge base inferencing, this working memory would be assigned one or more processors to collect the new data actively.

Results From The Encore Multimax.

The system has also been run on a 10 processor Encore Multimax [ENC87] in a version of Multilisp which runs on the Multimax. The Encore is a shared memory machine with all processors hung off a common bus. In addition each processor has 64K of local cache. The results from experiments on this system were as follows.

With a knowledge base about fevers which consists of 64 rules, the system speeded up by a factor of about 6.3 times with 9 processors. This particular knowledge base has been implemented without variables. This is consistent with our quest to determine the limits of parallelism in reasoning systems. However, a generally useful system will need the use of variables. Note, one processor is left to do system's sorts of tasks, such as run the login shell and handle ethernet traffic.

On eight processors a knowledge base of one hundred rules provides us with a speed-up of almost linear proportions (a factor of 7.3). This rule base contains variables and the variable pattern matching, which has been parallelized, makes the task granularity larger than it would otherwise be. With a twenty-five variable rule knowledge base with nine processors a speed-up of about five and a half times was observed. These knowledge bases have to do with relationships in a family. They are really based on tests of logical structure and have no real depth of knowledge in them. The 100 rule knowledge base contains 25 rules per level. The 25 rule knowledge base has five rules per level. Basically, a Model can not be a substitute for a reality. Though the results from the real machine is the easist justification of (usefulness of) a proposed scheme, the results from the model simulation could be a better suggestion for an idealistic machine structure itself.

CONCLUSIONS.

Fundamentally, a forward chaining system is a kind of exhaustive search system and it is commonly known that parallel architectures are very effective for that kind of search system. Significant speed-up improvement in

our system demonstrates that the system is a good parallel scheme for forward chaining rule based inference systems. In this parallel inferencing schemes, the central control part becomes relatively small by distributing most of its decision to each individual rule processor. That is another indication that the system is really close to the one of the idealistic parallel processing concepts that controls are distributed over individual processors. We see that as a knowledge base grows larger, further parallelism and speed-up is possible, almost linearly proportional to the number of independent sets of rules which means that there's no explosion with any big knowledge base (at least in the range of CSIM simulations and the inputs we used.) The superiority of parallelism to the sequential execution tells about the suitability of the algorithm in parallel processing. But the superority of our hierarchical inferencing algorithm to the *random* inferencing can not be overlooked. Building a robust construct for more speedy and efficient parallel logical and/or subprocesses will be one of the main breakthroughs for an upgraded parallel inferencing scheme which, with current Multilisp features, is not easily achieved. We observe that the size and the structure of the knowledge base affects the performance of the system.

Acknowledgments: This research was partially supported by NASA-Ames Research Center Grant NAG-2-487 and a minority add-on from Nasa headquarters. Thanks to Claudia P. Industrious for testing code on the Encore. Thanks to Encore Computer Corporation for the use of their machine. Thanks to Randy Osborne for suggestions and pointing out some deficiencies in an earlier version of Datapac.

References

[Fc82] Forgy, C.L. (1982), Rete: A fast algorithm for the many pattern/many object pattern match problem, *Artificial Intelligence*, 19, 17-37.

[FG84] Forgy, C., Gupta, A., Newell, A, Wedig, R. (1984), Initial Assessment of Architectures for Production Systems, *Proceedings of the National Conference on Artificial Intelligence*, Austin, Tx., pp. 116-120.

[ENC87] Multimax Technical Summary (1987), Encore Computer Corporation, Marlboro, MA 01752.

[Hl87] Hall, L.O. (1987), Architectures for Reasoning in Parallel, NASA-Ames Research Center, RCR branch report 2018, Moffett Field, CA.

[Hl88] Hall, L.O. (1988), Parallelism in Fuzzy Rule-Based Reasoning, University of South Florida, Dept. of C.S., Report CSE-88-00003, Tampa, Fl.

[Hl88a] Hall, L.O. (1988), Parallel Rule-based Algorithms for Reasoning Systems, AAAI Spring Symposium Series, Stanford, Ca., March.

[Hl88b] Hall, L.O. (1988), Parallelism Applied to Fuzzy Rule-based Reasoning, NAFIPS '88, S.F., Ca., pp. 81-85.

[Hr85] Halstead, R.H. (1985), Multilisp: A Language for Concurrent Symbolic Computation, *ACM Transactions on Programming Languages and Systems*, Vol. 7, No. 4, October.

[Hr86] Halstead, R.H. (1986), Parallel Symbolic Computing, *IEEE Computer*, August, pp. 35-43.

[Hr86a] Halstead, R.H. (1986), Concurrent Lisp Machines, M.I.T. Lab for Computer Science, To appear.

[HS86] Hillyer, B.K. and Shaw, D.E. (1986), Execution of OPS5 Production Systems on a Massively Parallel Machine, *Journal of Parallel and Distributed Computing*, 3, pp. 236-268.

[KH89] Kim, O. and Hall L.O. (1989), Datapac, Technical Report, Department of Computer Science and Engineering, University of South Florida, Tampa, Fl. 33617.

[KM86] Krall, E.J. and McGehearty, P.F. (1986), A Case Study of Parallel Execution of a Rule-Based Expert System, *International Journal of Parallel Programming*, Vol. 15, No.1.

[KM88] Krall, E.J. and McGehearty, P.F. (1988), Execution of Common Lisp Programs in a Parallel Environment, in Parallel Computation and Computers for Artificial Intelligence (ed. J.S. Kowalik), Kluwer, Boston, Ma., pp. 51-62.

[Ok84] Oflazer, K. (1984), Partitioning in Parallel Processing of Production Systems, *Proceedings 1984 International Conference Parallel Processing*, IEEE Computer Society Press, pp. 92-100.

[Re83] Rich, E. (1983), Artificial Intelligence, McGraw-Hill, N.Y.

[Sh86] Schwetman, H. (1986), CSIM Reference Manual, Microelectronics and Computer Technology Corporation, Austin, TX.

[Th86] Tanaka, H. (1986), A Parallel Inference Machine, *Computer*, V. 19, No. 5, pp. 48-54.

[Ul87] Uhr, L. (1987), Multi-Computer Architectures for Artificial Intelligence, Wiley-Interscience, N.Y.

CREATING ABSTRACT FEATURE SETS TO SUPPORT KNOWLEDGE-BASED REASONING IN DESIGN

Gale E. Nevill, Jr., John H. Garcelon, and Kent B. Tambling

Department of Aerospace Engineering,
Mechanics and Engineering Science
University of Florida
Gainesville, Florida 32611

ABSTRACT

Top down abstract refinement models appear well suited to preliminary design. The success of these models is highly dependent on the quality of the abstractions and features supporting the problem solving capabilities required at each level of representation used. Results given here include specific criteria of quality for individual abstractions and features, criteria for abstraction/feature sets at a particular abstract representation level and criteria for sets of levels of representation. Also presented is a suggested process for the generation of good abstraction/feature sets.

INTRODUCTION

The top down refinement model of design (Steinberg 1987, Nevill 1988a) seems well suited to the preliminary design of mechanical structures. The essence of application of this model is the creation of a top level abstract problem description which is refined using several abstract representation levels, each with increasing detail (decreasing abstraction), to a final design.

The success of this approach is critically dependent on the creation of good sets of abstractions and features at each representation level used. Here the term abstraction will be used to mean any entity, useful in the design process, in which less important details have been suppressed. These may be entities associated with the artifact being designed or the design process. A feature is any useful attribute associated with the artifact being designed. Features will be present at all levels of abstraction.

It is common for initial problem descriptions to be represented in terms having different abstraction levels (Tong 1987). This mixed level description provides the starting point for the task of creating the abstraction and feature sets required for each representation level, that is, a good set of sets. The task of interest to us is thus three fold: (1) to create a suitable set of representations (abstractions) at various levels between the top and primitive component levels, (2) to create good sets of features and abstractions supporting design problem solving at each of these levels, and (3) to create abstractions and features to support mechanisms for moving between these levels. Since our starting point is a description with mixed levels, this task will of necessity include both bottom up and top down explorations.

This paper describes the results of our experiences to date with the MOSAIC project (Nevill 1987, 1988b) in trying to carry out this three fold task as part of the development of an automated capability for the preliminary design of 2-D brackets and fixtures used in high precision machining. The paper is organized as follows. First, a general discussion of principles which have emerged is presented. This discussion is divided into sections on types of abstractions and features required, criteria for good abstraction/feature (A/F) sets, sources of abstractions and features, and a feasible process for generating good A/F sets. This discussion is followed by an example illustrating many of the points presented.

RANGE OF ABSTRACTIONS AND FEATURES

The requirement for a complete problem solving capability at each abstraction level means that a broad range of abstractions and features (A/Fs) are needed. A/Fs are required at each level to support reasoning about form and function and involve activities such as evaluation, generation, prediction, decision, implementation and notification.

Since a principal goal of the top down refinement approach is to create simplified problems at the upper levels, it is desirable that the A/Fs at these levels will be reasonably simple. In addition, there is a need to be able to move between representation levels and to make decisions involving multiple levels. Thus, A/Fs must support translation and transformation of the design artifact descriptions between levels and among multiple levels.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

CRITERIA FOR ABSTRACTION/FEATURE SETS

Creating good A/F sets involves criteria for abstractions and features considered both individually and collectively. Our experience with the MOSAIC project has identified the following criteria as significant.

Criteria for Individual A/Fs

The quality of an individual abstraction or feature is judged largely based on its ability to increase the efficiency of the design process. From an individual perspective:

(1) embody relatively important attributes (details) from the next lower level and omit less important ones,

(2) have a natural fit with the problem domain and design process,

(3) have some dominant, unifying concept which makes it relatively easy to understand, use and communicate,

(4) be compatible with the design problem solving tools which are to be used, human or machine,

(5) be compatible with the knowledge available regarding it and with the organization of that knowledge,

(6) be readily abstracted to create higher level A/Fs and be readily expanded or specialized to identify lower level A/Fs embodied in it,

(7) facilitate moving between levels of abstraction during the design activity.

A/Fs of course do not occur alone, they belong to sets associated with different levels. The quality of an A/F set must therefore be judged collectively and must include considerations of soundness (completeness), internal compatibility and efficiency.

Criteria for Sets of Abstraction Levels

Next consider the desirable number of distinct abstraction levels and thus of A/F sets. The primary criterion here appears to be fit with the task domain. Our observation is that there are usually a number of natural levels of abstraction which have evolved with the field. The naturalness is usually identified by a sense of conceptual cohesiveness about a dominant theme. Compatibility with these natural levels appears essential if human understanding and use of human knowledge is sought.

Our experience to date is that a majority of possible benefits occur from using a limited number of levels of abstraction and that additional levels should be added only when clearly required by the structure or complexity of the domain.

Criteria for A/F Sets at an Abstraction Level

Finally, criteria for good A/F sets at any particular level of abstraction are considered. From a collective perspective, a good A/F set should:

(1) be sound, that is, include all of the factors which are important at this level of abstraction,

(2) involve A/Fs forming an independent set to the extent possible,

(3) be compatible in the sense that operators and reasoning mechanisms fit with the features available for input and output descriptions,

(4) minimize the need for different kinds of tools and reasoning mechanisms by enabling different abstractions and features to share tools and mechanisms,

(5) involve A/Fs with similar degrees of abstraction, i.e. levels of detail,

(6) have a structure compatible with any structure associated with the design process model used.

SOURCES OF ABSTRACTIONS AND FEATURES

Abstractions and features in general come from knowledge associated with solving the design problem. Successful execution of the process commonly involves application of a wide range of additional knowledge to reason about the problem. This knowledge may be theoretical, expressing well known laws of nature, in which case it is commonly expressed in equation or other mathematical or logical form. Alternatively, the knowledge may be heuristic, commonly originating from unstructured experience. Heuristics usually express relations among problem and process parameters.

Commonly these relations have a sentence-like structure involving a subject-like part, a verb-like part and a noun phrase-like part. When knowledge is identified as relevant, the subject-like and noun phrase-like parts suggest features and the verb-like parts suggest abstractions of operators and transformations. A powerful aspect of the use of these sentence-like structures is that when any two parts have been created, they can be used as the basis for creating the third part. This process of identifying relevant knowledge and using it incrementally to create new A/Fs based on existing ones is the principal source of A/Fs.

A PROCESS FOR GENERATING A/F SETS

The generation of good A/F sets is itself a design problem. The starting point is the collection of A/Fs provided by the original initial problem description; the principal tools are our knowledge of the problem domain (both theoretical and heuristic), design processes, design resources, and reasoning processes such as induction and deduction.

Here focus will be on creating good A/F sets in new situations, that is when there is no

directly applicable past experience to reference. The following three stage process combining some structure with considerable iteration and opportunistic exploration is suggested:

(1) start from the A/Fs identified in the initial problem description. Follow a strongly opportunistic approach and use available knowledge to expand the set of possible A/Fs so as to roughly span the range of abstraction levels,

(2) apply the criteria identified above and select a set of basic abstraction levels to be used,

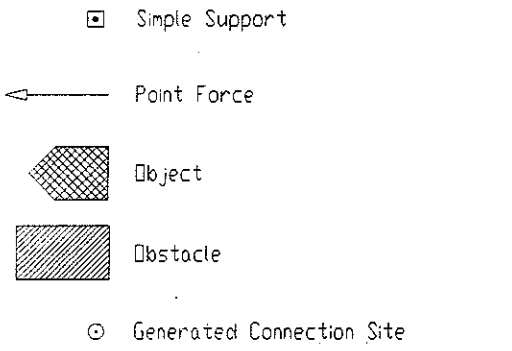
(3) considering the levels one at a time, cycle through them, refining each toward satisfaction of the criteria.

To date there has been little progress reported toward automating the task of creating good A/F sets.

EXAMPLE

The discussion thus far has been rather general. This section will present an example intended both to clarify the points made in previous sections and to indicate to some extent the basis of our experience. The example is typical of the class of problems addressed by the MOSAIC project.

Fig. 1: Legend

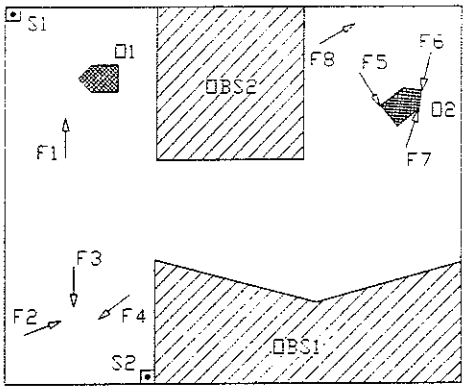


Consider the 2-D design problem shown in Fig. 2. The design task is to create a preliminary design of a minimum weight, kinematically stable structure, using the available simple supports, supporting all the point forces and objects shown, avoiding obstacles, and satisfying the requirement that the fundamental frequency be greater than f_o. Note that this initial problem description includes detailed information, such as the location and magnitude of the point forces, and high level abstractions such as "stable structure".

Generate Candidate Spanning Collection of A/Fs

Stage 1 of the process described above might proceed as follows. First, starting from the bottom level (maximum detail), A/Fs are suggested

Fig. 2:



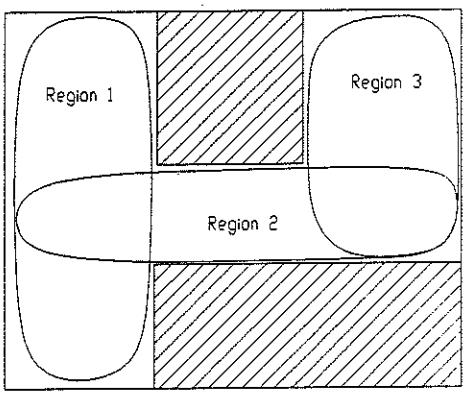
by domain knowledge. For example:

Knowledge that local irregularities in the boundary are usually important only locally suggests that obstacles (such as OBS1) be abstracted to their convex hull and that "convex obstacle" be used as an abstract feature.

Knowledge that connection of forces which occur close together is awkward and may involve substantial weight penalties suggests that forces in close spatial proximity (F2, F3, F4) be treated as a "cluster" and connected efficiently.

Knowledge that straight members with purely axial loads are particularly efficient suggests use of the abstraction "line of sight region" in which every point can see every other point (Fig. 3).

Fig. 3:



Statics knowledge regarding the requirements for kinematic stability suggest the attribute "stable region" for Region1 (Fig. 2) which has two simple supports (S1, S2) present, "connecting region" for Region2 which is empty and "unstable region" for Region3 which has the force F8 and the object O2 to be stabilized but no supports.

Knowledge-based reasoning about problem solution at the region level of abstraction suggests that the concepts of "stable structure" (in Region1),

"rigid structure" (in Region3) and "connecting structure" (in Region2) may prove useful. This in turn leads to the generation of abstractions such as "specify stable structure" as a possible subproblem and "build stable structure" as an abstract operator.

The constraint "fundamental frequency greater than f_o" can be transformed into the abstract requirement for "high natural frequency".

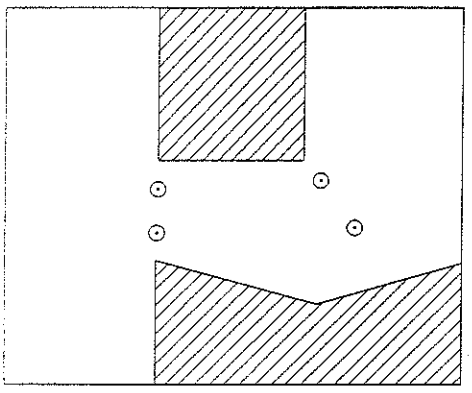
Once high level abstractions are generated, additional A/Fs may be obtained in a top down manner by decomposing and specializing them. The abstract goal of high natural frequency presents an interesting example.

General knowledge of structural vibrations provides the relation that natural frequency is determined by "stiffness", "mass" and "mode shape" and hence these abstractions are generated.

If the top level problem is decomposed into subproblems (for example, SpecifyStableStructure: Region1, SpecifyRigidStructure: Region3, and SpecifyConnectingStructure: Region2), then goals and constraints for each subproblem are required. Heuristic knowledge indicates that the stiffness of a structure close to supports is particularly important to the natural frequency, and that the mass located at points of maximum "amplitude" in the fundamental mode is particularly important. Another heuristic suggests that "distance" from the nearest support is a reasonable estimator of the amplitude of points vibrating in fundamental mode. These considerations suggest use of the abstractions stiffness, mass and amplitude or distance to characterize subproblems at the region level.

The importance of connections and attachments leads to identification of "connection sites" as significant features (Fig. 4). Knowledge of the behavior of connections leads to distinguishing attributes including "location", "width" and "moment capacity".

Fig. 4: Connection Sites



This process is continued until the candidate set of abstractions and features is judged to be sufficiently complete to form an adequate basis for selecting the set of levels of representation to be used. The process will be terminated here in this example for brevity.

Selection of the Basic Set of Abstraction Levels

The set of candidate A/Fs is now sorted and organized, seeking a minimum number of natural levels spanning the range from the top level of abstraction to the most detailed. In this case we have identified the following levels as appropriate for our preliminary design task.

(1) Top Level This level describes the problem in terms of A/Fs such as "complete stabilizing structure", "low weight", "low displacement" and "high frequency". It provides the most abstract description of the overall task.

(2) Planning Level This level describes the problem in terms of A/Fs such as "stable region", "force cluster", "rigid object", and "connected to". Operators such as "build stable structure" are available and subproblems such as SpecifyRigidStructure: Region3 are created. Problem solving at this level provides an abstract, conceptual plan for the preliminary design.

(3) Connection Level This level involves design decisions associated with connecting one partial solution structure to another. A/Fs such as "connection sites" with attributes such as "moment carrying" and "compact" and requirements such as "very stiff" and "light weight" are used.

(4) Detail Level This is the lowest level of abstraction. It involves details of configuration and attachment, types and sizes of components (beams, columns and rods), and material choices. Requirements are imposed in terms of displacements (expressed in mm for example) and frequency (expressed in cps).

Refinement of A/F Sets

The final stage is to refine the A/F sets within each of the representation levels selected. The Planning Level will be used for illustration.

All the problem definition parameters shown in Fig. 2 must be incorporated. This has been done by using the abstract features stable region, semistable region, unstable region, connecting region, force cluster, total mass, and connected to. The kinematic stability requirement is covered by using the concepts of stable structure, semistable structure, and rigid structure. The natural frequency requirement is dealt with using requirements on the abstract subproblem attributes stiffness and total mass.

As the A/F sets at each level are refined, compatibility between levels must be considered. In particular it is necessary to ensure that A/Fs at each level can be readily and meaningfully translated into A/Fs at the next lower level and that any additional A/Fs necessary for the operators that will perform these translations are available. For example, consider the natural

frequency requirement. The Top Level requirement for high natural frequency in this case is translated (with the use of considerable domain knowledge) into requirements on stiffness and total mass in Planning Level subproblem specifications.

Finally, refinement of the Planning Level set focuses on efficiency. Generally this involves aggregation or subdivision of existing A/Fs. New A/Fs are most often generated when new types of requirements, new knowledge or new capabilities are added to the system.

CONCLUSIONS

When top down refinement models are used in automating design, it is necessary to create a complete problem solving system at each representation level. The success of this approach is critically dependent on: (1) the quality of the set of abstract representation levels used, (2) the quality of the set of abstractions and features used at each level, and (3) the ease with which focus can be shifted between levels.

The criteria and process presented have evolved during several years work on development of the MOSAIC system for automated preliminary design of mechanical structures. Thus they are a response to the specific needs of this domain and to the approach taken. Considered in this context, they appear to be sound and valuable.

Overall, our experience has been that the process described is a comfortable, natural one which works well. We believe that the criteria listed are all of real significance and that the set of criteria is reasonably complete for our domain. It is difficult to estimate the extent to which the criteria and process may generalize to other design models and domains; undoubtedly significant changes would be required. On the other hand, many of the criteria identified seem to have broad applicability. In any case, the criteria listed should prove a useful source of possibilities and a helpful checklist for other domains and design models. Similarly, the process described should provide a starting point for development of processes better suited to other situations.

ACKNOWLEDGEMENT

The work reported was supported by the National Science Foundation under Grant No. DMC-8813808.

REFERENCES

Cunningham, J.J. and J.R. Dixon, 1988. "Designing with Features: The Origin of Features", In *Proceedings of the 1988 ASME International Computers in Engineering Conference* (San Francisco, CA). ASME, New York, NY, Vol. 1, 237-244.

Nevill, G.E., Jr. and G.H. Paul Jr., 1987. "Knowledge-Based Spatial Reasoning for Designing Structural Configurations," In *Proceedings of the 1987 ASME International Computers in Engineering Conference* (New York, NY). ASME, New York, NY, Vol. 1, 155-161.

Nevill, G.E. Jr., 1988a. "Computational Models of Design Processes", to be published, In *Proceedings of the NSF Grantee Workshop on Design Theory and Methodology* (Troy, NY). NSF, Washington, D.C., June 1988.

Nevill, G.E. Jr., L.A. Jackson, J.H. Clinton, 1988b. "Automated Hierarchical Planning for Structural Design," In *Proceedings of the 1988 ASME International Computers in Engineering Conference* (San Francisco, CA). ASME, New York, NY, Vol. 1, 395-402.

Steinberg, L.I., 1987. "Design as Refinement Plus Constraint Propagation: The VEXED Experience", In *Proceedings of the AAAI 87* (Seattle, WA). AAAI, Los Altos, CA, Vol. 2, 830-835.

Tikerpuu, J. and D.G. Ullman, 1988. "General Feature-Based Frame Representation for Describing Mechanical Engineering Design Developed from Empirical Data", In *Proceedings of the 1988 ASME International Computers in Engineering Conference* (San Francisco, CA). ASME, New York, NY, Vol. 1, 245-254.

Tong, C., 1987. "Toward an Engineering Science of Knowledge-Based Design", *Artificial Intelligence in Engineering*, Vol.2, Billerica, MA, No. 3, 133-166.

Ullman, D.G., T.G. Dietterich, and L.A. Stauffer, 1988. "A Model of the Mechanical Design Process Based on Empirical Data", Design Process Research Group Report DPRG-88-1-Revised, Oregon State University, (August)

Woodbury, R. and I. Oppenheim, 1987. "An Approach to Geometric Reasoning", EDRC-48-06-87, Engineering Design Research Center, Carnegie-Mellon University

ADAPTIVE INSTRUCTION THROUGH INTELLIGENT SCENARIO DEVELOPMENT AND MODIFICATION

Cheryl E. Bagshaw and Avelino. J. Gonzalez
Department of Computer Engineering
University of Central Florida
P.O. Box 25000
Orlando, Florida 32816

ABSTRACT

The use of computers for instructional purposes is steadily increasing, along with an emphasis on developing systems which create environments tailored to human beings. Artificial Intelligence techniques have been incorporated into these systems with an aim at developing better methods of modeling knowledge and intelligent behavior.

This paper addresses the tutoring component of one type of these systems, Intelligent Simulation Training Systems (ISTS). We focus on the ability to configure scenarios for the simulation which meet the objectives of a current lesson. The implementation of simulation-adaptation applied to teaching the domain of Air Traffic Control is presented.

INTRODUCTION

The application of computers to provide course content instruction in the form of drills, tutorials, and simulations is referred to as computer-aided instruction (CAI). Unfortunately, CAI systems are generally inflexible and provide no individualized instruction. The evaluation and planning process tends to be static, in that no modification of the lesson occurs until after the lesson is completed.

The introduction of artificial intelligence techniques were later incorporated into computer-aided instruction with an aim at developing better methods of modeling or simulating knowledge and intelligent behavior. Systems incorporating artificial intelligence are referred to as intelligent computer-aided instruction (ICAI) or Intelligent Tutoring Systems (ITS) (Sleeman and Brown, 1982).

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

Skills to be instructed may be divided into two types: cognitive skills and skills which focus on the ability to understand the time and space relationships of objects. Most prior work within ITS has dealt with the training of cognitive skills. Simulation-based training may be utilized for the training of skills required for the manipulation of objects within a time and space domain. This involves use of a simulation to dynamically display the status and location of objects. Depending on the domain of instruction, different procedures, rules, and criteria must be exercised by the student for correct manipulation of the objects. The objective of simulation-based training is to teach these procedures, rules, and criteria, and to provide a dynamic environment in which these skills may be trained.

Current research does not reflect much development of ITS within simulation-based training. The acronym "ISTS," for Intelligent Simulation Training Systems, will be used to distinguish such systems (Biegel 1988). Individualized training involves creating a task for a student which is appropriate, and providing assistance in a fit and timely manner. Three main areas of knowledge are required for individualized training and these may be divided and represented by three separate components. The expert module, the student model, and the tutor module are these components (Woolf 1984, 25-27).

The knowledge which embodies teaching strategies resides within the tutor module and is necessary to achieve an effective teaching system. The tutor module is the component responsible for utilizing the pedagogical knowledge to make these tutorial decisions. One goal of an intelligent tutor module is the ability to conduct a lesson according to the individual needs of a student. Each lesson defines an objective which states the skills and topics to be covered, and

the level of mastery to be achieved. The tutor module is required to configure scenarios for the simulation, in order to meet the objectives of a current lesson.

METHODOLOGY FOR SIMULATION-ADAPTATION

A tutor module needs to perform simulation-adaptation with the intent to provide tasks for the simulation which fulfill the objectives of the student's current lesson and to dynamically adapt the session to meet the needs of the student. This function distinguishes tutor modules for Intelligent Simulation Training Systems from those of Intelligent Tutoring Systems, which, in general, lack this type of simulation interface.

One method to accomplish task generation in an ISTS involves following a lesson sequence which has been previously outlined by a human instructor. This lesson sequence comprised of individual lessons concentrating on specific concepts. The lessons are logically ordered by increasing complexity and in a manner ensuring prerequisites for a particular lesson would have been satisfied by prior lessons. This methodology allows the tutor module to proceed through the subject matter in a manner defined as well organized by a human instructor. All concepts deemed necessary for coverage will be referenced by the tutor module. The effectiveness of this method relies heavily on how much information is rigidly specified in a lesson. If each lesson has a completely defined task to be used to exercise the topic specified by the lesson, the overall effect of task generation resembles a workbook. Each student proceeding through the sequence will receive the same tasks. The concepts of lessons and lesson sequences do, however, have potential for creative and flexible generation of tasks in ISTS.

Off-Line Task Generation in ISTS

As mentioned previously in this paper, the teaching of skills which involve understanding the time and space relationships of objects can be effectively accomplished by the use of a simulation. The tutor module must generate tasks for a student which reflect the current topic of discussion. The generated tasks will be presented by the simulation. Since the simulation is updated dynamically, the task initially presented will change as time passes. Therefore, the tutor module can only generate the starting conditions of the task. These tasks which are generated off-line will be referred to as "scenarios" because they specify the situations to be present within the simulation at the time in which the session begins running.

The method proposed for off-line scenario generation involves having the tutor module utilize a lesson sequence (see Figure 1). Lessons will be defined generally enough to allow the system to generate a different scenario for a lesson, each time the lesson is referenced by the tutor module. Each lesson specifies a description of what should be present within its corresponding scenario, but does not specify how this should be represented within the simulation. For example, a lesson may specify five objects to appear in a scenario, each with different capabilities in speeds, and heading in directions which will cause no future intersections amongst them, unless otherwise changed. The system will then have to calculate the required coordinates, headings, speeds, and other directives for the simulation which reflect the conditions specified by the lesson. The directives will be generated with as much randomness as allowed by the guidelines specified by the lesson. This promotes sessions which provide instruction specified by the current lesson, but with enough variability to prevent a rigid lesson sequence.

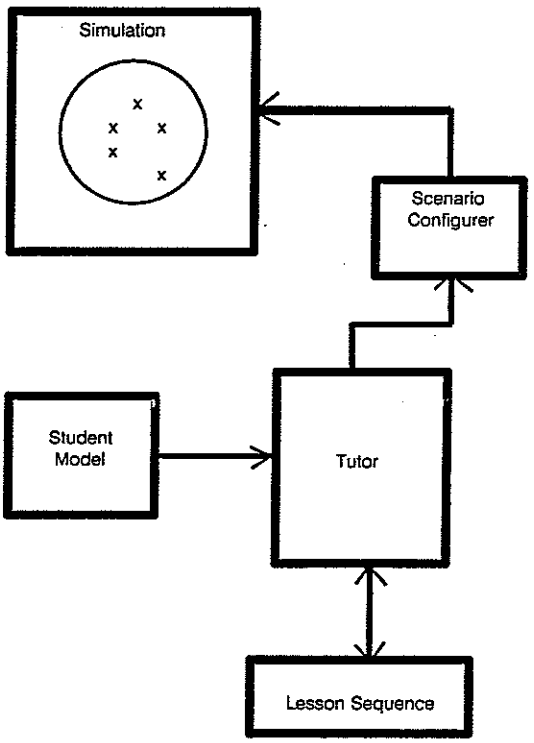


Figure 1. Interfaces Between the Components of the System.

On-Line Generation of Tasks

After the generated scenario has been loaded into the simulation, the simulation begins to run, thus initializing a student's session. The student will proceed with executing commands to control

the objects as required by the lesson. The performance of the student can be dynamically monitored and made available for the tutor module. The tutor module can therefore make decisions concerning the student's demonstrated performance of the skills covered by the present lesson. At times the tutor module may decide to increase the difficulty of the current session.

The lessons and lesson sequencing concepts can again be utilized to accomplish increasing difficulty. In addition to having the lessons specify, in general, what scenario should be present in the simulation at the start, the lesson can also specify techniques which the tutor module should use to generate additional tasks that increase the difficulty of the lesson. For example, a topic for instruction within Air Traffic Control is the maintenance of separation standards between aircraft. This topic, along with the directives for the starting scenario, will be defined by a lesson. Maintenance of separation standards becomes more difficult with the increase in traffic or with the introduction of inclement weather. Methods, such as these, for increasing the difficulty of the topic of separation standards will also be specified by the lesson. The tutor module can reference these methods at times when it is determined the student needs to be challenged. The tasks generated on-line to be added to the currently running simulation should again be created with as much variability as possible.

IMPLEMENTATION

The methods described above were implemented on a Symbolics LISP Machine and were developed for use in the instruction of handoffs and the maintenance of separation standards in the domain of Air Traffic Control. The Automated Reasoning Tool (ART), developed by Inference Corporation, was used to develop the rules necessary for making decisions concerning a student's past and present performance. It was also utilized to create a menu-driven authoring process to generate lessons.

Common LISP was used to implement the techniques which generate the directives necessary to drive the simulation. The simulation utilized for this research mimics a radar scope for an air traffic controller, and has the ability to display and update aircraft on the scope with time. The simulation was developed at the Simulation and Control Department of the General Electric Company in Daytona Beach, Florida, by Mr. Michael S. Kelsen and Mr. Blake Moselle, under the direction of Ms. Janice Eisele.

The Authoring Process

This process was implemented to allow a human instructor to create lessons and organize these within a lesson sequence for future reference by the tutor module. Whenever the system is loaded, the tutor's main menu is provided, which allows the user to choose between the authoring process or the initialization of a session (see Figure 2). If the authoring process is selected, the lesson menu is presented which provides options concerning lesson sequence construction.

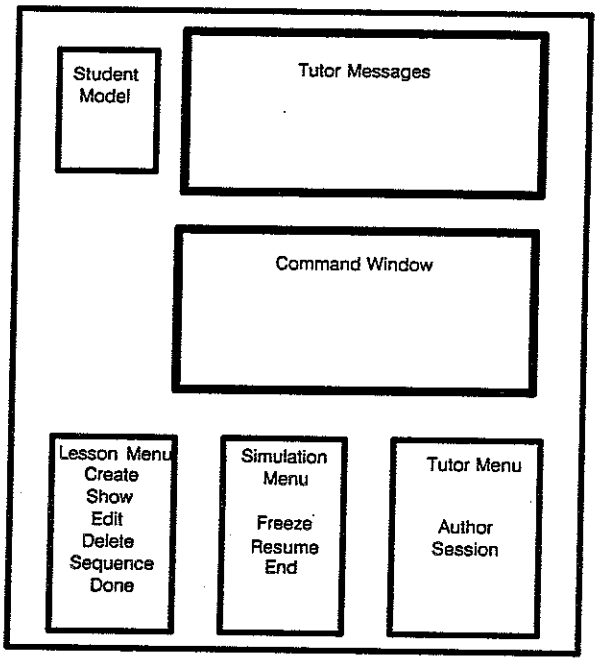


Figure 2. Windows and Menus for Tutor Communication.

Six functions may be selected from the lesson menu to assist instructors in the construction of lessons and sequences. The **create** function allows an instructor to create a new lesson and insert the created lesson within the lesson sequence at the position of his/her choice. The lesson menu provides three functions for use on existing lessons. The **show** option permits an instructor to view a lesson and verify the values assigned to the attributes. If an instructor feels that a lesson's specifications should be modified, the **edit** option provides the ability to change any values specified. The **delete** utility allows the deletion of a lesson from the lesson sequence. These functions must be performed on lessons existing within the lesson sequence and the system will notify the user of attempts to perform these functions on non-existing lessons.

When the authoring process is exited, the tutor's main menu reappears, permitting a student to initiate a session with the system. The implementation of the tutor module's actions at this stage is discussed next.

The Initialization of a Session

When a session is initialized by a student, the tutor prompts for information regarding the last lesson from which the student received instruction. This knowledge is used by the tutor to determine which lesson in the lesson sequence will be referenced for the student's current session. The specifications of the current lesson are consulted and utilized by the tutor to generate the starting scenario to appear in the simulation.

The calculations for generating the directives necessary for creating a scenario were divided into two classes. One class handled calculations for creating scenarios with problems existing between the planes appearing on the radar scope. The second class determined directives which placed planes on the scope with no problems present at the start of the session. Each lesson specifies if the planes appearing will have mixed abilities in speeds and if they will appear at different altitudes initially. This information is utilized to generate the student's initial scenario. As each plane is created, a scenario file is updated which will be loaded by the simulation when the scenario has been generated completely.

Once the initial scenario has been created and stored into a file, the system initiates the session. The simulation references the scenario file and displays the planes on the scope accordingly. The clock is then started and the simulation proceeds to update the status of the planes as time passes. The tutor is now ready to make decisions concerning when to challenge the participating student.

Challenging the Student

If the student is performing well during a session, the tutor needs to add to the simulation to make it more challenging. Implementation of this feature required simulating data which would be provided by a student model. This was accomplished by providing the user with a student model window which allowed the input of student model information.

Each lesson created contains a slot which holds the name of a function to be called by the system when the student needs to be challenged during that lesson's coverage. Whenever the simulated student data indicated good performance, the tutor referenced the current lesson to determine what function was listed in this slot. Any function listed had to have been defined and available for the system to call. The blackboard inferencing mechanism of ART was used for the dynamic monitoring of a student's performance and modification of the simulation. At the instant the simulated student data indicated the student needed to be challenged, this "fact" was posted on the blackboard which triggered the rules governing the on-line task generation scheme. This process was nearly instantaneous which provided a timely manner of updating the simulation.

The function defined for implementation by this system added planes to a currently running simulation. In Air Traffic Control, the increase in traffic increases the difficulty of maintaining separation standards between aircraft. Each time the simulated student data imply the student should be challenged, an aircraft is added to the simulation. The coordinates, headings, etc., for each aircraft are generated in manners similar to those calculations used to generate startup scenarios.

CONCLUSIONS

In the introduction of this paper, the need for individualized tutoring within Intelligent Tutoring Systems was stressed. This was the underlying approach taken for the development of techniques to perform simulation-adaptation for an Intelligent Simulation Training System.

The approach taken to perform simulation-adaptation proved to be effective by the resulting system. Air Traffic Control is carefully regulated and strict guidelines govern how controllers are instructed. The subject matter is taught in a well organized sequence which lends itself well to the methodology developed by this research using lessons and lesson sequences.

ACKNOWLEDGEMENTS

This research was performed in conjunction with Intelligent Simulation Training System (ISTS) project, IEMS department, University of Central Florida. The project director is Dr. John E. Biegel. The ISTS team members during the time of this particular research include Dr. J. Biegel, Dr. G. Brooks, Dr. J. Sepulveda, Dr. C. Lee, M. Draman, L. Interrante, G. Nadoli, N. Paz, C. Bagshaw, T. Sidani, C. Dixon, A. Roney, and J. Sargeant.

REFERENCES

Biegel, John E., Leslie D. Interrante, Jenifer M. Sargeant, Cheryl E. Bagshaw, Camille M. Dixon, George H. Brooks, Jose A. Sepulveda, and Chin Lee. 1988. Input and instruction paradigms for an intelligent simulation-based training system. In Proceedings of the First Florida Artificial Intelligence Research Symposium, 250-253. St. Petersburg: The Florida AI Research Symposium.

Lesgold, Alan M. 1987. Toward a theory of curriculum for use in designing intelligent instructional systems. In Learning issues for intelligent tutoring systems, edited by H. Mandl and A. M. Lesgold. New York: Springer-Verlag.

Sleeman, Derek, and John Seely Brown. 1982. Intelligent tutoring systems. New York: Academic Press.

Wenger, Etienne. 1987. Artificial intelligence and tutoring systems. With a Foreword by John Seely Brown and James Greeno. Los Altos: Morgan Kaufmann Publishers, Inc.

Woolf, Beverly Park. 1984. Context dependent planning in a machine tutor. Ph.D. diss., University of Massachusetts.

An Expert System for Managing Instruction in an On-line Learning Environment

By

John A. Scigliano, Don L. Joslyn, and Jacques Levin

Nova University

3301 College Avenue
Fort Lauderdale, FL 33314

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

This paper presents an expert system that was designed to manage an on-line university environment for graduate students. The focus of the paper is on the user-interface to the expert system to provide an understanding of the various screens that both students and teacher see as they use the system. The system was needed to solve several problems that emerged in processing large numbers of student assignments and projects received over telecommunications lines. Information overload in an on-line environment can be a severe barrier to effectiveness (Hiltz and Turoff, 1985). The purpose of the expert system is to provide an effective structure (an electronic educational environment) to assist faculty in developing and teaching courses, grading assignments, reviewing projects, answering students' questions, and advising students on their progress. The expert system provides both the academic and administrative functions necessary to the successful conduct of learning at the graduate level. The system works in conjunction with a real-time on-line group instructional utility called the Electronic Classroom or ECR (Scigliano, Joslyn, and Levin, 1989).

The graduate programs using the expert system enroll about 400 students in masters and doctoral programs with specializations in information systems, information science, computer science, computer-based training, and computer education. Students in these programs do much of their coursework on-line and use a personal computer to prepare assignments at their home or work. The two basic ways in which the students use the system are to transmit required assignments in the various courses or to ask questions of an advisor or teacher (the terms "advisor" and "teacher" are synonymous in the paper). Once the assignments or questions are received by the system, the system takes over and serves as a central communications center for the faculty and administrative staff. Tedious paperwork tasks are eliminated and electronic records are maintained for future reference.

The system was written in C and operates on a DEC 8550 mainframe computer with the DEC version of Unix* called Ultrix. Expert systems and AI in general have become prevalent in Unix environment where a host of tools make it possible to build modular systems (Landesberg, 1986; Anderson, 1986). The design presented in this paper consists of several modules including the electronic student, electronic teacher, electronic information utility, electronic question, and several staff modules for processing and displaying internal transactions. The expert system has been modularized to reflect the activities of students as they communicate with faculty. During this communication, the expert system stores the information in a way that enables it to be re-used in later communications, thus providing an instructional expert system environment.

DESCRIPTION OF THE EXPERT SYSTEM:

The expert system is based on an original design with a menu driven front-end that controls the selection of Unix "shell" programs. These programs form the "back-plane" of the system. The selection of any program depends upon the information already stored in the knowledge base. If the knowledge base contains the answer, then the answer is sent automatically to the student. If the knowledge base does not "know" the answer, the expert system selects the appropriate faculty member and forwards the question to him or her for action. The knowledge base grows as more and more transactions between student and teacher take place. The expert system consists of several modules as follows:

THE ELECTRONIC MENU (emenu):

This module presents the student choices that serve to control the other modules in the system. The "emenu" is the gateway to the expert system and provides easy entry to all the utilities. The menu is shown in Figure 1. The choices enable the student to: 1.) submit assignments to be read by a faculty advisor; 2.) submit questions for a faculty advisor to read; 3.) read news about the student's program; 4.) access the electronic library; and 5.) read projects that have been completed by other students. Items 6 and 7 on the menu are available to faculty advisors only. These items enable faculty to make responses to students questions and provide feedback to students on assignments. A display function

```
emenu

        * System type *

1.      Electronic Student      -> to submit assignments.......
2.      Electronic Question     -> to submit questions.........
3.      Electronic Information  -> to access the latest news...
4.      Electronic Library      -> to access the electronic lib
5.      Display programs        -> to access practicums & mfp's
6.      Electronic Response     -> for faculty only............
7.      Usage statistics        -> for faculty only............

Please enter your selection, or x to exit (1-7)
```

Figure 1 The Electronic Menu (the master gateway to the expert system)

* Unix is a trademark of AT&T and Bell Laboratories

enables advisors to view student records and to gain access to summative reports about the records by student or by course. The display feature is explained later in the paper.

ELECTRONIC STUDENT (es):

The electronic student module of the expert system "es" was designed to assist the staff in processing large numbers of assignments by making the work of both student and teacher easier. Students use the "es" program to submit work to the central program office for processing. When called through use of the "es" command, or through the selection of item #1 on the "ememu" described above, the electronic student presents an additional menu that contains a list of various graduate programs (etdais, etdatl, esmstl, etc.). This menu is shown in Figure 2. Once a student selects their

```
emenu

   * program *

1.    Doctor of Arts in Information Science..... -> etdais
2.    Doctor of Arts in Training and Learning..... -> etdatl
3.    Master of Science in Training and Learning.. -> esmstl
4.    Master of Science in Information Systems.... -> esmis
5.    Master of Science in Adult Education........ -> esmae
6.    Master of Science in Electronic Education... -> esmee
7.    Master of Science in Info Tech & Res. Mgt... -> esmit
8.    Doctor of Arts in Information Systems....... -> etdis
9.    Former Computer Education Doctoral Program.. -> etced
10.   New Computer Education Doctoral Program..... -> etedd
11.   Question/Response System.................... -> ercbl
12.   CBL Information Response System............. -> eircbl
13.   Student Information Response System......... -> eiruser

Please enter your selection, or x to exit (1-13)
```

Figure 2 The Electronic Student and the Electronic Teacher Menu

own program from the list a customized menu appears that shows the courses required in that program. For example, if a student selects item #8 from the menu, the Doctor of Science in Information Systems course list appears on the screen as shown in Figure 3. The student then selects the course title for a given course, for example, the course on Human Factors (item #4). When the student selects this item the list of assignments for that course is shown (Figure 4). Once the student selects an assignment,

```
esdis

   * Menu for DIS Courses and Projects *

1.    Computer-Based Research and Statistics..... (res)
2.    Database Management Systems................ (dms)
3.    Strategic Management for Information Systems..... (mgt)
4.    Human Factors.............................. (hum)
5.    Systems, Expert Systems and Artif. Intelligence.. (xai)
6.    Summer Specialty #1 Management Info Systems..... (su1)
7.    Summer Specialty #2 Emerging Technologies..... (su2)
8.    Long Practicum Proposal #1..... (pp1)
9.    Long Practicum #1..... ( p1)
10.   Long Practicum Proposal #2..... (pp2)
11.   Long Practicum #2..... ( p2)
12.   Major Field Project Proposal..... (mfpp)
13.   Major Field Project..... (mfp)

Please enter your selection, or x to exit (1-13)
```

Figure 3 Doctoral Course Menu in the Information Systems Program

```
esdis

   * Menu for DIS Tasks *

1.    Assignment  A  Journal Reading Reviews
2.    Assignment  B  Human Characteristics and Computer Interface
3.    Assignment  C  The User Interface Part 1 (Hardware)
4.    Assignment  D  The User Interface Part 2 (Software)
5.    Assignment  E  Environmental and Work Station Factors
6.    Moe Assign  1  Reading Program
7.    Moe Assign  2  Interviews with Organizations
8.    Moe Assign  3  Research Study
9.    Moe Assign  4  Case Analysis
10.   Moe Assign  5  Writing an Original Case
11.   Examination

Please enter your selection, or x to exit (1-11)
```

Figure 4 Assignment Menu for the course on Human Factors

in this case Assignment B on Human Characteristics and Computer Interface (item 2), the system prompts the student for a Unix filename, or if one is not available, the student can create a file using the vi editor in Unix (see Figure 5). Once the file has been inserted in

```
Sun Feb 19

Course   : Human Factors.....
Task     : Assignment  B  Human Characteristics and Computer Interface

Your advisor is: mientje (Dr. Mientje Levin)

Please enter filename:
```

Figure 5 Student Prompt to Enter an Assignment

the system, the student closes out the transaction and several automatic events take place:

1. The student's assignment is indexed and stored in an appropriate directory. A unique five digit sequence number is given each transaction.

2. An advisor (teacher) is selected by the system and an electronic mail message is sent automatically to that individual.

3. A queue entry is generated and stored in chronological order.

4. Several Unix shell programs are fired depending on the selection made by the student. After all the programs have fired, the student gets an electronic mail message indicating that the assignment has been accepted by the central queue.

Figure 6 provides an example of the message received by a student selecting Assignment C for Human Factors course.

ELECTRONIC TEACHER (et)

Assignments received through the student's selection of one of the "es" options are stored in a queue for each program. The electronic teacher module

```
>N 15 dis      Sun Feb 19 18:54  18/520 "esdis verification number"
 N 16 scigl    Sun Feb 19 18:57  30/687 "Assignment grade and current "
& 15
Message 15:
From scigl Sun Feb 19 18:54:43 1989
Received: by novavax.UUCP (1.2/smail2.5/02-19-88)
        id AA01450; Sun, 19 Feb 89 18:54:42 est
Date: Sun, 19 Feb 89 18:54:42 est
Message-Id: <8902192354.AA01450@novavax.UUCP>
From: dis (Dr. Info Systems)
To: scigl
Subject: esdis verification number
Status: R


Your verification number is: 12013
Please use it as a reference number.

Course   : Human Factors.....................................
Task     : Assignment  C  The User Interface Part 1 (Hardware)

Your Assignment (dddd) has been sent.

&
```

Figure 6 Electronic Mail Verification Message

takes over when any electronic student transaction is fired. The "et" module provides the teacher a means for viewing assignments submitted through the electronic student "es" program. The teacher controls the "et" module by typing et[program name]. The teacher is then presented with the list of assignments in the queue. An example of a "teacher" screen is shown in Figure 7. This presentation contains the sequence number of the transaction, the date it entered the

```
| que# |     date     |  student  | group | course |  task  | spec | advisor |
(11953) Fri Feb  3      dwayne       dis      mfp      PRAC      -     dennism
(11956) Sat Feb  4        katz       dis      ppl      prop      mi    dennism
(11967) Wed Feb  8        raab       dis      ppl      prop      xa    dennism
(12008) Sat Feb 18       perra       dis      xai         f      -     gforn
(12009) Sat Feb 18        tomt       dis      mgt         c      -     doctorj
(12010) Sat Feb 18       ericm       dis      xai         f      -     gforn
(12011) Sat Feb 18   patrickm        dis      xai         d      -     gforn

        Please enter your selection, or x to exit (1-12011)


Page  1 of  1
r - read,  m - mail,  l - print,  k - kill,  d - print ALL, o - open, w - work
```

Figure 7 Assignments Waiting in the Queue for Teacher Action

queue, the student's usercode and group, the course acronym in which the student is enrolled, the task or assignment indicator, the specialty for practicum proposals, and the advisor or teacher's usercode. Several responses are available to the teacher that include: reading the assignment, mailing a copy of it to their own or other usercodes, printing to the line printer, killing or removing the assignment from the queue, opening the assignment or project to public view, forwarding the assignment to a different teacher (called bouncing), or placing the assignment in a background or work-in-progress queue.

The teacher can also use the et utility to grade an assignment. The sequence of actions for grading any assignment is started when the teacher enters the "et" program and types the sequence number of any of the assignments listed in the queue. When the teacher begins the "grading" transaction for an assignment, several actions occur that include the creation of the required office records. The actual grade for an assignment is entered by the teacher through the use of a screen that contains the complete list of assignments for a student in a given course (see Figure 8).

```
(12013) Sun Feb 19      scigl      nova      hum       c       -    mientje |

Students current status (John Scigliano):

a - Assignment
b - Assignment
c - Assignment
d - Assignment
e - Assignment
1 - Moe
2 - Moe
3 - Moe
4 - Moe
5 - Moe
E - Examination

Valid grades are:  p (pass)  m (modify)  n (no)

Enter choice:
```

Figure 8 Teacher Screen for Entering Student Grades

The teacher enters a choice of p(pass), m(modify), or n(no) for NO PASS (doctoral courses are graded on a PASS/FAIL basis). In masters level courses, a letter grade can be assigned by the teacher.

Several other files or records are maintained by the et module. These records are a comment file with the teacher's feedback to the student, a status file that contains the grades for all assignments in this particular course for the student, and electronic mail is sent to the student. The format for this mail message is displayed in Figure 9. In this illustration, the student has received a passing grade for Assignment c (p_c). No other grades have been received from this student to date for the course. If a student is asked to modify any particular assignment (for example, Assignment d), the teacher would depress the

```
From: scigl (John Scigliano)
To: scigl
Subject: Assignment grade and current status
Status: R

Human Factors.....................................

This is your current status.
************************************************************
a       - Assignment
b       - Assignment
c       - Assignment p_c
d       - Assignment
e       - Assignment
1       - Moe
2       - Moe
3       - Moe
4       - Moe
5       - Moe
E       - Examination

******************************
(12013) Sun Feb 19  (hum c )
--More--
```

Figure 9 Electronic Mail Message Received By the Student Showing Grade

"m" option when grading the assignment (see Figure 8 above). and the notation "m_d" would be placed in the student's record. Once the student completes the

required modifications, the teacher can change the grade to a pass. The teacher has an option to send electronic mail copies of student records to other faculty members or administrators by typing in the appropriate usercodes in response to a prompt that asks if carbon copies of a grade report are required.

ELECTRONIC QUESTION (eq)

The system includes a module for students to get help with various facets of their program (assignments, registration, schedules, etc). In this module, the student has a menu to select the services required. The system has built-in intelligence to react to the student selections so that the appropriate office in the University receives the requests. Figure 10 shows the main electronic question (eq) for the entire University through Academic Computing Services, and Figure 11 displays the menu for the question utility for the Center for Computer-based Learning. As in the other modules described above, a queue is formed where each item is displayed for action by an official in the University or the appropriate academic center personnel.

```
        * Main Topics (Academic Computing Services) *

1.      The Electronic Classroom (ECR).....
2.      Communications (Tymnet, System access, emulation, etc.)
3.      Unix commands and utilities.....
4.      Unix Accounts (Addition, deletion, quotas, disk, etc.).
5.      Additional information on Unix utilities.....
6.      Unix Primer by Greg Simco.....
7.      Computer Languages (C, Pascal, GPSS, C-Pilot, etc.)....
8.      INGRES  relational DataBase Management System.....
9.      USENET (Worldwide News Network for Unix Users).....
10.     Miscellaneous.....
11.     Make comment(s).....
12.     Current Academic Computing Services Projects..9/26/88..
13.     Academic Computing Services NEWS (last update 12/2/88).
14.     Academic Computing Services C function library.....
15.     Request service from Academic Computing Services.....
16.     Thank an Academic Computing Services employee.....

Please enter your selection, or x to exit (1-16)
```

Figure 10 The Electronic Question Menu for University-Wide Academic Computing

```
        * Menu for CBL Question Main Categories *

1.      help              -> ecr, eicbl, ss, eqcbl, Unix..... (hlp)
2.      cluster seminars  -> seminars, hotels, schedules..... (sem)
3.      summer institute  -> hotel, schedule..... (sum)
4.      program news      -> CED news, CBL news..... (new)
5.      tutorials         -> database, statistics, systems, CBLUA. (tut)
6.      slides            -> database, statistics, systems..... (sli)
7.      concepts          -> PASCAL & C..... (con)
8.      class schedules   -> ecr, unix workshops, calendar..... (sch)
9.      job openings      -> universities, high schools..... (job)
10.     PEP               -> New Professional Experience Plan..... (pep)
11.     Data Base         -> Data Base Update via .plan..... ( db)
12.     Unix Help         -> communications, upload, download..... (tel)
13.     Graduation        -> Graduation Process..... (gre)

Please enter your selection, or x to exit (1-13)
```

Figure 11 The Electronic Question Menu for Computer-based Learning Center (CBL)

MAINTENANCE UTILITIES

Grading Module

Several other useful features are available for the teacher. When a student completes all the requirements for a course, the teacher can enter a grade for the student that will be posted directly to the student's permanent record file. From this file transcripts are printed according to the needs of the student. The data entry screen for the course grade program (cgrade) is shown in Figure 12. The teacher completes this form for each student in the class whenever a final course grade is required.

```
Program Name:

Student User Name:

Course:

Grade:

Term:

Press: <ESC> for menu, <TAB> next field, ^P previous field,
       ^F to move cursor right, ^B to move left.



Help Clear_Fields Display_Courses Write Exit :
```

Figure 12 Screen Used by the Teacher to Enter Grade for a Course

Display Module (dsp)

Several options are available to both the student and the teacher for viewing completed assignments or student records. The student can only review his or her record of assignments for any given course by typing the dsp command and the name of the course. The teacher can review a record for an entire class or for an individual student by using the appropriate display option labels. An example of a display for a single student is shown in Figure 13, and the display screen for an entire class is shown in Figure 14.

```
walt   (Walt   F. Dea )                                  Sun Feb 19 07:31:43 1989

Course  Grades

res     | p_a| p_b| p_c| p_d| p_e| p_f|
dms     | p_a| p_b| p_c| p_d| p_e| p_E|
mgt
hum     Course_Grade=P|
xai     Course_Grade=P|
ins
csw
sui     Course_Grade=P|
su2     Course_Grade=P|
pp1     Course_Grade=P|
p1      Course_Grade=P|
pp2     Course_Grade=P|
p2      |    |p_cs|    |p_cs|
mfpp    |    |m_CONC|p_PROP|
mfp     |p_MFP|

Last login: Sat Feb 18 08:38:15 1989
```

Figure 13 Display of a Student Record

```
dcourse dis xai

xai - Systems, Expert Systems and Artif. Intelligence..
              a    b    c    d    e    f    E
al          | p_a| p_b| p_c| p_d| p_e| p_f| p_E|
alfons      | p_a| p_b| p_c| p_d| p_e| p_f| p_E|
barne       | p_a| p_b| p_c| p_d| p_e| m_f| p_E|
chay        | p_a| p_b| p_c| p_d| p_e| p_f| p_E|
chen        |    |    |    |    |    | p_f| p_E|
che         | p_a|
cle         | p_a| p_b| p_c| p_d| p_e| p_f| p_E|
sbre        | p_a| p_b| p_c| p_d| p_e|    | p_E|Course_Grade=p|
perr        | p_a| p_b| p_c| p_d| p_e| m_f|
phyl        | p_a| p_b| p_c| p_d| p_e|    | p_E|Course_Grade=p|
smit        |    |    |    |    |    | p_f|
```

Figure 14 Display of a Class Record for the Course in Systems and AI

FUTURE DEVELOPMENTS

The design staff of the Office of Academic Computing in cooperation with the Center for Computer-Based Learning are working on several additions to the array of modules in the expert system. A high priority item is an evaluation module for maintaining results of course and program evaluations, an interface to the Ingres database management system, and the interface to the electronic library.

The Electronic Evaluation Module

This module is currently under development and will enable faculty and administration to display and analyze data collected from student evaluations of courses and programs. The module will provide a means for presenting to the student, all available online evaluation questionnaires, present the questions to the students for response, and then store the results for later display and manipulation. Once the data are collected by the system, the master evaluation files will be brought to date after each student entry. Teachers and administrators as well as external reviewers will be given access to the database as needed. A analysis feature will be a part of the system to enable any of the authorized users to display results of statistical treatments of the data.

Tracking System and Ingres Interface

An interface between the expert system and the RTI Ingres relational database is under development. This feature will enhance the tracking capabilities of the expert system. As a student submits a question or an assignment through the expert system, a query will be generated to the Ingres database to verify several things, for example, that the student is properly registered in the course, the course in which the student is enrolled, and other important characteristics of the student. Based on the results of the query, a message will be forwarded to both the student and the department office in charge of the indicated task. Other new features under development hold promise for making significant improvements in the way education is conducted in an on-line environment.

The Electronic Library (el)

A pilot on-line library project is underway that works in conjunction with the set of expert system modules described above. The on-line library has become a common feature in computer-intensive Unix environments at several major universities (Balkovich et. al., 1985; Morris, et. al., 1986; Updegrove, 1986). In the Nova University system at this time, students are able to request database searches through the assistance of a retrieval specialist, make requests for periodicals, and get reference help on-line. A list of holdings is presented to the students through a user interface and a database designed with Relational Technology's Ingres database system.

SUMMARY

In this paper an expert system was described that is used to manage an on-line environment for graduate degree programs. The electronic student and electronic teacher modules were presented along with other utilities that have been designed to make work on-line less tedious and more effective for both students and teachers. The menu structure, depicted through the use of figures in the text, is a key to the successful implementation of the system. Overall, the system has made it possible for the staff to process large numbers of assignments with little worry over the usual paperwork problems or communication troubles.

References

Anderson, R. H. and Greenberg, R. "Unix and AI: A Beautiful Marriage". *Unix World.* 3(8): August 1986, pp. 26-28, 30, 32-33.

Balkovich, E., Lerman, S., and Parmelee, R. "Computing in Higher Education: The Athena Experience". *Computer*, November 1985, pp. 112-123, 125.

Hiltz, R. and Turoff, M. "Structuring Computer-Mediated Communication Systems to Avoid Information Overload". *Communications of the ACM,* July 1985, pp. 680-689.

Landesberg, T. J. "Unix-Based Tools Help Develop Expert Systems". *Unix World.* 3(8): August 1986, pp. 34-35, 38-39.

Morris, J. et. al. "ANDREW: A Distributed Personal Computing Environment". *Communications of the ACM,* March1986, pp. 184-201.

Scigliano, J. A., Joslyn, D. L., and Levin, J. "The Non-School Learning Environment: ECR". *T.H.E. Journal,* 16(6); February 1989, pp. 63-67.

Updegrove, D. A. Computer-Intensive Campuses: Strategies, Plans, Implications. *Educom Bulletin,* Spring 1986, pp. 11-14.

A FUNCTIONAL (CONTINGENCY-BASED) APPROACH TO MACHINE LEARNING

William B. Noffsinger
University of Florida
Gainesville, Florida 32611

ABSTRACT

An approach to machine learning and knowledge-based systems is offered based upon a functional/ behavioral orientation rather than a cognitive one. The representational, structural, intentional and rationalist foundations common to artificial intelligence (AI) and cognitive science are discussed. This cognitive (strong - AI) position is then contrasted with a functional one based upon the operant psychology of B. F. Skinner and recent work by Winograd and Flores. The weak-AI and behavioral approaches developed by Skinner, Winogard and Flores are then discussed as supporting a more productive approach.

INTRODUCTION

Cognitive science has been described (Collins, 1977) as an interdisciplinary approach involving psychologists, computational linguists and computer scientists, particularly those investigating artificial intelligence or knowledge-based systems. The problems investigated by cognitive scientists have included knowledge representation, language generation and understanding, image analysis, inference, learning, and planning. The methodology used to investigate these problems has usually featured a balance between an analytic orientation (as in psychology or linguistics) and a synthetic orientation (as in artificial intelligence).

The dominant paradigm common to psychological and computational models in recent cognitive science has been largely isomorphic to that of its close cousin, cognitive psychology. However, there are some differences in methodology and subject matter that should be briefly mentioned.

Cognitive psychology is a branch of experimental psychology concerned with the modeling and investigation of mental processes and having a body of experimental methods that limit its scope. These experiments are intended to test competing hypotheses about cognitive mechanisms. Recently, information processing psychology (Gardner, 1985) has come into competition with more traditional cognitive psychology.

In information processing psychology, cognitive systems are understood by analogy to computer programs. Researchers in this area believe that the computer allows cognitive theories to be more complex and intricate but still remain under empirical control. Researchers involved in areas collectively referred to as artificial intelligence also use the computer as a tool but often have less interest in the development or testing of theories of cognition.

The orientation common to all the aforementioned areas: cognitive psychology, information processing psychology and artificial intelligence (Rumelhart et al., 1986) is characterized by a shared commitment to structuralism, mentalism, intentionality and rationalism. This list of 'isms' may seem at first glance rather broad, but as we will see later, these are the traditions along which these cognitive sciences have been conducted.

The thesis of this paper is that significant strides are more likely to be made in cognitive science, including artificial intelligence, if the intentional system (Dennett, 1978), composed of both software and hardware, can be made to be responsive to (i.e., learn in response to) environmental contingencies (Skinner, 1938; 1957). If this is possible, such as system may be capable of what Winograd and Flores (1986) refer to as 'commitment' and thus worthy of potential functional status as 'intelligent'. To use terminology introduced by John Searle (1969), this paper takes a 'weak approach' to artificial intelligence (AI).

RATIONALISTIC BACKGROUND

Mentalism assumes a representational theory of mind. Such theories of mind depend, to some degree or another, upon the representational hypothesis (also referred to as the physical symbol system hypothesis): the assumption that cognition rests on the manipulation of symbolic representations that can be understood as referring to objects and properties in the world. A very naive statement of this might go so far as to suggest the existence of 'engrams' or some other coded or symbolic distillation or

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

transformation of experience in the brain. This extreme view is not widely held today. A less naive view might suggest that cortical structures or chemistry may carry the coded knowledge of experience in some form or another.

Structuralist claims hold that the internal structure of a system (natural or artificial) largely determines the behavior of that system. Carried to extremes, such a position becomes anti-environmentalist, in the sense that environmental determinants of behavior are minimized.

Intentionality, (Dennett, 1971) is by definition that feature of certain mental states by which they are directed at or about objects and states of affairs in the world. Stated a little more concretely: An intentional system is one whose behavior may be 'explained' (in the layman's sense) by relying on ascriptions to the system of beliefs, desires, hopes, fears, hunches, purpose, etc. We human observers often take the intentional stance (Dennett, 1978) when we claim that a purely physical (and artificial) system may be so complex, yet organized and capable of seemingly orderly behavior, that we find it convenient or pragmatically necessary to treat it as if it possesses beliefs, desires, rationality, intelligence, etc.

The rational tradition is characterized by an implicit dependence upon representational theories of mind and has been the mainspring behind the modern emphasis upon, and successes with, technology and applied science. The rationalist tradition also features a typical process for problem solving or decision making (Simon, 1976):

1. Characterize the problem situation in terms of objects with well defined properties.

2. Find general rules that apply to the situation in terms of those objects and properties.

3. Apply the rules logically to the situation of concern, drawing conclusions about what should be done.

In the rationalist tradition language is seen as having a correspondence between words and what the words denote in the world, in which the meaning of the words or utterances in speech are fixed with little reference to the context in which they appear. Thus, the words, and their referents, have inherent qualities that exist due to logical or rational necessity, apart from any particular environment or context. This is basically an Aristotelian position.

OPERANT APPROACH TO LEARNING

In contrast to representational, intentional and structural approaches inherent in the rationalistic tradition, an operant approach to learning (also referred to as radical behaviorism) can be characterized by the following:

1. An environmental emphasis rather than a structural one. Behavior is a joint function of environmental events, and the organism's or system's own ontogenic history, and is not determined a priori by the structure of the system or organism. Of course, the structural characteristics (e.g., sensory capability) of an organism or system do set parameter limits upon its capacity to respond to the environment.

2. It is important to note that the environment may be internal or private with respect to the system or organism and not strictly external to the system, i.e., outside the skin of the system. Critics of behaviorism often misjudge the internal states, or private events, of the organism or system as being distinct from the environment.

   If it is erroneously assumed that environment always exists only outside of a system's or organism's skin, in keeping with a strong dualist (Cartesian) stance, then a structuralist position is diametrically opposed to a behavioral one. However, behaviorism is in fact not rigidly dualistic.

3. Behaviorism is a non-dualist (non-cartesian) position. Attention on the body-mind problem is replaced with a focus upon the explication of functional relations between the system's or organism's response (broadly construed) and the environmental conditions of which it is a function.

4. The simplist functional relation consists of a response and an environmental event consequence to that response. This response-consequent unit is called an operant. The scale and measurement properties of behavioral units are currently debated (Thompson et al., 1986). However, operants large in scale, such as those relevant to verbal or linguistic behavior, may be aggregates of smaller and more fundamental ones; albeit an empirical question.

5. The simple operant is a relation between responses (behaviors) and environmental consequences (which are stimuli), hence an R-S relation. A succession of R-S relations over time constitute an organism's or system's history with respect to that operant. We may use the following notation to describe such a history.

$$(R_1, R_2 \cdots R_n) \rightarrow (S_1, S_2 \cdots S_n)$$

6. The explication of functional relations using rigorous empirical techniques is pursued rather than favoring 'explanatory fictions' - Skinner's term for relying upon representational or structural mental states as inferred determinants or explanations of behavior. Behaviorism relys on the inductive method.

7. Rate of responding, speed of acquisition, and resistance to decay (extinction) of an operant are fundamental data in an operant approach to learning.

8. In an operant, the relation between a response

and its consequent environmental event(s) is described by a contingency of reinforcement. For example, reinforcing consequences are defined functionally as those which increase the probability or frequency of the preceding response.

9. Verbal and linguistic behavior is just as much in the domain of operant psychology as are more overt muscle movements or physical behavior. The investigation of verbal behavior does pose an additional challenge to the researcher.

In summary, the behavioral approach to learning stands in contrast to the traditional cognitive approaches with their representational, intentional and structural orientation in the rationalist tradition.

A FUNCTIONAL ORIENTATION

Winograd and Flores (1986) have also developed a careful and thorough critique of the cognitivist (or strong-AI) approach to system design. Their position has much in common with the characteristics of the behavioral paradigm briefly outlined above.

Winograd and Flores take a position that is strongly non-representational, discussing the futility of a representational orientation toward human cognition. Behaviorists also see no reason to postulate mental representations.

Winograd and Flores approach is also non-intentional. Despite the temptation to ascribe intentional characteristics to artificial systems (desires, beliefs, purpose, understanding, responsibility), non-biological entities lack other required attributes, such as capacity for commitment or need to reduce biological drive. Winograd and Flores might argue that humans or other organisms are capable of intentionality, whereas artificial systems are not. Behaviorists would argue that the intentional stance also brings little to our understanding of behavior and is best left out of scientific accounts of verbal or linguistic behavior, while recognizing that we might make intentional ascriptions to our own behavior or that of complex machines.

The approach favored by Winograd and Flores is also in disagreement with the rationalistic tradition. Winograd and Flores argue that, like intentionality and representation, the rationalistic tradition has not served system designers in their attempt to produce artificial intelligence. The attempt to model machines after intentional human qualities of rational decision making has failed. The Cartesian (mind vs. body dualistic) position inherent in the rationalistic orientation of cognitive science has failed to produce the 'intelligent' machines sought by artificial intelligence researchers. Also, the rationalist position has failed to support a scientific and rigorous understanding of verbal or linguistic behavior sought by behaviorists.

Wittgenstein (1963) is another analytic philosopher whose later work in the philosophy of language abandoned a metaphysical and formal logico-deductive position in favor of a strictly functional one. Wittgenstein's view has been compared with Skinner's behaviorism (Day, 1969) and found to be highly compatible. Wittgenstein's most well known quote is "the meaning is the use".

Winograd and Flores also cite the Speech Act theory of Searle (1969) which recognizes that speech occurs within a social context and is most productively studied from a functionalist perspective. In other words, the most productive questions about speech are those that ask what social or biological consequences are obtained for the system or organism engaging in speech. This orientation takes an instrumental and behavioral view by focusing upon consequences and functions of language.

Contrast this position with a traditional rationalistic/dualistic approach that suggests language and cognition exist on a primarily logical level apart from an organism's or system's need to operate and survive within an environment (Fodor, 1980; Chomsky, 1957).

Murry Sidman (1986) has made a provocative statement that agrees with the arguments made by Winograd and Flores:

> "Their lack of success in providing a useful account of contextual control [in language behavior] has prevented cognitivists form programming computers to comprehend the English language. One could begin to solve that particular contextual problem by programming a computer to acquire a repertoire of four-term contingencies [a unit of Skinnerian learning]..... The computer would therefore have to be made sensitive to consequences, some 'hard-wired' into two-term contingencies, and others derivable from three and higher term contingencies."

Sidman is speaking from the radical behavioral tradition which emphasizes explication of functional relations between a behaving organism and its environment (in the biological sense). A functional approach to machine learning advocates that the hardware/software platform used in cognitive science or artificial intelligence research be capable of the same functional relations but demonstrated in an artificial (Simon, 1969) and intentional system. Successful demonstration of contingency-based learning (in contrast to rule-based learning) in an artificial intentional system will present an alternative to the pitfalls of the formal and structural (mentalistic, representational and rational) cognitive paradigm described above. In addition, the machine-based functional contingency learning model suggested herein will not be subject to the limitations of naive S-R behaviorism (Chomsky, 1957).

Asking that cognitive science in general, and artificial intelligence in particular, adopt a functional (i.e., contingency-based) model of cognition and machine learning is to ask that the 'homunculus' be taken out of the learning model

that we are attempting to develop. As long as a structural or rational model (in the technical sense of the term) is assumed for machine learning, the enterprise will be chronically impaired by infinite regress (Dreyfus, 1982) as the ultimate location of cognitive 'understanding' is sought.

For example, if it assumed that machine or computational models of cognition must model human understanding, and humans must somehow have access to a mental representation of such understanding, then our hardware/software implementation must also own an analogous representation. In our machine such a representation would be in the form of a data structure. However, our machine would always lack, at some final point, sufficient 'judgement' to evaluate mental representations of cognitive function. We can not engineer an homunculus for inclusion in our machine to enable it to evaluate or comprehend. However, we may incrementally approach intelligent behavior by engineering ever more elaborate function. This is a weak-AI position.

Dreyfus (1982, page 202) has made this point cogently:

> "One response, shared by existential phenomenologists such as Merleau-Ponty and ordinary language philosophers such as Wittgenstein, is to say that such 'knowledge' of human interests and practices need not be represented at all. Just as it seems plausible that I can learn to swim by practicing until I develop the necessary patterns of responses, without representing my body and muscular movements in some data structure, so too what I 'know' about the cultural practices [or other deep knowledge] which enables me to recognize and act in specific situations has been gradually acquired through training in which no one ever did or could, again on pain of regress, make explicit what was being learned."

Other examples of representational computational theories in AI research and cognitive science include knowledge-frames (Minsky, 1975), scripts (Shank, et al., 1977) and other representational formalisms such as Bobrow's knowledge representation language (KRL; Bobrow, et al., 1975). All three of these works are significant efforts to model mental processes and produce knowledge-based systems founded on the representational, or strong-AI, position.

ALTERNATIVE FUNCTIONAL IMPLEMENTATIONS

After these arguments against cognitivism are heard, the researcher or developer of machine-learning systems must ultimately get back to work and assemble parts and logic to build working systems. If structural and formal models of representation and learning are abandoned for a functional approach, what are the implications for design and implementation of a working system? How will the hardware be interconnected if we eschew a representational model of cognition in favor of a system capable of contingency-based learning? Unlike the philosopher, the system designer must make engineering commitments to assembly of real parts.

The following works are developments that may show promise for a more functional (less cognitive or representational) approach to machine learning and a foundation from which a contingency-based approach may be engineered:

- Minsky's (1986) society theory of mind. This proposes a large aggregation of processing functions, none of which possess any 'intelligence', but when assembled are capable of emergent high-level functions to which observers often ascribe intelligence.
- Research in implementations of parallel distributed processing (Rumelhart et al., 1987) models such as Boltzman machines or Hopfield nets (Hopfield, 1982).

The two items in the list include a theory of mind that has a strong functional flavor, and a system architecture (parallel processing) that may provide a platform for systems with emergent intentional behavior. The society theory of mind describes a way of looking at intentional behavior (intelligence) that builds hierarchically upon ever more elaborate levels of function. Past a point, the elaborate functionality built upon simple elements begins to support emergent behaviors (responses to the environment) that encourage ascriptions of intentional behavior.

The work in parallel processing and network models suggests a machine architecture that may support systems capable of contingency-based learning. A system capable of such behavior may be described as follows.

As discussed in Winograd and Flores, artificial systems are incapable of making commitments, having responsibility, etc., in short - behaving 'intentionally'. However, following Sidman's ideas, a system may be capable of such intentional and emergent 'intelligent' behavior if it could be developed with (at least at the lowest level) a primitive unit of intentional necessity. By intentional necessity, we mean a basic 'need' to maintain a certain operant or respond to a particular contingency. Natural systems have appetites and 'drives'; without such a unit of intentional necessity, artificial systems do not and are hence incapable of intentional behavior. If a system could have such a low level intentional response, then despite its artificial status, if may have (in the functional sense) limited 'self-interest'. This limited self-interest may be adequate to render the system, after tremendous elaboration of function, capable of 'intentional' or 'intelligent' behavior.

As mentioned earlier, an organism's or system's behavior is a function of its ontogenic history, within limits set by physical or structural capacity. That ontogenic history is the accumulation of response-stimulus relations $(R_1, R_2 \cdots R_n) \rightarrow (S_1, S_2 \cdots S_n)$ that are antecedent to the particular behaviors (operants) or class of behaviors we observe at any point in time. In the case of our machine, an artificial

system, we are inclined to make an intentional ascription of 'intelligence', 'purposiveness', etc. because the antecedent history is unobservable. By explicating this history, the system's response may be seen for what it is - an orderly response to environmental events. The same explanation may be made for human behavior. When previously unseen antecedent history is explicated, human behavior is also seen as an orderly response to an environment.

We may allow our machine to gradually acquire a history by providing it with a facility to 'learn from experience', which is to say that it will store information obtained during each response-stimulus (R-S) transaction with its environment. Over time, a history will be acquired.

With that capability as a foundation, a large and complex system built around massively parallel architecture, would not just be left 'lost in its intentions'. Given sufficient sensory apparatus, an artificial system may be able to interact with its environment and to a limited degree make commitments (in Winograd's sense). An intentional artificial system may be capable of responding to behavioral shaping during the history of its interactions with its environment. By virtue of interacting with its environment, certain responses (behaviors) may be shaped over time such that the most beneficial ones are most likely to occur.

ACKNOWLEDGEMENT:

I wish to thank Dr. H. S. Pennypacker for a critical review of this paper.

REFERENCES

Bobrow, D. and Collins, A. M. (1975). (Eds.) Representation and understanding: Studies in Cognitive Science, New York: Academic Press.

Chomsky, N. (1957). Syntactic Structures, The Hague: Mouton.

Collins, A. (1977). Why cognitive science, Cognitive Science, 1, page 1-2.

Day, W. F. (1969). On certain similarities between the philosophical investigations of Ludwig Wittgenstein and the operationism of B. F. Skinner. Journal of the Experimental Analysis of Behavior, 12, 489-506.

Dennett, D. C. (1971). Intentional systems. Journal of Philosophy, 68, 87-106. Reprinted in Haugeland, 1981.

Dennett, D. C. (1978). Brainstorms, Montgomery, Vermont: Bradford books.

Dreyfus, H. L. (1982). From micro-worlds to knowledge representation: AI at an impasse. In Haugland, (Ed.), Mind Design, Cambridge, MA: MIT Press.

Fodor, J. (1980. Methodological solipsism considered as a research strategy in Cognitive psychology. The Behavioral and Brain Sciences, 3, 63-70.

Gardner, H. (1985). The mind's new science: A history of the cognitive revolution, New York: Basic Books.

Haugeland, J. (1981). Mind design. Montgomery, VT: Bradford/MIT Press.

Hopfield, J. J. (1982). Neutral networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences, USA, 79, page 2554-2558.

Minsky, M. L. (1975). A framework for representing knowledge. In Haugeland (Ed.), Mind design, Cambridge, MA: MIT Press.

Minsky, M. L. (1986). The Society Theory of Mind, New York: Simon and Schuster.

Rumelhart, D. E.; Hinton, G. E. and McClelland, J. L. (1986). A general framework for parallel distributed processing. In Rumelhart and and McClelland (Eds.), Parallel Distributed Processing, Vol. 1, Cambridge, MA: MIT Press.

Rumelhart, D. E. and McClelland, J. L. (1986) PDP models and general issues in cognitive science. In Rumelhart and McClelland (Eds.), Parallel Distributed Processing, Vol. 1, Cambridge, MA: MIT Press.

Shank, R. and Abelson, R. P. (1977). Scripts plans goals and understanding, Hillsdale, NJ: LEA Press.

Searle, J. R. (1969). Speech acts, Cambridge: Cambridge University Press.

Searle, J. R. (1981). Minds brains and programs. The Behavioral and Brain Sciences, 3, 417-424.

Sidman, M. (1986).Functional analysis of emergent verbal classes. In Thompson and Zeiler (Eds.), Analysis and Integration of Behavioral Units, New Jersey: LEA Press.

Simon, H. J. (1969). The Sciences of the Artificial, Compton lectures, Cambridge, MA: MIT Press.

Simon, H. J. (1976). Administrative Behavior, (3rd edition), New York: Free Press.

Skinner, B. F. (1938). Behavior of Organisms, New York: Appleton-Century.

Skinner, F. B. (1957). Verbal Behavior, New York: Appleton-Century.

Winogard, T. and Flores, C. F. (1986). Understanding computers and cognition, New York: Addison Wesley.

Wittgenstein, L. (1963). Philosophical Investigations (translated by G. E. Anscombe), Oxford: Blackwell.

AN INTELLIGENT SPECIFICATION ASSISTANT

Sheryl Duggins
Software Engineering Research Center
Computer and Information Sciences Department
University of Florida
Gainesville, Fl. 32611

ABSTRACT

Software development is a knowledge-intensive activity that can be simplified by utilizing artificial intelligence techniques to manage the knowledge and facilitate software development and maintenance. The Intelligent Programming Environment project involves the creation of an artificial intelligence based software development environment. This paper presents an overview of the Intelligent Programming Environment and examines how one component of that environment, the Specification Assistant, simplifies the task of modeling requirements specifications.

1. INTRODUCTION

Software development is a knowledge-intensive activity that can be simplified by utilizing artificial intelligence (AI) techniques to manage the knowledge and facilitate software development and maintenance. This simplification will directly result in increased programmer productivity and decreased maintenance costs (Stamps 1987).

The degree of improvement in software development productivity resulting from the utilization of knowledge maintenance tools depends more on the integration of the various tools than on the selection of the tools themselves (Voelcker 1988). Thus there is a need for an integrated knowledge-based software engineering methodology (Balzer, Cheatham, and Green 1983; Barstow 1987; Dankel et al. 1987; Harandi 1986). This paper presents an overview of the development of a programming environment that embodies such a methodology, and examines one component of that environment, the Specification Assistant, in more detail.

2. PROJECT OVERVIEW

The Intelligent Programming Environment (IPE) project being developed at the University of Florida (Dankel et al. 1987) involves the creation of an environment to assist analysts in the development of software. This environment provides assistance during all phases of software development, from the initial gathering of requirements to the maintenance of the resulting code. At the heart of this environment are a number of intelligent assistants, each of which is knowledgeable about some specific stage of software development. These assistants provide expertise to the analyst and direct the development process.

The IPE is actually an extensive application of artificial intelligence techniques to software engineering. It consists of a number of intelligent assistants (expert systems) that interact with an analyst, with each other, and with a Specific Program Description Data Base. The result of this collaboration is a new approach to software development that we believe may effectively replace the traditional Waterfall Model of software development. In our model the intelligent assistants monitor all tasks being performed, maintain a record of all tasks, verify the correctness and completeness of all actions, identify potential trouble spots to prevent future problems from occurring, and offer expert advice concerning the particular phases of development.

Our current emphasis in the development of this environment is on the design of those subsystems dealing with the early stages of program development: requirements acquisition and specification. We introduced an Object-Oriented Box Structured Analysis Methodology (OOBSAM) as our methodological approach to software development (Dankel 1988). OOBSAM combines aspects of Structured Analysis (SA) (Duggins 1988; Marca and McGowan 1988; Ross 1977; Ross 1985; Ross and Schoman 1977), Requirements Modeling Language (RML) (Borgida, Greenspan, and Myloupoulos 1985; Greenspan 1984; Greenspan, Myloupoulos, and Borgida 1982), and Box Structured Methodology (BSM) (Mills 1988; Mills, Linger, and Hevner 1987) into a unified approach which serves as the foundation of our programming environment.

The initial stage of program development in OOBSAM involves the structured decomposition of the problem domain and is guided by the

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

Requirements Assistant. The analyst interacts with this assistant via an intelligent, interactive version of SA. SA is a box-and-arrow language used for initial problem decomposition in any domain. It yields models that look similar to data-flow diagrams, but behave according to the emphasis placed on the specified constraints. The output of the Requirements Assistant is a low-level SA description of the problem domain in which all of the relevant terms have been identified and structurally organized.

This resulting SA model will then be transformed into an equivalent model in RML via interaction with the Specification Assistant. RML is an object-oriented requirements specification language that is compatible with SA. A model expressed in RML is a semantic model and consequently has significantly more detail than one expressed in SA due to the precision of RML. Thus, the transformation from a structural SA model to a semantic RML model is not direct. The RML model must contain all of the information embedded in the SA model plus additional semantic information. The precise nature of this extraneous information and how to effectively utilize this knowledge to aid the transformation process will be examined in Section 3 of this paper.

The RML model constitutes a complete description of the software and its environment. It specifies precisely what actions must be performed by the software but does not include any implementation details. This description is further refined in the IPE by design assistants that refine the RML model by specifying abstract and concrete data types. Finally, the detailed design is converted into a specific implementation in a particular programming language by the Coding Assistant. This assistant is knowledgeable of several programming languages and will be able to produce the final program in the language selected by the analyst.

This model of software development provides a unified approach towards development and maintenance of software. The various models being created and all corresponding development decisions are maintained within the data base. The availability of this information should significantly simplify the tasks of redesign and maintenance by providing a guide to the original development process.

Our current efforts on this project include the development of the SA prototype, the development of a preliminary RML prototype, the development of a rule-base for the Requirements Assistant, and a preliminary investigation of the process involved in the transformation from an SA model to a RML model. Section 3 of this paper explores work being done on this transformation system and Section 4 presents an example to illustrate this phase of development. Research contributions and concluding remarks are presented in Section 5.

3. THE SPECIFICATION ASSISTANT

Structural and Semantic Modeling

Those tasks involved in the early phases of the software development life cycle are the most knowledge-intensive and therefore, would benefit the greatest from an artificial intelligence based software methodology. However, they also happen to be the hardest to systematize due to the nature and abundance of the knowledge that must be captured and the fact that the required knowledge lies completely in the minds of the users and the analysts. The task of modeling requirements specifications involves extracting that knowledge and expressing it in the desired requirements specification language.

Regardless of the particular requirements specification language chosen, there are two conceptually different modeling tasks that must be addressed during this phase of software development (Greenspan 1984). These two different modeling tasks can be viewed as comprising two distinct levels inherent in modeling requirements specifications.

The first of these involves modeling done at a structural level in which the relevant terms in the application domain are identified and their relationships are specified. The second level involves creating a semantic model in which the information explicated at the first level is enhanced with additional semantic information.

Current modeling techniques exist that operate at one or the other of these levels. The typical scenario involves an analyst who first describes the requirements in a structural model, utilizing one modeling technique. To complete the requirements modeling, all of the information contained in the structural model must be expressed utilizing the semantic modeling technique. In essence, the analyst must respecify the information contained in the structural model and perform two separate decompositions. This not only wastes the time of the analyst but it also introduces the possibility of errors and inconsistencies in the semantic model due to the informal transformation between the two models.

The Transformation Methodology

What is needed is a methodology that provides for modeling at both levels. This can be done by having a unified underlying representation scheme and an implicit transformation methodology that:

* facilitates the level change
* automates as much of the transformation as possible
* directs the transfer with the aid of knowledgeable assistants.

This transformation should appear to the analyst as simply another step in the continuum of software development rather than as an abrupt

change in modeling strategies as was described in the previous scenario.

Within the IPE, the structural modeling level is facilitated by an interactive implementation of the SA box-and-arrow language. A number of intelligent assistants monitor all actions performed during the structural modeling phase. These assistants are knowledgeable about the SADT decomposition methodology and provide expert advice regarding that methodology. They also perform consistency checking to ensure that no errors exist within the model being created.

RML is used in the IPE to facilitate the semantic modeling phase of requirements modeling. As in the structural level, a number of intelligent assistants will monitor the actions being performed during this phase of development.

The Intelligent Specification Assistant controls the transformation between the structural SA model and the semantic RML model. RML was designed to be used with SA, and some of the SA concepts map directly into RML objects. However, the transformation between SA and RML is not direct. There is significantly more information in a RML model than in the corresponding SA model. Consequently, there will never be a fully automated transformation between these two levels because additional design information must be specified in RML. That is, the transformation will be guided by heuristics and will always involve a human "in the loop."

The Relationship Between SA and RML

We have identified the following four tasks that must be performed to obtain the successful completion of the Specification Assistant:

1. A thorough investigation of the two modeling languages (SA and RML) to discover precisely what can and cannot be expressed at each level.

2. The identification of a mapping function that will provide the underlying transformation for the maximal set to be automatically refined.

3. The identification of a complete set of extraneous information that will provide all of the information needed at the RML level that is not provided directly at the SA level.

4. The development of a facility to easily extract the set of extraneous information from the user to provide all of the information necessary to build the RML model.

The above tasks all touch upon various aspects concerning the relationship between SA and RML. Our preliminary investigation has provided us with the following insights:

* SA models consist of information that is expressed in the diagrams and in accompanying text:

SA = SA(Graph) + SA(Text)

* The information contained in SA accompanying text is of three types:

1. text that restates what is on the diagram (summary information)

2. text that includes information needed at the RML level (auxiliary information)

3. text that is superfluous from the perspective of the RML transformation (documentation information)

That is,

SA(Text) = SA(Text.Graph) + SA(Text.RML) + SA(Text.Sup)

* The information contained in a RML model contains all of the information in the associated SA diagram plus some extraneous information:

RML = SA(Graph) + Ext

* Some of the extraneous information can be derived from the accompanying SA text:

SA(Text.Graph) + SA(Text.RML) => Ext(SA)

* All of the information needed to define any RML model given the associated SA model can be obtained from the SA diagram, the SA accompanying text, the extraneous information derivable from SA, and a set of additional extraneous information:

RML = SA(Graph) + SA(Text.Graph) + SA(Text.RML) + Ext(SA) + Ext

What the above insights suggest is that a large portion of the information needed for the transformation from an SA model to the equivalent RML model either exists in or is derivable from the SA diagram and text. The remaining set of extraneous information can be further defined based on the investigation of what can be expressed in RML.

4. A CONCRETE EXAMPLE

To help us better understand the kinds of information that can be expressed in SA and RML, we have developed a sample problem that we are decomposing (Duggins 1989). The system being explored is an inventory system for a bookstore. To illustrate the differences between a SA model and a RML model, we present one component of the inventory system in SA and RML.

The diagram illustrates the decomposition of the Process Update subtask of the inventory system (see Figure 1). This activity describes how the inventory system responds when it receives a valid update command.

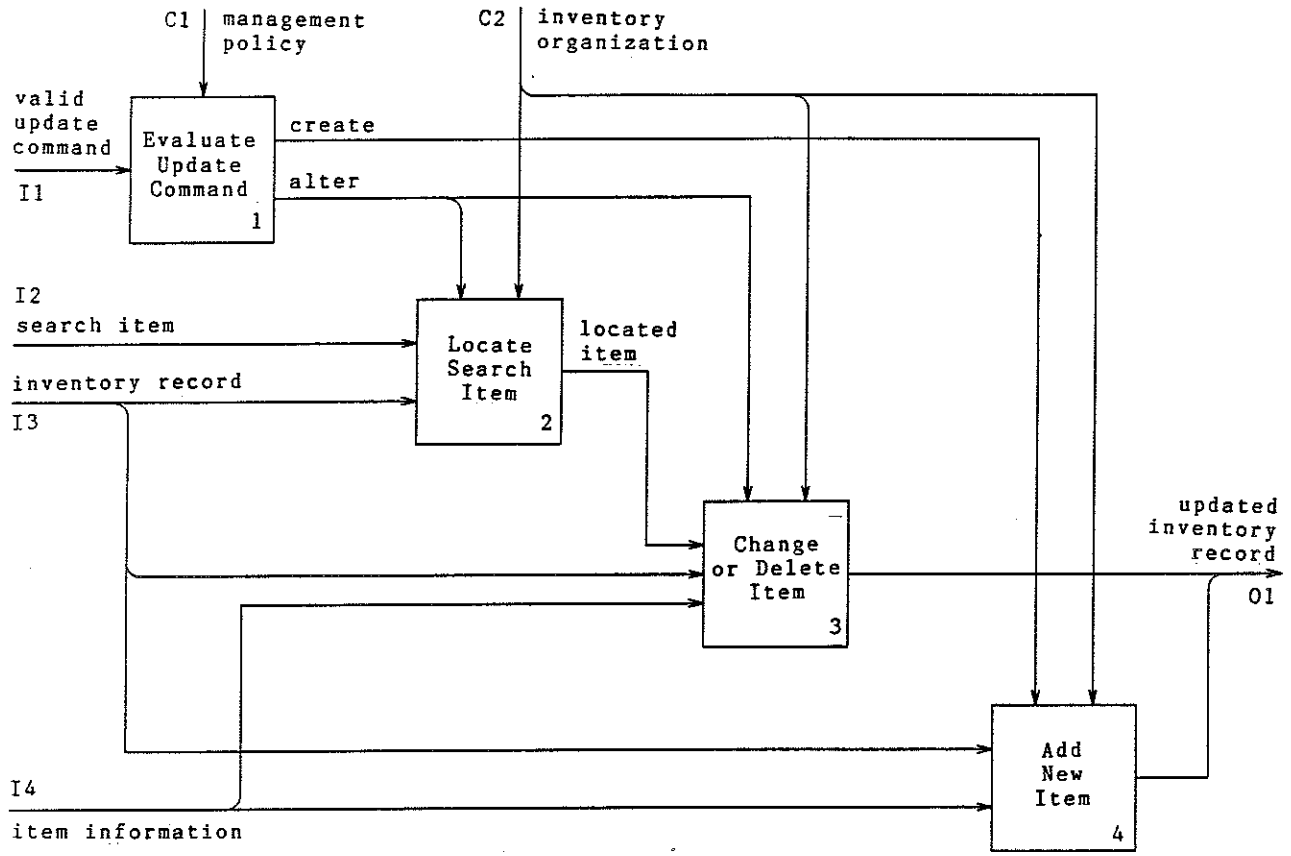


Figure 1. An SA decomposition of Process Update.

In addition to the valid update command, this activity accepts as input the inventory record which contains the recorded information about the current inventory, a search item that is used to locate the existing inventory item that is to be updated, and the item information to be added to the inventory record.

The update command either creates a new item or it alters some information about an existing item. Which of these two actions occur, and how the particular subtasks are defined, depends upon management policy and the inventory organization.

The above discussion would typically be included with the SA diagram as accompanying text. Utilizing the framework described in the previous section to categorize SA text, this text simply restates information being conveyed on the diagram.

Given the information contained in the diagram in Figure 1 and the accompanying text as presented above, the translation between SA and RML yields the preliminary RML model shown in Figure 2. The undefined fields in this model will have to be completed using a combination of information that can be derived from the diagram and text, and information that must be supplied

```
PROCESS_UPDATE in ACTIVITY_CLASS with

   input
      I1:VALID_UPDATE_COMMAND
      I2:SEARCH_ITEM
      I3:INVENTORY_RECORD
      I4:ITEM_INFORMATION
   control
      C1:MANAGEMENT_POLICY
      C2:INVENTORY_ORGANIZATION
   output
      O1:UPDATED_INVENTORY_RECORD
   parts
      A1:EVALUATE_UPDATE_COMMAND
      A2:LOCATE_SEARCH_ITEM
      A3:CHANGE_OR_DELETE_ITEM
      A4:ADD_NEW_ITEM
   precond
      ...
   postcond
      ...
   actcond
      ...
   stopcond
      ...
   constraints
      ...
```

Figure 2. A RML description of Process Update.

by the analyst. We are trying to minimize the additional information the analyst must supply, and maximize the amount of information that can be derived from the diagram and text.

5. CONCLUDING REMARKS

The Specification Assistant consists of a set of intelligent assistants that will facilitate the transformation from a structural SA model to an equivalent semantic RML model. They will automatically refine as much as possible from the information expressed in SA. They will direct and monitor all phases of the transformation to ensure consistency between the modeling levels.

The analyst utilizing our Intelligent Programming Environment for modeling requirements specifications will benefit in the following ways:

1. By interacting with the intelligent interactive SA system, the analyst will be able to abstractly identify the objects and activities that comprise the world being modeled.

2. Through a series of top-down refinement techniques, the analyst is able to introduce more and more detail while the consistency between levels is automatically checked and the record-keeping tasks are automatically performed by intelligent assistants.

3. Since the transformation from structural modeling to conceptual modeling will be incorporated into the system, the analyst will view this task as just another refinement level and will not have to expend additional effort restating information identified earlier in the methodology.

We believe the Intelligent Programming Environment described in this paper presents an alternative approach to the traditional Waterfall Model of software development. We further believe that the research being done on the Intelligent Specification Assistant will simplify the task of requirements specification modeling, and consequently, will improve programming productivity and decrease the cost of developing software.

REFERENCES

Balzer, R., Cheatham, T. E. Jr. and Green, C. 1983. "Software Technology in the 1990's: Using a New Paradigm." Computer, 16, no. 11: 39-45.

Barstow, D. 1987. "Artificial Intelligence and Software Engineering." In Proceedings of the International Conference on Software Engineering. IEEE Computer Society, Long Beach, Ca., 200-211.

Borgida, A., Greenspan, S. J. and Myloupoulos, J. 1985. "Knowledge Representation as the Basis for Requirements Specifications." Computer, 18, no. 4: 82-91.

Dankel, D. 1988. "Intelligent Programming Environment: Overview of the Current Design." In Proceedings of the Workshop on Automating Software Design (AAAI, St. Paul, Mn., Aug. 25). AAAI, Menlo Park, Ca., 16-20.

Dankel, D., Duggins, S., Hutches, D. and Sarver, T. 1987. "Intelligent Programming Environment: Overview of the Current Design, Fall 1987." Technical Report SERC-TR-8-F. Software Engineering Research Center, University of Florida, Gainesville, Fl.

Duggins, S. 1988. "SADT--An Overview of the Methodology." Technical Report SERC-TR-21-F. Software Engineering Research Center, University of Florida, Gainesville, Fl.

Duggins, S. 1989. "The Bookstore Inventory System: A Detailed SA Example." Unpublished manuscript. Software Engineering Research Center, University of Florida, Gainesville, Fl.

Greenspan, S. J. 1984. "Requirements Modeling: A Knowledge Representation Approach to Software Requirements Definition." Technical Report CSRG-155. Computer Systems Research Group, University of Toronto, Toronto.

Greenspan, S. J., Myloupoulos, J. and Borgida, A. 1982. "Capturing More World Knowledge in the Requirements Specification." In Proceedings of the Sixth International Conference on Software Engineering. IEEE-CS, Los Alamitos, Ca., 225-234.

Harandi, M. T. 1986. "Applying Knowledge-Based Techniques to Software Development." Perspectives in Computing, 6, no. 1: 14-21.

Marca, D. A. and McGowan, C. L. 1988. SADT: Structured Analysis and Design Technique. McGraw-Hill, New York.

Mills, H. 1988. "Stepwise Refinement and Verification in Box-Structured Systems." Computer, 21, no. 6: 23-36.

Mills, H. D., Linger, R. C. and Hevner, A. R. 1987. "Box Structured Information Systems." IBM Systems Journal, 26, no. 4: 395-413.

Ross, D. T. 1977. "Structured Analysis (SA): A Language for Communicating Ideas." IEEE Transactions on Software Engineering, SE-3, no. 1: 16-34.

Ross, D. T. 1985. "Applications and Extensions of SADT." Computer, 18, no. 4: 25-34.

Ross, D. T. and Schoman, K. E. Jr. 1977. "Structured Analysis for Requirements Definition." IEEE Transactions on Software Engineering, SE-3, no. 1: 6-15.

Stamps, D. 1987. "CASE: Cranking Out Productivity." Datamation, 33, no. 13: 55-58.

Voelcker, J. 1988. "Automating Software: Proceed With Caution." IEEE Spectrum, 26, no. 7: 25-27.

DYNAMICAL SYSTEMS PHASE PLANE MAP APPROACH TO REPRESENTING A NEURAL NETWORK'S LEARNING HISTORY

Erich G. Nold, Greg L. Heileman and Francis J.Gerrity

Martin Marietta Electronic System Division
P.O. Box 5837
Orlando, Florida 32855

Department of Electrical Engineering
University of Central Florida
Orlando, Florida 32816

ABSTRACT

In this paper we present phase space (or state space) 2-D maps of the learning process which dynamically proceeds within the space of weights of an adaptive weight neural network. The motivation for this research is to present a qualitative, and quantitatively useful representation of the learning system's time evolution by applying graphical and analytical techniques used extensively in nonlinear dynamical systems theory. The phase plane maps indicate: similarity of trajectories of specific learning algorithms (independent of initial conditions), changing cyclical behavior (orbits) not apparent in simply the time or frequency domains, a useful relation to several dynamical systems analytical approaches.

I. INTRODUCTION

The design of large neural network connection topologies and associated learning algorithms which can be trained in a reasonable amount of time is difficult. In fact, a generalized learning problem of memorizing given input/output pairs in a non-recurrent (feed forward) network has been proven to be NP-complete (Judd 1987). This essentially means that the time it takes to train increasingly larger networks of this type grows exponentially. The theorem holds for all possible learning algorithms applied to training this general type of mapping on this specific architecture.

This theorem encourages experimental research in at least three areas.

1) Try to define learning algorithms which, when applied to a limited domain of input problems exhibiting certain types of regularity, will learn, or converge, in polynomial time. Grossberg (1976) has proven a related theorem concerning the usefulness of regular, as opposed to arbitrary input domains.
2) Continue to research various types feedback network topologies.
3) Continue to research probabilistic algorithms.

Much of dynamical systems theory (Rosen 1970, Hirsch 1987) and experimental method (Moon 1987) can be directly applied to studying the long term learning behaviors which occur during the training of neural networks. We hope to show that the use of phase maps, and other dynamics analysis tools provide a necessary contribution to the evolving theoretical and experimental framework of neural network architecture and learning algorithm development.

Phase maps have two important attributes. First, they show a qualitative graphical synopsis of a complex temporal trajectory. In our application to neural network learning, these trajectories can be used to represent the pattern by pattern error minimization route, or the magnitude of a weight, or group of weights, as they fluctuate during the training process. In short, the phase map gives a snap shot of the learning history, an observable of the dynamics involved. Secondly, theory concerning stability of dynamical systems can be utilized by applying analysis methods relating phase map representations to network learning performance.

II. PLOT RESULTS

This section defines a single feed forward network and three different learning algorithms used to create the reported phase plane maps.

The "XOR problem" (see Rumelhart and McClelland,1986) was digitally simulated on three identical feed forward perceptron structures which each utilized a different learning algorithm. The network was simply two layers, three nodes, and nine variable weights, of which three were fixed input threshold adjustment weights. The network training can be thought of as a learning system consisting of an input ensemble or forcing function:(four pattern vectors) and their accompanying desired outputs which are simultaneously (same iteration) forced as inputs to the "output" node. The transfer function (neural net) receives these related forcing functions at both the input and output. The actual system out-put is forced to within a desired convergence tolerance. This tolerance was chosen to be ± 0.3, or 30% of the total swing between two classes being trained at the output, 0 or 1.

Three variations of the steepest gradient descent algorithm were plotted.

1) Approximation to true gradient descent, adjusting the weights only after a full cycle of all four patterns were presented, gradients being averaged over this full cycle. It is an approximation due to the finite learning rate or step size of 2 used. See Figure 1.
2) Back propagation (Rumelhart and McClelland 1986) where the weights are adjusted after each pattern

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

c. 1989 FLAIRS 0-9620-1731-0/89/0400-0055

presentation. See Figure 2.
3) Back propagation with holding of the same input pattern until the 0.3 convergence tolerance is met for that single pattern. This can be thought of as an attention feedback mechanism. The algorithm holds attention on one pattern by feeding back to the input, to control the sequence of patterns presented. See Figure 3.

At each discrete iteration, four events are recorded in the phase plane maps. See Figures 1b, 2b, and 3b.
1) A single pattern pair is presented at the input and output.
2) The signed error for that pair (desired minus actual) is plotted against the abscissa.
3) The difference between the present and the immediately preceding pattern's error (present error minus previous error) is plotted against the ordinate.

difficult, and we are still investigating this (see Rosen 1970). The first two algorithms were presented periodic input cycles which accounts for much of the orderly rotation of orbits during convergence seen in the phase maps. The third holding algorithm would have the identical periodic inputs if it did not feedback to hold on an input pattern.

Many more phase maps were plotted of these three algorithms on this three node network. All phase maps for a specific algorithm had strikingly similar orbital patterns, regardless of initial weight settings or the convergence time of the network. This phase map similarity was also true for learning rates of 0.5, 2, and 4. These orbital similarities suggest investigating a bifurcation route to possible chaotic behavior (Oblow 1988).

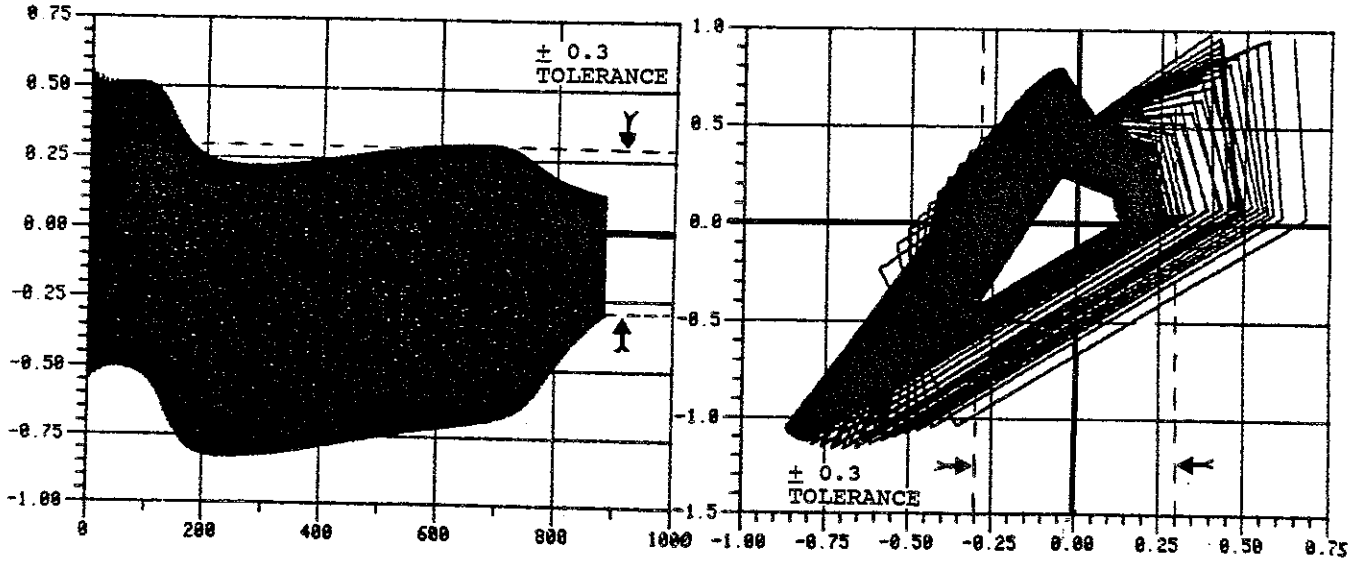


Fig. 1 (a) Successive pattern error (y axis) vs. iteration for a discrete approximation to gradient descent.

(b) Phase map, approximated derivative of successive errors vs. instantaneous error.

4) A line is drawn to connect the sequence of phase plane points to help show the time evolution trajectories.

These phase maps portray differing error surfaces or "energy surfaces" (Hopfield 1982) as basins of attraction which each have an attracting equilibrium point at the origin, the point of zero error. Figure 3b however, shows two peripheral orbits, centered at the plus and minus 0.3 convergence tolerance boundaries. This is due to the holding of single patterns. The global or main orbit is centered at the origin as in the other figures 1b and 2b. The time domain plots associated with each phase map are included in Figures 1a,2a, and 3a. Time is incremented with each pattern presentation. Each increment is associated with a point on the plots reported. Note the vertical axis of the time plots is identical to the horizontal axis of the phase maps.

The gradient descent algorithms plotted here are non-autonomous, meaning they have an explicit dependence on time due to their input pattern sequencing. This makes interpretation of phase maps much more

It has been shown by Oblow (1988) that stable and chaotic phase map trajectories in the case of the quadratic map, are bounded. He also shows these boundaries to be independent of initial conditions, and constant for specific regions of a system parameter similar to the learning rate parameter used in the gradient descent algorithms. Oblow's results may be helpful in interpreting the invariance in orbital patterns we have reported.

Similar phase maps for single, as well as multiple groups of weights have been plotted and will be reported later. Figures 4a and b show the time and phase plots for the value of a single weight as it adjusts towards its final value in a back propagation trained network. Grossberg (1988) predicted this type of erratic phase trajectory in weight space for back propagation learning.

III. PHASE PLANE MAPS FOR LONG TERM TRAJECTORY ANALYSIS

Phase plane maps are qualitative graphical tools used to observe the varying orbital trajectories inherent in most complex dynamical systems (Moon 1987). Most adaptive neural networks today, are designed to converge towards a point or so called equilibrium attractor (Hirsch 1987). These points are the origins, or zero error points, in figures 1b, 2b, and 3b.

The phase plane map, as applied here, is a discrete time plot of a value of interest versus its derivative with respect to time. A helpful analogy for interpreting these phase plane maps is to picture the x,y axis as position and velocity, respectively. A point on the map indicates how fast the value is changing (y coordinate) and what is its instantaneous value (x coordinate). Relating this to a supervised training algorithm, the value of interest may be the error for a particular pattern presented to the network.

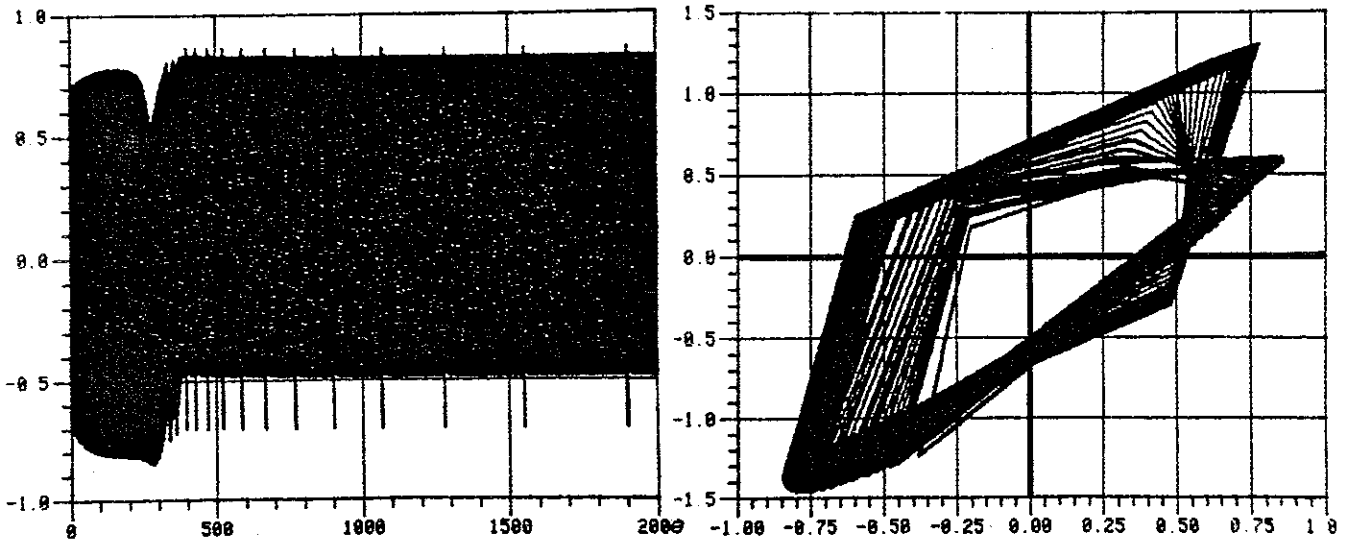


Fig. 2 (a) Error vs. iteration for back propagation using learning rate of 4, run was terminated after 2000 iterations.

(b) Phase map, converging towards origin till iteration 400, then bouncing between two orbits for iterations 400 - 2000.

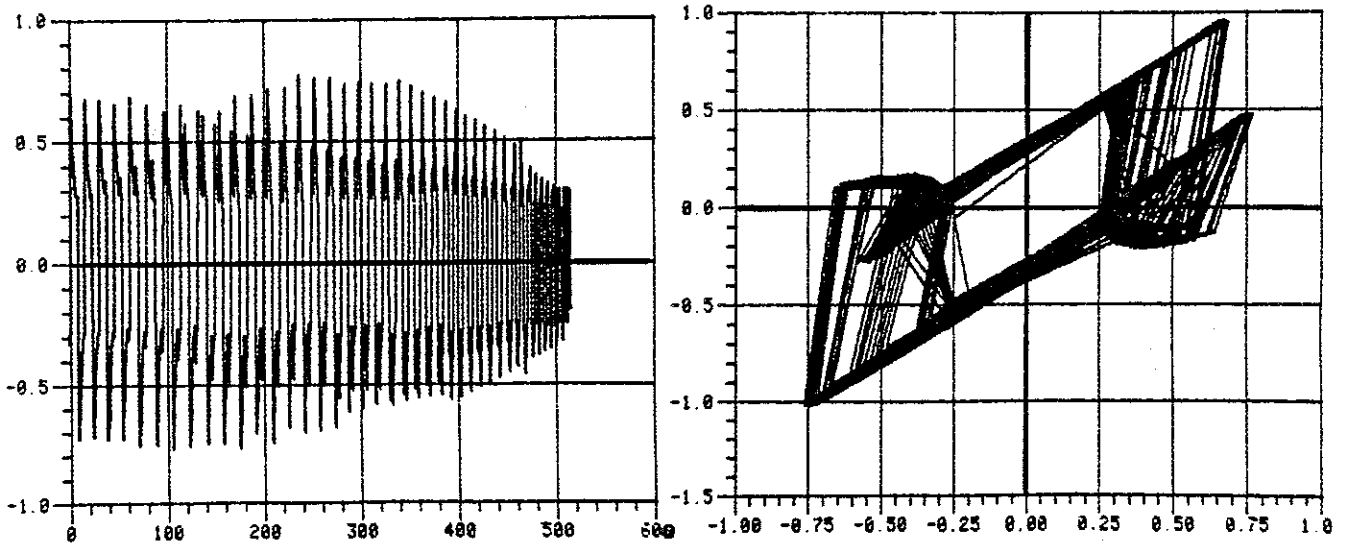


Fig. 3 (a) Error vs. iteration for back propagation which holds each input pattern to the 0.3 convergence tolerance, then changes patterns and holds again.

(b) Phase map with three related orbits.
Two at the 0.3 tolerance thresholds, and one at the origin for all four patterns.

The object of the learning system is to reduce the error (and in turn, the error "velocity"), which appears to spiral in towards the origin during convergence, as in figures 1-3b. It is interesting to note that the convergence tolerances, such as ± 0.3 used below, would allow a cyclical or limit cycle solution within the tolerance bounds.

IV. RELATED DYNAMICAL SYSTEMS ANALYSIS

The steepest gradient descent method is Lyapunov stable (error is monotonically decreasing) when the learning rate approaches zero. This is the first algorithm described above and is plotted in figure 1, using a learning rate of 2. Note the monotonic decreasing behavior of the error versus time plot in figure 1a. In order to speed up convergence, many variations of this gradient descent method have been developed (Heileman et. al. 1989, Hush and Salas 1988, Jacobs 1988).

When changing the algorithm, Lyapunov stability no longer holds, as can be seen in the non-monotonic, erratic behavior of figures 2a and 3a. The hopefully damped erratic behavior which appears in these modified, faster algorithms, exhibit sensitivity to initial conditions (w.r.t. convergence times, but not w.r.t. orbital shapes), and lend themselves to chaotic dynamical analysis.

Several analysis methods for classifying a broad range of chaotic trajectories are mentioned below (see Moon 1987). Note that chaotic behavior can be transient, or damped over time, a necessary attribute for convergence. Additionally, since networks are only trained to finite tolerances (say 30% allowable error), a constant chaotic orbit which would remain bounded within this error tolerance would be acceptable. In fact, if neural nets are true to their charter of attempting to mimic real brains, experimentally measurable chaotic dynamics will be a necessary part of their repertoire (Babloyntz 1985, Guevara 1983).

Four analysis methods related to phase maps are listed.

1) A spectrum of Lyapunov exponents can be calculated from the time series contained in the above reported phase maps (Wolf 1985). This spectrum characterizes the dynamical system with respect to stability or degrees of chaotic behavior.

2) Types of spectral broadening in the frequency domain indicate chaos (Moon 1987).

3) Invariant probability distributions of attractors in phase space has been studied by Hsu (1987).

4) Fractal dimension of attractors, and fractal basin boundaries (Moon 1987) may prove useful in classifying phase space trajectories as well as defining regions of initial conditions (initial weight settings) for more rapid training convergence. Attractor basin study may also help define system input regularities, which can be exploited for application specific tasks.

V. CONCLUSION

This work is aimed at developing useful tools for continued design of adaptive neural networks. It is the author's opinion that many of the large supervised adaptive neural networks which can be trained in real time (or some reasonable amount of time) will necessarily exhibit forms of erratic behavior which can quantifiably be understood using methods developed to study deterministic dynamics, including chaos. The application of the design tools described above will help qualify and quantify the many types of erratic behaviors noted in neural network learning processes.

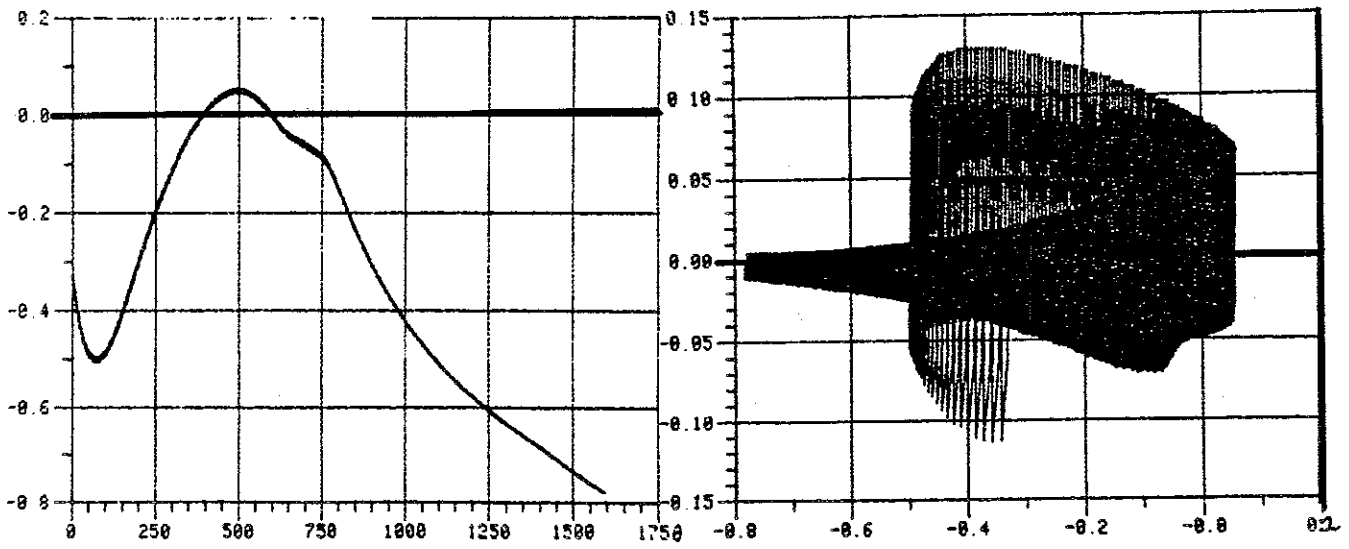


Fig. 4 (a) Single back propagation weight value vs. iteration. Decreasing oscillation of weight only apparent in phase map.

(b) Phase map of approximated weight value derivative vs. the instantaneous weight value. Note damped oscillation.

References

Babloyntz, A. 1985 "Evidence of Chaotic Dynamics of Brain Activity During the Sleep Cycle", Physics Letters, Vol. III, Sept. 2, pg. 152.

Grossberg, S. 1976. "Adaptive pattern classification and universal recoding, I: Parallel development and coding of neural feature detectors." Biol. Cyber. 23, 121-134.

Grossberg, S. 1988. "Nonlinear Neural Networks: Principles, Mechanisms, and Architectures." Neural Networks, Vol. 1, No. 1. Section L, pg. 44

Guevera, M. R., Glass, L., Mackey, M. C., Shrier, A. 1983. "Chaos in Neurobiology." IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-13, No. 5, 790.

Heileman G. L., Georgiopoulos M., Papadourakis, G. 1989. "Comparison of Learning Algorithms for Multi-Layer Neural Networks." 2nd annual FLAIRS.

Hirsch, M. 1987. "Convergence in Neural Networks." IEEE International Conference on Neural Networks. Vol II. pp. 115-125

Hopfield, J. J. 1982. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences, 79, 2554-2558

Hsu C. S. 1987. Cell to Cell Mapping, Springer-Verlag.

Hush, D. R., Salas, J. M. 1988. "Improving the Learning Rate of Back-Propagation with the Gradient Reuse Algorithm." IEEE International Conference on Neural Networks. pp.441-447.

Jacobs, R. A. 1988. "Increased Rates of Convergence Through Learning Rate Adaptation." Neural Networks, Vol. 1, 295

Judd, S. 1987 Learning in Networks is Hard, IEEE Conference on Neural Networks, Vol. II,685.

Moon, F. 1987. "Chaotic Vibrations: An Introduction for Applied Scientists and Engineers." Wiley-Interscience.

Oblow, E. M. 1988. "Supertracks, Supertrack Functions and Chaos in the Quadratic Map." Physics Letters A, Vol. 128, no. 8, 18 March.

Rosen R. 1970. Dynamical System Theory in Biology. Wiley-Interscience.

Rumelhart D. E., McClelland J. L. 1986. Parallel Distributed Processing. Vol. 1: Foundations. MIT Press. Chapter 8, pg. 332, Figure 4.

Wolf, A., Swift, J., Swinney, H., and Vasano, J. 1985. "Determining Lyapunov Exponents from a Time Series", Physica 16D, 285-317.

FPROLOG: A PROLOG INTERPRETER IMPLEMENTED IN FORTH

Johnny Y. Tang
Computer Science Department
Florida Institute of Technology

Sandra E. Cheung
Department of Computer and Information Sciences
University of Florida

1 Abstract

This paper describes the implementation of a simple Prolog interpreter in FORTH. This implementation was written in TTFORTH [3] which runs on an ATARI 1040 ST. In conventional Prolog, all clauses are on the same hierarchical level. To match a goal, the interpreter needs to potentially examine every clause in the knowledge base. This is a simple approach, but also inefficient. Due to the advanced features of FORTH the knowledge base of Prolog is modularized somewhat as is done in object-oriented programming; this makes Prolog's searching algorithm more efficient. FPROLOG uses Forth vocabularies to modularize its knowledge base, and thanks to the extensibility of Forth, the user can define his own local vocabularies to solve the problem of having metawords mixed with the knowledge base. The searching process is the core of a Prolog system. In FPROLOG, a depth first search is used for the search process. The search involves two stacks, the GOAL-STACK, containing the goals to be satisfied, and the BACKPOINT-STACK, which holds the backtracking points and keeps a trail of clauses used to satisfy a goal.

KEYWORDS: FORTH, Prolog, FPROLOG, knowledge base, interpreter, facts, rules, built-in functions, word, metaword.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

2 History and Motivation

2.1 About FORTH

FORTH was developed by Charles Moore in the early 1960s. FORTH [1,2] is a collection of words that are invoked by their name. It is also a high level language; it has a powerful set of primitive commands and provides a mechanism by which one can define new primitives. The structured process of building definitions upon previous definitions is equivalent to high level coding. At the same time, FORTH is also an assembly language; FORTH words may be defined directly in assembler mnemonics, using FORTH's assembler. FORTH is an operating system; it does everything a conventional operating system does, including interpretation, compilation, assembling, virtual memory handling, I/O, text editing, etc. The main advantage of FORTH is that it is fast, high level FORTH executes faster than other high level languages. Without a traditional operating system, FORTH eliminates redundancy and needless run-time error checking. FORTH machines are also much faster than other CPUs.

2.2 About Prolog

Prolog, which stands for PROgramming LOGic, is a simple yet powerful language. It has been used for a wide spectrum of applications, including natural language processing, expert systems, and problem solving. In Prolog, the programmer only needs to specify what needs to be done, without bothering with how this should be achieved. FPROLOG, the Prolog programming language implemented here, is based on the Prolog standard syntax as described in [6].

A Prolog system contains a knowledge base and an interpreter. The knowledge base consists of a collection of clauses which are either facts or rules. To execute a Prolog program, a question is typed after a Prolog's prompt. Questions are also called goals or queries. In Prolog, there are two types of questions. The first type is called a *simple query*, which is a direct request for confirmation that a fact is in the knowledge base. The Prolog interpreter responds with either *'yes'* or *'no'* to the simple queries[5]. The other type of question is a simple query with variables. The interpreter then tries to find corresponding bindings for these variables. This process is called unification. An inference engine, embedded in the Prolog's interpreter, searches the knowledge base for responses to questions. So, the structure of the knowledge base decides how the inference engine works and affects the efficiency of a Prolog.

3 Knowledge Base Design

3.1 A Modularized Knowledge Base Structure

In conventional Prolog, the knowledge base is not organized into groups of related clauses. Instead, all clauses are on the same hierarchical level. When the interpreter attempts to match a goal with a clause in the knowledge base, it has to potentially examine every clause in the entire knowledge base. The major drawback to the conventional knowledge base is that even though this kind of structure makes searching simple, it is also inefficient. FPROLOG provides a modularized knowledge base. The clauses in the knowledge base are grouped according to their predicate names.

3.2 TTFORTH Environment

In TTForth, each word consists six fields: 1) the General Purpose field, which is reserved for use by the programmer for any desired purpose; 2) the Token field, which holds the token number unique to that FORTH word; 3) the Link field, which holds the address of name field of the previously defined word (this provides a thread for FORTH when searching is necessary); 4) the Name field, which contains the name of the word; 5) the Code field, which points to the run-time routine for this type of word and 6) the Parameter field, which contains the token numbers or code necessary to perform the run-time action for this word.

In TTForth, words are partitioned into various groups which are called vocabularies. The words in a vocabulary are linked together as a linked list. In TTForth, as with other Forths, each vocabulary, itself, is a Forth word. All vocabularies are linked to a special group of words called the metawords.

The Parameter field of each vocabulary consists of three fields: 1) SLF - the search link field; 2) VLF - the vocabulary link field and 3) CLF - the chronological link field. The SLF points to the the search link field of the next vocabulary to be searched. If the content of SLF is zero, then this vocabulary is the last vocabulary in the search order. The VLF points to the latest defined word in that vocabulary. After a vocabulary has been created, its VLF has the address of the name field of the last word in the metawords. The CLF points to the chronological link field of the vocabulary defined just prior to the present vocabulary. If a vocabulary is the first one created in the environment, then the content of its CLF is zero [3].

The metawords are always the last words examined during any vocabulary search. They do not, however, constitute a vocabulary by themselves. Forth provides words to control the dictionary search path. The word ONLY puts only one vocabulary in the search order. The word ALSO adds a vocabulary to the search order.

FPROLOG utilizes Forth vocabularies to modularize its knowledge base. But this is not good enough. The problem is that the knowledge base is mixed with the metawords. To solve this problem, local vocabularies are needed. The current version of TTForth does not have local vocabularies. Fortunately, the extensibility of Forth allows the user to define local vocabularies. A local vocabulary is similar to an ordinary vocabulary except for the following: 1) when a local vocabulary has just been created, the VLF contains the address of the name field of the first word in the metawords, and 2) a local vocabulary is not in the metawords. It is linked with the words within its parent vocabulary.

4 Data Representation

FPROLOG's knowledge base contains three parts: facts, rules, and built-in functions. Rules and built-in functions are grouped in two local vocabularies. Facts are partitioned into sets of local vocabularies. All facts with the same predicate name are in the same local vocabulary.

4.1 Representation of Facts

FPROLOG uses local vocabularies to contain facts, and uses Forth's word structure to represent these facts. In general, the predicate name of a fact is the name of a local vocabulary. The number of facts in that local vocabulary is contained in an additional field at the end of the local vocabulary structure, and is used to generate the name of a Forth word for each fact. The argument of a fact is represented with a pair of fields: the ARGUMENT field and the TYPE field. Another number, stored before these fields is the NUMBER OF ARGUMENTS field, which indicates how many arguments there are in this fact.

In FORTH, the Code field is used to point to the run-time routine for that word. For facts in FPROLOG, there is no corresponding run-time routine for the word that represents a fact. FPROLOG reserves this field, because it is working within the FORTH environment, and retains the same structure as FORTH's words.

4.2 Representation of Rules

In FPROLOG, rules are simply grouped under one local vocabulary RULES. Unlike normal Prolog, rules in FPROLOG are separated from facts. This is because FPROLOG always searches facts first when it evaluates a goal. It considers rules as facts that are conditionally true. To satisfy a goal, there must be a corresponding clause in a fact base. If the corresponding clause is a rule, the interpreter is going to evaluate the conditions of the rule, known as subgoals, instead of that goal. Satisfying these subgoals determines the truth of that goal. If the goal is finally satisfied, a fact must be matched with that goal or some subgoals must be satisfied.

Similar to facts, the local vocabulary RULES has an extra field at the end of its local vocabulary structure; this field holds the number of the rules in that knowledge base. Each rule is stored in the parameter field of a FORTH word.

In general, the head of a rule is represented in a fashion similar to the arguments of a fact except that there is a PREDICATE NAME field on top. This field holds the predicate name of the head of the rule. The predicate name is stored as a counted string. The representation for the body of a rule has a NUMBER-OF-SUBGOAL field first. That field holds the number of subgoals for that goal. After the NUMBER-OF-SUBGOAL field, there is a number of subgoals. The representation of each subgoal is exactly the same as the representation for the head of a rule.

4.3 Representation for Built-In Functions

The built-in functions in FPROLOG are implemented in Forth code. So, the representation for built-in functions is the same as for Forth words. A function is executed in the same way as other Forth words are executed.

5 FPROLOG Interpreter

FPROLOG is implemented as an interpretive language. The interpreter of FPROLOG contains a loop that gets input, processes the input and reports the results of the processing. The FPROLOG interpreter consists of three main processes: (1) a parsing process[7], (2) a search of the knowledge base, and (3) a manipulation of the knowledge base. The parser checks the syntax of all input, and determines whether the input is a fact, rule or goal. The searching process searches the knowledge base to match a given goal. The knowledge base manipulator inserts clauses into the knowledge base or deletes clauses from the knowledge base.

5.1 The Searching Process

The searching process is the core of a Prolog system. It searches through the knowledge base for a clause that matches the goal. If one such clause is found, then its subgoals in turn must be satisfied through the searching process. This searching process is known as *backward chaining*. A depth first search algorithm is used for the

search process in FPROLOG. It uses two stacks, the GOAL-STACK which contains the goals to be satisfied, and the BACKPOINT-STACK which holds the backtracking points and keeps a trail of clauses used to satisfy a goal. The searching process first checks the GOAL-STACK. If it is empty, no more goals remain to be proven and the search has succeeded. Otherwise, it pops the first goal from the GOAL-STACK and searches for it in the fact base. If it is not found, it continues the search in the rule base. If either a fact or a rule is found, a pointer to this clause is pushed onto the BACKPOINT-STACK along with the goal itself, and the subgoals of this clause are pushed on the GOAL-STACK. On the other hand, if the clause is not found, the search checks the BACKPOINT-STACK. If it is empty, the search fails; otherwise, the backtracking proceeds.

5.2 Implementation of the Search Process

After the parser recognizes that the current input is a goal, the interpreter sets the GOALSTACK to that input and calls the search process. The word SEARCH-GOAL consists of a simple loop that iterates (SEARCH-GOAL). (SEARCH-GOAL) does the work of the search and returns a success or a failure flag to SEARCH-GOAL. (SEARCH-GOAL) first checks the GOAL-STACK with EMPTY?. If the GOAL-STACK is not empty, it gets a goal from the GOAL-STACK with POP-GOAL. It then starts the search for the goal from the fact base with FIND-FACT?. The word FIND-FACT? takes the pointer which points to the current goal and tries to find a fact that matches that goal. If it succeeds, it returns the current goal pointer and the nfa of the fact; otherwise, it returns the goal pointer. The word FIND-RULE? takes the goal pointer and continues searching for the goal in the rule base. If a rule is found, (SEARCH-GOAL) calls the word FOUND-RULE-PROCESS which saves the goal pointer and the nfa of the rule on the BACKPOINT-STACK, and also adds the subgoals of the rule on the GOAL-STACK. If FIND-FACT? returns a false flag, BACKTRACK is called. It checks the BACKPOINT-STACK with EMPTY?. If the BACKPOINT-STACK is not empty, a goal pointer and the nfa of the clause are popped from the BACKPOINT-STACK. BACKTRACK then calls FIND-CLAUSE which continues the search for the goal starting after the clause that is pointed at by the nfa. If no match was found, SEARCH-GOAL fails. BACKTRACK returns a flag indicating either success or failure.

5.3 Unification

The implementation discussed above can not handle variables, and thus the system could only provide final answers like "yes" or "no" to a Prolog query.

To extend FPROLOG so that it can handle goals that have variables, the pattern matching function of the previous section has to be modified. It needs to take two parameters: formal and actual. It returns true if all the formal and actual parameters are compatible; otherwise unification fails.

5.4 Improving the Search Process

To include the unification in the search process, one thing must be considered. The bindings might be different for the same variable name. For example, if we consider the following two facts and one rule in a knowledge base.

weather(tuesday, fair)

weather(friday, fair)

color(sky, blue, Day) :- weather(Day, fair)

When a goal *color(sky, blue, When)* is typed, *When* is bound to *Day* and *Day* is bound to *tuesday*. The interpreter will display *When* to be *tuesday*. If an alternative answer is requested, *Day* will be bound to the new value *friday*. To be able to bind a new value, the search process requires *Day* to be uninstantiated. There are two techniques to keep track of different bindings to the same variable name. One is called *structure sharing* [4] which uses one GOAL-STACK with different local substitutions which give the new variables that replace original variables. Another approach is called *structure copying* which makes a copy of GOAL-STACK every time a variable is bound to a new value. The BACKPOINT-STACK includes a pointer to the current GOAL-STACK. CURRENT-GOAL-STACK is a variable which holds a pointer to the current GOAL-STACK to be satisfied.

6 Knowledge Base Manipulator

The knowledge base manipulator in FPROLOG is capable of doing the following functions:

(1) Insert a fact or rule into the knowledge base.

(2) Delete a fact or rule from the knowledge base.

(3) List all the clauses in the current knowledge base.

(4) Save the current knowledge base in a disk file.

(5) Load a knowledge base into memory.

For the first function, once the parser recognizes a fact or rule, the interpreter inserts that fact or rule into the knowledge base. The other functions are implemented as built-in functions.

7 Built-In Functions

There are a set of sophisticated built-in functions that are normally supported by Prolog systems. In FPROLOG, the built-in functions are implemented as FORTH words and are grouped together within the vocabulary "functions". FPROLOG is a simple subset of Prolog. The current version of FPROLOG supports the following functions: 1) clearbase: removes all clauses from the knowledge base; 2) halt: exits from the interpreter; 3) help: lists some built-in functions; 4) listing: lists all the clauses in the knowledge base; 5) retract: removes clause from the knowledge base; 6) file-list: saves clauses in the knowledge base and 7) consult: reads clauses into the base from a file.

8 Conclusion

FPROLOG, a Prolog interpreter, has been successfully implemented in TTFORTH and runs on the ATARI 1040 ST. The advantages of FORTH for knowledge system development are its extreme extensibility, its interactive nature and its interpretive environment. Vocabularies and local vocabularies were used to build a modularized knowledge base which makes the searching process in FPROLOG more efficient than the usual one in normal Prolog. In conclusion, FORTH is an extensive AI language. It makes the representation of facts and rules easier and more efficient.

As a complete computer programming language, FPROLOG needs to be expanded with more built-in functions and a more powerful parser. These could be considered in future work.

References

[1] Brodie, Leo. Starting FORTH. Prentice Hall, Inc., 1981.

[2] Kelly, Mahlon G. and Nicholas Spies. FORTH A Text and Reference. Prentice Hall, Inc., 1986.

[3] Lee, Hong-Soon. TTFORTH A Token Threaded Forth. Master's Thesis, Florida Institute of Technology, August 1986.

[4] Maier, David and David S. Warren. COMPUTING WITH LOGIC. The Benjamin/Cummings Publishing Company, Inc., 1988.

[5] Rogers, Jean B. A Prolog Primer. Addison-Wesley Publishing Company, 1986.

[6] Clocksin, W. F. and C. S. Mellish. Programming in Prolog. Springer-Verlag, 1984.

[7] Gustafson David A. et al. COMPILER CONSTRUCTION: Theory and Practice. SCIENCE RESEARCH ASSOCIATES, INC., 1979.

TOWARDS THE INTEGRATION OF KNOWLEDGE, REASONING, AND LEARNING

Uta M. Ziegler and Lois W. Hawkes
Department of Computer Science
206 Love Building
Florida State University
Tallahassee, Fl 32306

ABSTRACT

Many AI systems have not been as successful as expected. This paper suggests two major reasons for this: the separation of knowledge representation from the reasoning and learning processes; and the concentration on detailed, thorough analyses. Based on these points, principles of intelligence are developed, which address issues, such as integration of knowledge, reasoning, and learning, trial and error processing, contradictions, and mental models. Neural networks with appropriate structures and additional features are suggested as the processing paradigm which can overcome most of the suggested difficulties with conventional AI systems.

INTRODUCTION

Artificial intelligence and expert systems have developed rapidly and achieved impressive results in the last decade. However, they do lack the flexibility humans demonstrate in the application of knowledge as well as in the integration of new information. In order for AI programs to become truly intelligent and flexible, researchers must re-consider approaches to knowledge representation, reasoning, and learning. More powerful, general theories, principles, and methods must be developed to explain intelligent reasoning and learning in common domains. Focusing on very specialized and very inflexible methods, as is done in many expert systems, holds little promise for the future.

In this paper, we discuss some reasons for the inflexibility of AI systems and develop some general principles of intelligence. Then we suggest that neural networks are an approach which overcomes the discussed difficulties. The last section introduces some ideas about how the developed principles of intelligence can be incorporated into a neural network approach to AI.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

INFLEXIBILITY IN AI SYSTEMS

There are several reasons why AI systems are inflexible and unable to show really intelligent behavior. Some of those reasons are based on the presumed necessity of designing flawless basic operations for intelligent systems. Others are due to an incorrect understanding of the interplay of the various components of an intelligent system.

One of the difficulties with current AI systems is that they are designed primarily to use correct information. The validity of any *stored* knowledge is rarely questioned during processing. Therefore, input information which seems to contradict stored knowledge is ignored in most cases. Such contradictions are not used as an opportunity to improve the representation or the application conditions of the involved knowledge and methods. However, alert and intelligent humans often acquire, extend, change, or fine-tune their knowledge and reasoning methods as a consequence of a confrontation with contradictory information. Human processing methods are able to cope with the constant mis-information and contradictions inevitably encountered if knowledge is based on limited experiences. Humans re-interpret and modify stored experiences, if necessary, in order to solve given tasks and to incorporate new experiences into the system's knowledge. The basic operations of a conventional AI system do not enable the system to flexibly improve its knowledge and reasoning methods based on experiences.

The 'perfectness assumption' for stored system methods and knowledge in AI system leads to learning algorithms which over-emphasize detailed analyses and future application possibilities. Those algorithms require a complete analysis during learning, rather than a more efficient, situation-specific analysis (re-interpretation) during the application of the experiences. That is,

the resulting knowledge and methods cannot be adapted to specific situations as closely as desirable. However, since humans can re-interpret stored knowledge in new situations, they do not need to fully analyze the possible consequences of each experience. Rather, they need to structure and organize their experiences such that future re-interpretations are possible. Conventional learning methods favor a scrutinizing analysis of an experience and neglect the importance of organizing and structuring experiences to enable situation-specific analyses.

Besides these misconceptions about the necessity of perfectness of basic knowledge in intelligent systems, much of the inflexibility of AI programs is directly due to the separation of the knowledge of the system from the inference engine. Every piece of knowledge is handled in a similar manner. Knowledge is processed based on its form rather than based on its contents. The resulting reasoning methods and knowledge applications are stiff and superficial. In contrast, humans can apply knowledge and reasoning methods in a very opportunistic manner. Activated information influences reasoning processes, and reasoning influences the activation of information. Many AI systems are less flexible than expected, since they cannot incorporate this tight connection between knowledge representation and reasoning methods.

Moreover, in an AI system with separate inference engine, every piece of knowledge must fit consistently and completely into the program's knowledge in order to be usable. Since the program has only one method of reasoning, explicit details must be added in order to make all knowledge suitable for that one method. However, the amount of details needed to be prepared for every expected situation prohibits such explicit representations in humans. Rather, humans organize their knowledge such that details can be computed quickly whenever needed. In AI systems, flexible human reasoning mechanisms are replaced by heavily pre-analyzed knowledge.

The impact of the separation of knowledge and inference methods goes beyond shallow reasoning mechanisms. It promotes the misunderstanding that knowledge and reasoning are separate, only loosely related aspects of intelligence. However, there are mutual influences among reasoning, knowledge representation, and learning. Not only does the representation of knowledge facilitate or hamper some types of reasoning, but reasoning alters the representation of the knowledge. That is, it induces learning. For example, successful reasoning strengthens the used knowledge and connections, whereas unsuccessful reasoning weakens the involved knowledge. Moreover, learning of new information requires reasoning and knowledge. Reasoning processes check the consistency of the new information with (local) old knowledge. They also determine which part of the new information can be inferred from already available knowledge and which part is truly new and must be incorporated. Ignoring the dynamic interactions among knowledge representation, reasoning, and learning leads to stiff and inflexible AI systems.

PRINCIPLES OF INTELLIGENCE

New artificial intelligence paradigms must be developed to overcome the difficulties mentioned in the previous section. This section develops some principles which seem to be important in capturing true intelligence. They are based on the assumption that intelligence is a tool which offers humans the possibility to react to their environment as efficiently, swiftly, and correctly as possible, thus improving their chance of survival. They should be understood as goals and (moving) targets for future research in AI. The authors are well aware that these principles are not set in concrete and will undergo many changes before really intelligent systems are built. However, if they precipitate a discussion about the principles of natural and artificial intelligence, then they have served their purpose.

The Principle Of Integration

The basic tenet of intelligence integrates knowledge, reasoning, and learning.

As explained above, the separation of knowledge representation, reasoning, and learning leads to too much inflexibility and inefficiency in intelligent systems. Human reasoning processes are generally not working on top of the knowledge. Rather, they exhibit strong interactions and mutual influences with knowledge and learning processes. AI needs to *integrate* knowledge representation, reasoning, and learning in order to build flexible, opportunistic, and intelligent systems. Powerful human processing features, such as generalization, concept formation, default assignment, and reasoning in noisy and uncertain environments are only possible through close connections and influences among knowledge, reasoning, and learning.

This is the main principle. The following principles concentrate on some features such an integrated approach needs to satisfy.

The Principle Of Flexible Learning

The learning processes of an intelligent system must flexibly integrate experiences based on their relative importance for the overall state of the system.

Human brain processes evolved to learn as much important information from an experience as possible. At the time of the experience, however, it is not known which features of the situation will be of impor-

tance later. The only information available about what might be relevant in the future are the experiences which have been collected in the past. Therefore, learning is the integration of new experiences into the structures built through earlier experiences. However, the changes to the experience structures must reflect the relative importance of a new experience. The relative importance of an experience depends on several factors, such as the system's current state and goal, the type and the certainty of environmental feedback, and the impact of the current experience to the survival or other basic goals of the system. The learning processes of an intelligent system must flexibly adjust the degree and the amount of changes to the impact of the current experience on the overall state of the system.

The Principle Of Trial And Error Processing

An intelligent system must have the capability to move toward a solution in overtaxing situations, using the environment to obtain additional, critical information.

Humans base their reasoning and problem solving on the structures of their collected experiences. These structures are like a summary report of previous experiences rather than a set of if-then rules. Therefore, they often do not unambiguously identify a reaction to a new situation. For example, experiences can indicate several reactions which were successful in similar situations, none of which (and no combination of which) is clearly the correct reaction to the current situation. That means that not enough internal knowledge is available to make a decision. Therefore, one alternative is selected more or less at random. The chosen alternative and the environmental feedback add a critical experience to the knowledge needed to make a correct decision. An incorrect, initial choice is followed by increasingly accurate choices until a suitable alternative (or combination of alternatives) is selected. During trial and error processing, intelligent systems acquire from the environment - step by step - the information necessary to make a correct decision.

The Principle Of Consistency

Contradictions between stored knowledge and reality (which cannot be prevented by any intelligent system) are a system's only possibilities to improve its internal representation of reality.

Experiences are stored to adapt an individual better to his environment, that is, to enable him to react reasonably in many situations. Since no two situations are exactly alike, it is useless to retain an exact trace of every situation. Rather, characteristic features and relations of situations are learned with experience. At any time, an individual has a subjective representation of reality on which he bases his decisions. New experiences are integrated into this representation to assure appropriate contexts and good access. However, new experiences often uncover contradictions between the subjective and the true reality. In order to maintain a consistent subjective reality, some of its features must be changed to accommodate the new experiences. The discrepancies between the internal representation of reality and the sensory information about the environment are the crystalization points for improving the internal representation of the environment.

The Principle Of Learning How To Learn

In an intelligent system, learning how to learn is the consequence of building integrated and consistent structures of experiences.

This principle is closely related to the principle of integration. As described earlier, humans are able to build up relatively accurate representations of complex situations through maintaining a consistent internal representation. Since knowledge representation and reasoning methods are integrated, methods to use those complex representations are developed simultaneously with the representations. The methods identify important elements of a situation, determine their features, and check their relationships. These representations and methods are the knowledge humans employ when they explore new and similar fields. That is, humans need an internal representation of situations in order to learn about similar situations. Therefore, building up those representations and their reasoning methods is learning how to learn.

The capability to learn how to learn depends to a large extent on how well the consistency and integration of the internal representation is maintained. Insufficient integration and consistency leads to incomplete identifications of important connections and common features. Learning based on such representations is limited and leads to insufficient representations of the new situations. Note, that inefficient human learning capacities are often explained by a poor organization of knowledge.

The Principle Of Mental Models

An intelligent system must be able to use its experience structures independently from direct, environmental input.

As long as experience structures can be activated only through input from the environment, they can only determine suitable reactions for current situations. However, such a representation of reality can free the individual to some extent from the complete control of the environment. For example, it can interpret environmen-

tal signals with respect to their future consequences for the individual. That is, based on input signals, it can anticipate a future event, thus providing the individual with an extended reaction time. Experience structures can also be used to predict the consequences or features of (hypothetical) situations or to determine the results of (not performed) actions. The more intelligent a system, the more it can use its internal representation of reality to separate its actions from direct environmental influences.

THE NEW PROCESSING PARADIGM

The major principle of integrating knowledge representation and reasoning can be accomplished by neural networks.

The basis of these models is a network of simple elements, which are connected with different strengths and which have excitatory and inhibitory influences on each other. The connection strengths control the spread of activity through the network. They are the stored knowledge. A piece of information is represented as a pattern of activation over the elements of the network. Similar knowledge pieces have similar representation, that is, their patterns of activation overlap. Activation spreads through the network until an equilibrium state is reached. The spread of activation, that is, the reasoning, is controlled by the connection strengths between elements, that is, by the knowledge stored in the network. Reasoning and knowledge are inseparable in neural networks.

Since knowledge and reasoning are inseparable, reasoning processes seemingly applicable to various different concepts actually are tied to the same piece(s) of knowledge. This knowledge is a subcomponent common to the representation of all those concepts. Therefore, the more similar two concepts are, the more subcomponents they share. This strong integration of knowledge and reasoning also suggests that the representation of a concept contains traces or mirror-images of the representations of concepts with which it interacts.

In a neural network system, changing the connection strengths between elements changes the knowledge representation and the reasoning processes. That is, it embodies learning. Every act of learning influences the knowledge represented and the related reasoning processes. A connection's relative position in the network determines the degree of change during learning. That is, the representation of knowledge and reasoning processes influences the learning process. Therefore, reasoning, knowledge representation, and learning are inseparable in neural networks.

NEURAL NETWORK STRUCTURE

The previous section describes a processing paradigm that integrates knowledge representation, reasoning, and learning. This section describes some features of such a network which facilitate the incorporation of the other principles mentioned.

Simple Network Structure

Humans organize their experiences based on similarities and regularities detected in old and new experiences. Detectable regularities are simple relationships among familiar features. Those features can either be basic sensory inputs or earlier learned regularities. That is, humans detect regularities in their environment in successive levels of complexity.

We propose a flexible, layered network as a structure to facilitates the organization of experiences. Each layer of the structure consists of groups of strongly interconnected elements. The groups at each level are only loosely connected to one another. A group at one level is connected to several groups at the next lower level and can be connected to one or more groups at the next higher level. Each level can use the information from the adjacent levels to detect regularities and dependencies.

The proposed network levels represent different structural features of knowledge. They correspond to structural levels in the system's input. Each level itself can represent complex, layered knowledge. The knowledge at each level develops dynamically through experience. In order to improve the separation of the different levels, the type of connections among the layers is different from the type of connections among elements and groups at one level. Thus, a new network level is only employed if the detected regularities and features are sufficiently different from those at the current level.

The suggested organization can support general as well as specific reasoning processes. Lower levels of this structure capture characteristics, regularities, and features specific to the represented items. Reasoning processes at those levels are integrated with the regularities among basic features and are fast, accurate, and applicable only to the represented items. Higher levels in the structure capture relations and regularities among composed features and interactions among them. Reasoning processes at these levels are integrated with the higher-order regularities and structures of those levels. They are more general and weaker than lower level reasoning processes, because they are not directly connected to the basic features of the involved items. New experiences can be integrated at the appropriate level and thus are integrated with the reasoning processes of that level.

Meta Structure

The proposed layered network can store and process experiences. However, there are many different modes in which these experiences can be used, such as trial and error processing, mental modeling of situations and events, or recall. The different modes of processing can be achieved through modifying the basic operation processes of a network. Such modifications cannot be initiated by the network whose processing is to be modified. They must be initiated by other components, which store different experiences and serve different functions. Each such component is a layered network which learns to execute its function through experience. Thus, a complete network consists of loosely connected, integrated, and layered network components.

Flexible learning processes need such a component to determine the effects of an experience or event on the various goals of the system. This component can organize experiences about the connections among different system goals. In effect, it might develop and form with the development of system goals. Trial and error processes need a component which checks the progress made toward the goal over a sequence of basic steps. Mental model processes need several such components to select and generate the input situation, to process the output reaction, and to control the processing in the simulation component.

Since these components influence the mode of processing of other components rather than the processing results, it can be assumed that their effect must be spread diffusely to most elements of the receiving component. Special asymmetric elements can connect the output of the initiating component with many elements of the modified component. Instead of changing the activation value of an element, each connection causes the change of an internal element variable which influences the mode of processing.

Stable Connections

Learning is the re-interpretation and the integration of new experiences into the experience structure. Every act of learning changes some connection strengths in the network. Which connections must be changed and by how much depends on the new and the stored experiences. Since the changes are made to improve the internal representation, the amount of change to an involved connection should be reciprocal to its correctness. That is, correct connections should be changed as little as possible, whereas incorrect connections can be changed more. The correctness of a connection correlates with the stability of the connection strength.

Basing connection strength changes on the stability rather than the strength of a connection can explain some interesting aspects of trial and error learning. For example, in an unfamiliar situation, trial and error learning leads to permanent changes which make a repetition of the successful action more likely in similar situations. This happens, since the situation is unfamiliar and hardly any stable connections are involved in determining the reaction. Therefore, substantial changes can be made to associate the situation with the successful reaction. In a familiar situation, the connections which determine a suitable reaction are too stable to be changed, even if the associated reaction fails. Therefore, some less stable connections must be changed to integrate the experience. That is, trial and error processing does not change the general reaction to that situation, but ties the changes to some context specific features.

The stability of a connection also influences which connections should be changed or weakened in the case of a contradiction. Since stable connections are correct, they probably do not contribute to the contradiction and should therefore not be changed.

CONCLUSION

Interactions among knowledge representation, reasoning, and learning are fundamental to the development of intelligence. Imprecisions and contradictions in knowledge, reasoning, and learning are fundamental to the functioning of an intelligent system in its environment. Research in AI must address these two issues and develop processing paradigms, which integrate knowledge, reasoning, and learning and which incorporate the useful aspects of imprecisions and contradictions.

Bibliography

Hinton, G. E. and Anderson, J. A. 1981. Parallel Models of Associative Memory. Lawrence-Erlbaum Associates, Hillsdale, New Jersey

Kohonen, T. 1984. *Self-Organization and Associative Memory*, Springer-Verlag, New York

Kolodner, J. L. and Riesbeck, C. K. 1986. *Experience, Memory and Reasoning*, Lawrence-Erlbaum Associates, Hillsdale, New Jersey

McClelland, J.C. and Rumelhart. 1986. *Parallel Distributed Processing* Vol 1+2, MIT-Press, Cambridge, New Jersey

Minsky, M. 1986. *The Society of Mind*, Simon & Schuster, New York

COMPARISON OF SOME LEARNING ALGORITHMS USED IN ARTIFICIAL INTELLIGENCE

Richard R. Gawronski
Computer Science Division
University of West Florida
Pensacola, Florida 32 514

ABSTRACT

A definition of learning based on Kalman's definition of dynamic systems is introduced. General differences in supervised and unsupervised learning are presented. The role of the task of the system and the quality of the performance of the task are emphasized in the description of any learning process. Then the difference between hierarchical and sequential learning was presented. Definitions introduced in this paper enable the comparison of some biological and technical systems. Examples of such comparison are given. Most learning systems which exist in biological system are sequential or hierarchical systems. Investigation of the convergence of such systems is very difficult because most dynamic systems involved in learning are strongly nonlinear. A hypothesis concerning the improvement of convergence by the introduction of sequential or hierarchical learning was presented. Introducing time hierarchy described in section 2, and application of a fix point method enables the investigation and improvement of the convergence for a defined class of learning systems.

1. INTRODUCTION

Ability to learn is considered one of the most important features of intelligent systems both natural and artificial. Designers of AI systems try to implement this feature as efficiently as possible. Nevertheless the term "learning" is understood in so many different ways that it is difficult to compare the ability to learn for different systems, and even to verify whether a system is able to learn or only to memorize some signals. There exist several definitions of learning in the fields of psychology, physiology, AI, Control Theory, and in related fields. Some definitions are very general, for example:

- "*Learning...the process of acquisition and extinction of modifications in existing knowledge, skills,habits, or action tendencies in a motivated organism* " (Webster, 1981);

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

- "*Learning is a more less permanent change in a behavioral tendency as a result of experience* "(Bruno J. 1986, Dictionary of Key Words in Psychology);
- *Learning is relatively permanent change in a behavioral potentiality that occurs as a result of reinforcement practice.*" (Encyclopedia Britannica - after G.A. Kimble).

The latter definition introduces reinforcement as a necessary condition of learning. H. A. Simon (Michalski et. al.,1983) gives a more specific definition: -"*Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same tasks drawn from the same population more effectively the next time.*" This definition uses another undefined term *adaptation* to define learning. More confusing explanations concerning learning we may find in the last book of M. Minsky (1986) "The Society of Mind". In the glossary of this book we find: "***Learning***. *An omnibus word for all processes that lead to long-term changes in our minds.*" This definition includes aging or disability developed by sickness--which can't be accepted. Nevertheless Minsky uses the term learning in many places in the above mentioned book. On the page 120 he claims that "*That word* (learning) *is hard to define.*" but six lines later he gives the definition: "*Learning is making useful changes in the workings of our minds.*" . We have to know what are meant by "*useful changes*" to use this definition. Fortunately for us on the page 78 Minsky discusses "*goal driven*" system which "*...treats the things it finds as objects to exploit, avoid, or ignore....,*" so we can connect learning with one or more goals of the system. More technically oriented is the definition of learning presented by Tsypkin (1973, 1983). Three important aspects of learning are stressed in his definition:

- There is a definite goal of the system usually including a definitive optimality criterion,
- influence of the environment,
- insufficient initial information for using deterministic or statistical decision theory.

"*...the goal of learning is to obtain optimal operating conditions under which the optimality criterion becomes extreme. This goal ...is reached by varying the parameter and, possibly, the structure of the learning system on the basis of processing the data*

on operating conditions and state of the system". This definition is restricted to situations in which it is possible to define an optimality criterion for which an extremum may be defined and measured. In complex systems involving symbolic data processing and decisions, it may be difficult to define an extreme value of the criterion for the selected goal.

Recently, learning processes in AI have been linked with connectionistic theory of neuronlike networks (for example Rumelhart and McClelland, 1986). This linking has generated some concern (Fodor and Pylyshin, 1988) because learning - particularly at higher levels of symbolic processing - is not necessarily restricted to parallel processing, and connection changes in neuronlike networks.

From this short survey of learning definitions we may conclude that it would be convenient to introduce a definition which will be more useful in the construction and evaluation of technical systems, and which would be also acceptable in qualitative psychology. An attempt to introduce such a definition will be presented in the next section.

ϕ

2. DEFINITION OF SELF-LEARNING

Learning is a characteristic which may exist in a dynamic system. Therefore it is necessary to start with a classical definition of the dynamic system (Kalman 1963, Pichler 1983). To define a dynamic system three basic concepts are used:

- The time "t" as an element of a complete ordered set "T" (usually $t \in R^1$);
- A state of the system "q". Mathematically the states of a dynamic system may be represented by elements of a set "Q" (the state set). Depending on the level of abstraction, the state set will have a given formal structure. In general the states of a dynamic system may be objects such as: vectors, matrices, functions, relations, directed graphs, strings of symbols, etc.
- A transformation rule "ϕ" for the system states. By the application of the transformation ϕ each state $q \in Q$ at a time $t \in T$ is transformed to a new state $q' \in Q$ at a time $t' \in T$, that is $\phi(t',t,q) = q'$. Then the state transformation can be described by the form $\phi: TxTxQ \rightarrow Q$.

An autonomous dynamic system is defined by a triple (Q,T,ϕ) satisfying two following axioms:

1. For all $t \in T$ and for all $q \in Q$, we should have $\phi(t,t,q) = q$ (consistency condition).

2. If $t \leq t' \leq t''$ and $\phi(t',t,q) = q'$ and $\phi(t'',t,q) = q''$, then $\phi(t'',t',q) = q''$ (causality condition).

To define nonautonomous dynamic system two additional concepts should be introduced:

- A set of all input states A ($a \in A$) and a related set of input functions "U". For any interval $[t',t) \in T$ we denote by $u[t,t') \in U$ a function $u[t,t'): [t,t') \rightarrow A$.

Input functions may be considered as control functions if a state transformation $\phi: TxTxQxU \rightarrow Q$ where $\phi(\underline{t}',\underline{t},q,u[t,t'))$ is only defined when $\underline{t}' = t'$ and $\underline{t} = t$ is valid.

- A set of output states B ($b \in B$) and a function $\beta: TxQxA \rightarrow B$. The function β assigns to a reached state q at a time t together with an input state a at a time t, an output state $\beta(t,q,a)$ at a time t.

For nonautonomous systems axioms 1 and 2 should be modified as follows:

1a. For $t' = t$, we have $u[t,t') = \emptyset$ (empty set), and $\phi(t,t,q,\emptyset) = q$.

2a With $\phi(t'',t,q,u[t,t'')) = q''$ and $\phi(t',t,q,u[t,t')) = q'$ we should also have $\phi(t'',t,q',u[t,t'')) = q''$

A seven-tuple (A,B,U,Q,T,ϕ,β) defines nonautonomous dynamic system **S** with an input A and anoutput B (Kalman et al., 1969). Behavior of a concrete dynamic system is described by a pair of transformations (ϕ,β).

There exist some more general and more rigorous descriptions of dynamic systems (Mesarovic, 1976).

A dynamic system **S** may be a part of another system $\mathbf{S}_\Omega$ (which is called the metasystem) if all input elements $a \in A$ and all output elements $b \in B$ of the system **S** belong to the set of states Q_μ describing metasystem $\mathbf{S}_\Omega$. Similarly serial and parallel connections of dynamic systems may be defined (Pichler 1983).

For the definition of learning some additional concepts will be introduced:

- An input event E_i is a set of all input states $a_i(t)$ appearing at the input for $t' < t < t''$.
- An output event Ω_i performed by the system is a set of outputs $\beta(t,q,a)$ for $t' < t < t''$
- $\partial_i(t'' - t') = \partial_i$ is the time of the occurrence of the event Ω_i.
- Sets $\mathbb{E}_a$ ($E_i \subset \mathbb{E}_a$) and Ω_b ($\Omega_i \subset \Omega_b$) which belong to appropriate metric spaces $\mathbb{E}$ and Ω; that is for, all elements belonging to the space $\mathbb{A}$ (or appropriately $\mathbb{B}$) a distance $\mu_a(E_1,E_2)$ (or appropriately $\mu_b(\Omega_1,\Omega_2)$) satisfying three metric conditions (identity, symmetry, and triangle inequality) may be introduced. Please notice that we do not introduce these conditions for other spaces Q, A, and B. It means that only the distance between some selected events should be measurable.
- A subset $\mathbf{E^k}$ ($\mathbf{E^k} \subset \mathbb{E}_a$) of distinguishable input events $E_k \in \mathbf{E^k}$

Elements of this subset may be interpreted as the result of the environmental influence which may be important for the system.

- A subset Ω^g ($\Omega^g \subset \Omega_b$) of distinguishable output tasks $\Omega_g \in \Omega^g$.

This subset may be selected by a

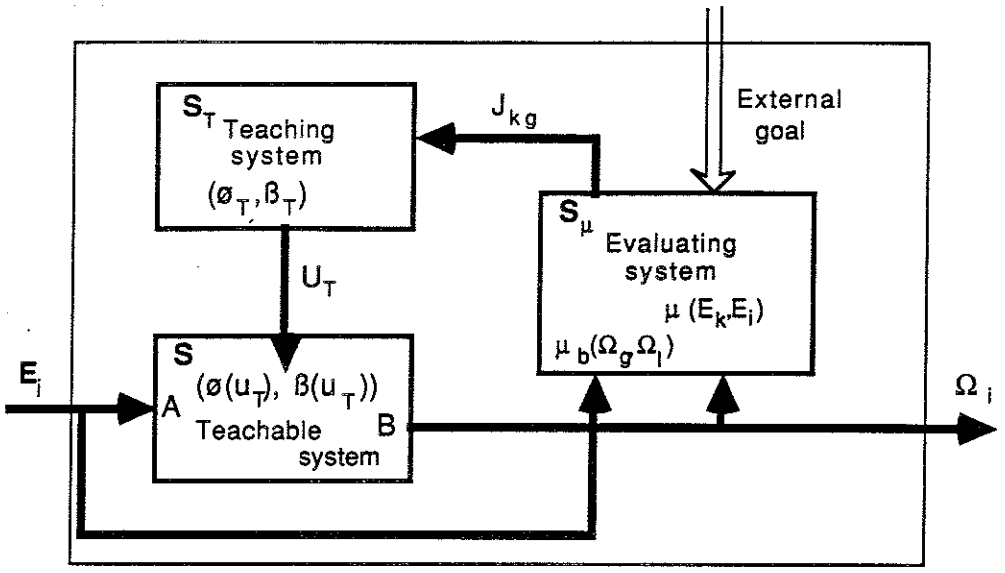


Figure 1. Functional diagram of a learning system without supervision.

metasystem as a subset of desired outputs (or tasks) which should appear under certain conditions at the input.

- A conditional performance factor

$$J_{kg}(\mu_b(\Omega_g,\Omega_i) / \mu_a(E_k,E_i)<d_j) \qquad R^1 \qquad (1)$$

that is a measure of the distance between output event Ω_i and a selected task Ω_g, under the condition that the distance between observed input event E_i and a selected input event E_k is smaller then a defined distance d_k. Let us introduce a simplified notation

$$J_{kg}(\mu_b(\Omega_g,\Omega_i)/\mu_a(E_k,E_i)<d_k) = J_{kg}(\Omega_i,E_i) \qquad (1a)$$

This factor may be interpreted as a measure of the quality of the achievement of the desired task under the condition that a specific input event existed at the input.

There exists also a specific control function $u_C[t,t')$ ($u_C[t,t') \in U_C$) which may be generated outside the dynamic system **S** and which may cause changes in the description of this system (that is in the pair of transformations $\{\phi(u_C[t,t')), \beta(u_C[t,t'))\}$. In this situation we may call the system **S** teachable (or adaptable). We may assume that there exist two additional dynamic systems: $\mathbf{S_T}$, which we call a teaching system, and an evaluating system $\mathbf{S}_\mu$. . The system $\mathbf{S}_\mu$ calculates the performance factor $J_{kg}(\Omega_i,E_i)$ (Figure 1) using the signals from the input and output of the system **S**. The control functions $u_T[t,t')$ may be applied to the system $\mathbf{S_T} = (\phi_T, \beta_T)$, and at the output of this system we have pair of functions (ϕ, β) describing our primary dynamic system **S**. We assume that

$$u_T[t,t') = J_{kg}(\Omega_i,E_i) \; . \qquad (2)$$

Definition: A triple of dynamic sustems $(\mathbf{S}, \mathbf{S}_\mu, \mathbf{S_T})$ may be called a self-learning system with respect to the factor J_{kg} if the processes in the triple $(\mathbf{S}, \mathbf{S}_\mu, \mathbf{S_T})$ satisfy the condition

$$J_{kg}(\Omega_i,E_i) > J_{kg}(\Omega_j,E_j) \qquad (3)$$

for every pair (Ω_j,E_j) satisfying the conditions:

Ω_i and E_i are defined over time interval $\partial_i(t^{(2)}-t^{(1)})$,

Ω_j and E_j are defined over time interval $\partial_j(t^{(4)}-t^{(3)})$, } (4)

$$\mu_a(E_k,E_i) = \mu_a(E_k,E_j) + \varepsilon$$

(ε -small value with respect to the average value of μ), and $t^{(1)} < t^{(2)} < t^{(3)} < t^{(4)}$.

It means that the performance factor J_{kg} defined for the triple of systems $(\mathbf{S}, \mathbf{S}_\mu, \mathbf{S_T})$ is smaller (system's performance is improved) when the same (or similar with a measure of similarity ε) event appear at the input and output of the system.

The definition for a self-learning system $(\mathbf{S},\mathbf{S}_\mu,\mathbf{S_T})$ presented here is consistent with remarks presented by T. Kohonen on non-supervised learning and on self-organizing mode (Kohonen 1988 pp. 83-84). All rules which forces changes in the dynamic system $\mathbf{S} = (\phi,\beta)$ are embedded in systems $\mathbf{S_T}$ (that is in its structure (ϕ_T,β_T)), and $\mathbf{S}_\mu$, and no external (with respect to $(\mathbf{S}, \mathbf{S}_\mu, \mathbf{S_T})$) influence on the description of the system (ϕ,β) may exist. Also self-organizing nets described by Hopfield (1982) and Seinowski (1981) belong to the category of self-learning systems if all changes in the structure of these nets are generated by the imposed on the system tendency to minimize the variables representing energy in the equivalent thermodynamic system.

3. SUPERVISED LEARNING

The teaching system S_T defined in the previous section is connected only to the lteachable system through evaluating system S_μ , and has no connections with the environment. Only the learnable system **S** is connected to the environment. Now let us consider a different situation when functions of the evaluating system are replaced by external influences.

Let us define a pair (S,S_T) composed of two dynamic systems . The system S_T is controlled by a function $u_T[t,t')$ supplied by an external source called a supervisor (Figure 2). The pair of systems (S,S_T)

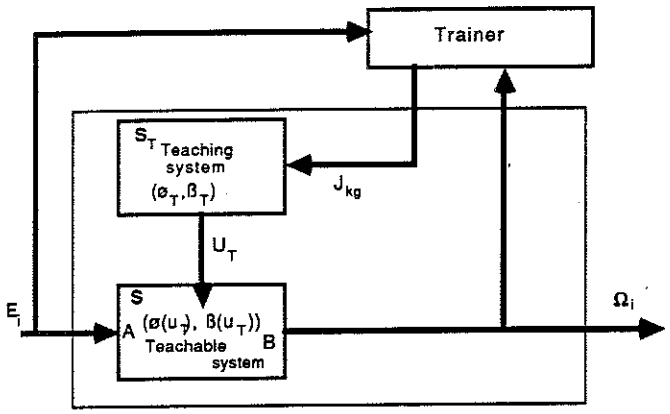


Figure 2. Functional diagram of a learning system with supevision and training.

has now two independent inputs , one u[t,t') for the system **S**, another one $u_T[t,t')$ for the sytem S_T. Equation (2) is no more valid. Evaluation system is replaced by the supervisor performing the evaluation of the performance of the teachable system **S**.

Definition: The pair (S,S_T) is called a *learning system with supervision* if the supervisor calculating performance factors $J_{kg}(\Omega_i,E_i)$ and $J_{kg}(\Omega_i,E_j)$ finds that the relation (3) is true and conditions (4) are satisfied.

If *in addition* to these conditions the sequence of input events E_i is also controlled by an external source then such system is *called a learning system with trainer.*

It is also possible to define a *hierarchical learning system* when the transformations describing the teaching system $S_T = S_{T1}$ (ϕ_{T1},β_{T2}) may be controlled by another teaching system S_{T2} . In this case another performance factor $J_{kg2}(\Omega_{j2},E_{j2})$ should be introduced. A multilevel hierarchical system is described by a 1+2*S-tuple $((...(S,S_{\mu1},S_{T1}),S_{\mu1},S_{T1}),...,S_{\mu S},S_{TS})$. A block diagram of the hierarchical system is depicted on Figure 3. A different situation exists when several learning systems with different learning goals are connected in series (Figure 4). In this situation the output of one system is S_1 is used as an input for another system S_2. Such a system will be called *sequential learning system.* Every system S_1, S_2,...may have his independent pair of teaching and evaluating systems, or may be supervised by an external system. Most probably sequential structures of learning may be found in middle stages of the visual systems of animals. A more specific situation appears when an additional "time hierarchy" exists in the hierarchical or sequential system. At the first stage of the learning process, only the lower level of the teaching system causes the changes in the system S_1. At the next stage of learning the second structural level of the system S_{T2} starts the generation of signals, improving the performance of the second S_2 system. Such situation is very common during the maturation period of the animals.

These definitions are not restricted to physical or technical systems because of very general assumptions on the concept of the state and state transformations. Most objects of investigation in biology also may be considered as dynamic systems. More controversial is the application of this formal

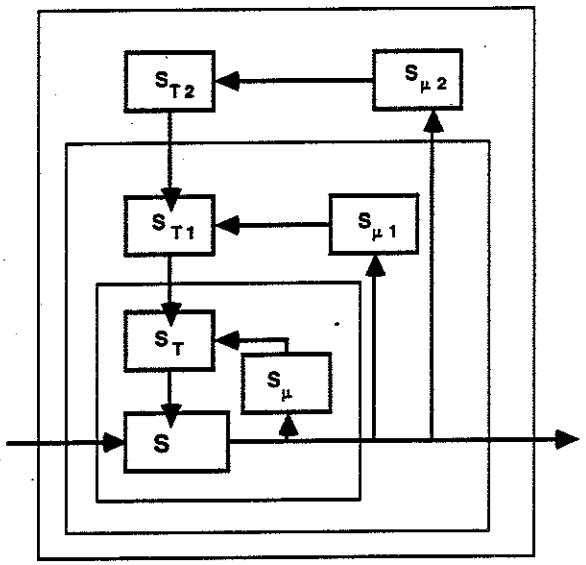


Figure 3. Functional diagram of a hierarchical learning system

descriptions of dynamic systems in psychology. Nevertheless more formal, explicit or implicit, description of learning processes in biological systems may be sometimes helpful, especially when we would like to transfer some results from biological research to technical systems.

4. SHORT COMPARISON OF LEARNING SYSTEMS BASED ON THE DEFINITION OF LEARNING.

The simplest learning process found in biological systems is the habituation when the system learns to ignore some specific continuous or repeating signals. A typical example is the elimination of the reaction (at

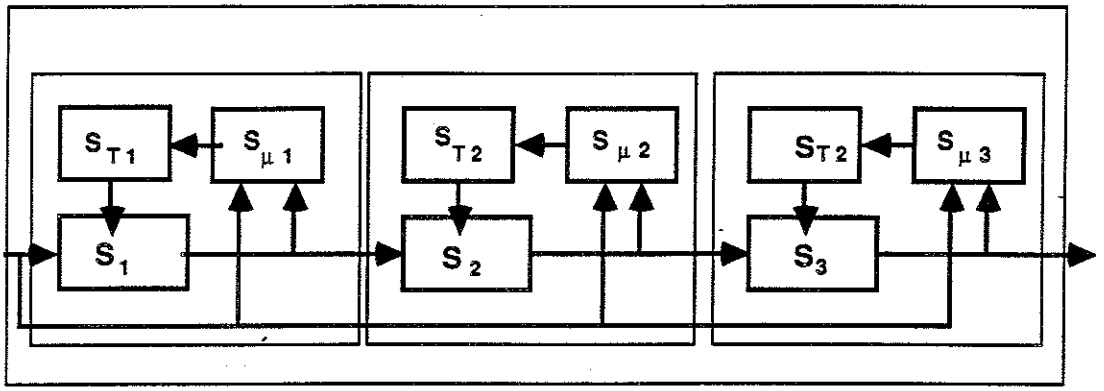


Figure 4. Functional diagram of a sequential learning system.

some higher level of the nervous system) to acoustic signals which repeat periodically. In this case E_i represents the repeating signal and $\Omega_k = 0$ --it means that repeating a signal should give no reaction at the higher level. Even in this simple example we observe hierarchical organization of the learning. Some kind of adaptive filtering enabling selection of undesirable signals must exist at the lower level of the acoustic system. This adaptive filter also is controlled by a learning (unsupervised?) system.

More complex and probably the best known is the the conditioned reflex in the alimentary system. In general it is a hierarchical system (Konorski, 1972, Grossberg, 1988). When the level of hunger signal (coming from another part of the nervous system) increases the performance index $J_{kg}(\Omega_i,E_i)$ should enforce the situation by improving the swallowing of food through an increased salivation Ω_i. To avoid unnecessary salivation, a learning system combines some events E_i at some sensory input preceding the feeding process with the application of the food E_k. The symbol Ω_k represents an optimal salivation, and $\mu_a(E_k,E_i)$ represents the time interval between conditioned and unconditioned signals.

Pattern recognition (visual, acoustical or any other which may isomorphic to a subspace of a metric space) is the most typical example of the possible application of the learning in technical systems. In most situations algorithms for image classification may be described as supervised learning with trainer. A class of input images E_i satisfying the condition $\mu_a(E_k,E_i)<d_i$, where E_k is a template image for the given class, is applied to the system during the training session. The supervisor evaluates the answer of the classifying system **S** and sends a signal (usually binary) to the teaching system $\mathbf{S_T}$, which in turn corrects the transformations (ϕ,β) describing the system **S**. The performance factor $J_{kg}(\Omega_i,E_i)$ in this case is the average number of errors at the output of the system **S**. If a neuronlike network is applied as a classifying system, then the system $\mathbf{S_T}$ imposes weight changes in the network. Best known rules for connection changes, that is delta rule (Widrow and Hoff, 1960) and then later backpropagation (Werbos, 1974) are executed by the teaching system $\mathbf{S_T}$ on the basis of internal or external information about errors at the output of the system. As we know (Minsky and Pappert, 1988) application of neuronlike nets in pattern recognition gives poor results and in many cases the system is not convergent. Much better results may be obtained if a sequential learning system is used. At every level *s* (s=1,2,...S) of the sequential system a different subtask Ω_{gs}, and a different performance index $J_{kgs}(\Omega_{is},E_{is})$ is introduced. Usually at the first stage a set of local features is detected. Then at the next stages some local and global topological features are selected and used for the final classification. Introduction of a time hierarchy may considerably improve the convergence of the whole learning procedure. Even more important is the fact that the combined introduction of sequential learning and time hierarchy may eliminate local extrema which usually appear when a multilayer neuronlike network is controlled only by one performance index. Such a time hierarchy in learning was also found in the visual tract of the nervous system of higher animals.

Another example of the learning system is the self-optimization of movement control of a manipulator (Gawronski 1984, Miyamoto et al. 1988). In this case the input event E_i is a selected command which should start a selected movement of the manipulator. In the non-supervised control a desired final point (or trajectory) Ω_g is introduced to the evaluating system $\mathbf{S_\mu}$. The performance index $J_{kg}(\Omega_i,E_i)$ describes the distance between the desired Ω_g and current Ω_i output state of the manipulator. The next level of optimization, for example minimum time of the movement performance (Gawronski 1984), may be introduced in the system as a next hierarchical level of the learning procedure. Also in this case introduction of additional time hierarchy improved the convergence of the learning procedure.

The definition of the learning presented in this paper does not pretend to a full mathematical rigor but may be useful in the description and comparison of different learning system, especially when we compare different kinds of objects (technical, biological, computer models). The role of the task of the system and the quality of the performance of the task was emphasized in the description of any learning process. Convergence is the most difficult problem in the learning systems. An important **hypothesis concerning convergence** of learning may be formulated. *If a learning system is not convergent it is reasonable to introduce sequential learning. If it is impossible to formulate local goals for every step of learning, try to introduce hierarchical learning.* Most learning systems which exist in biological system are hierarchical and/or sequential systems. Investigation of the convergence of such systems is very difficult because most dynamic systems involved in learning are strongly nonlinear. Introducing time hierarchy described in section 2, and application of a fix point method (Gawronski & Macukow 1977) enables the investigation of convergence for a defined class of learning systems.

Acknowledgment. The author is grateful for Dr. Royce Harbor who carefully read the manuscript and provided essential corrections.

REFERENCES

Fodor, J. A. and Pylyshin Z. W. 1988: "Connectionism and Cognitive Architecture: A Critical Analysis." In Pinker S. and Mehler J.: *Connections and Symbols.* Cambridge Mass.:The MIT Press.

Grossberg, S. *The Adaptive Brain Vol. I and II.* Amsterdam New York. North Holland Pub.

Gawronski R. R.1984 "Two Stage learning Learning Algorithm for Optimal Control of a Manipulator." In *Proceedings of the Conference Robotics Research: The Next Five Years and Beyond* (Bethlehem, Pa, August 14-18), SME, Deaborn Mi. MS84-488 :1-10.

Gawronski, R. and B. Macukow.1977. "The Adaptive Neuronlike Layer Net as a Control Learning System",in: *Progress in Cybernetics and System Research v.III.* , eds Adv.Publ.Ltd, London, 461-466.

Hopfield, J. J. 1982 "Neural Networks and Physical Systems with Emergent Collective Computational Abilities" Proc. Natl Acad. Sc. USA Vol. 79 April pp. 2554-2558.

Kalman, R. , Falb P. , and Arbib M. 1969. *Topics of Mathematical System Theory.* New York, McGraw-Hill.

Kohonen, T. 1988. *Self-Organization and Associative Learning.* Berlin, New York. Springer-Verlag.

Konorski,J. 1967. *The integrative Activity of the Brain.* Chicago, Chicago University Press.

Minsky, M., 1986.*The Society of Mind.* New York, Simon and Schuster.

Munsky, M. Papert, S. P. 1988. Perceptrons. Expanded edition. Cambridge Mass.:The MIT Press.

Mesarovic M. D., 1972. "Mathematical Theory of General Systems". In Klir J. G. ed. Trends in General System Theory. New York, John Wiley.

Michalski R. S. Carbonell J. G. Mitchell T. M. 1983 v I, 1986 v II: *Machine Learning.* Los Altos Cal. :Morgan Kaufman Pub.

Miyamoto, H., Kawato,M., Setoyama,T.,Suzuki,R. 1988. "Feedback-Error-learning Neural Network for Trajectory Control of a Robotic Manipulator." *Neural Netvorks.* 1, 251-265.

Pichler, F. 1983. Dynamical System Theory. In R. Trappl ed: *Cybernetics Theory and Applications.* Washington: Hemisphere Publ. Co.

Rumelhart, D. E., McClelland J.L. and PDP Research Group, 1986 *Parallel Distributed Processing"* vol. I and II, The MIT Press Cambridge Mass.

Seinowski, T. J. 1981. "Skeleton Filters in the Brain". In Hinton G. E. & Anderson (Eds) Parallel Models In Associative Memory. pp. 49-82. Hillsdale N.J. Erlbaum Pub.

Tsypkin, Y. Z. (1973): *Foundations of the Theory of the Learning Systems.* New York: Academic Press.

Tsypkin, Y.Z. (1983): The Theory of Adaptive and Learning Systems. In R. Trappl ed: *Cybernetics Theory and Applications.* Washington: Hemisphere Publ. Co.

Rumelhart, D. A., McClelland J. L. (1986): *Parallel Distributed Processing* v. I and II. Cambridge Mass.:The MIT Press.

Werbos, P 1974. *Beyond regression: New tools for prediction and analysis in the Behavioral Sciences.* Ph. Thesis, Harvard Univ. Aug. 1974

Widrow, G., and Hoff M. E. 1960. "Adaptive Switching Circuits" in *Institute of Radio Engineers, Western Electronic Show and Convention Record* Part 4, 96-104.

Comparison of Learning Algorithms for Multi-Layer Neural Networks

[1]Greg L. Heileman, [2]Michael Georgiopoulos, [1]Harley R. Myler and [3]George M. Papadourakis

[1]Department of Computer Engineering, [2]Department of Electrical Engineering, University of Central Florida, Orlando, FL 32816-0450

[3]Department of Computer Science, University of Crete, Iraklion, Crete, Greece

* This work was supported by the Institute of Simulation and Training at the University of Central Florida

Abstract

A general model of an artificial neural network is presented and learning algorithms are discussed in terms of this model. A number of algorithms used in training multi-layered artificial neural networks are simulated on an exclusive-or network. A comparison of these algorithms is performed based upon the ease of implementation and the number of iterations required to reach solution.

1. Introduction

Artificial Neural Network (ANN) models have recently received widespread attention in the scientific community. These models, which loosely resemble the anatomy of the human brain, offer potential solutions to problems that have plagued artificial intelligence researchers for many years.

Compared to the components in conventional computer systems, the neurons in the human nervous system are slow, with switching speeds on the order of 10^6 times slower than current logic devices. In spite of this, people are able to outperform computers in a wide range of cognitive tasks. For example, people are able to perceive objects in natural scenes, understand language, and retrieve contextually appropriate information from memory — tasks that overpower even the most sophisticated algorithms running on the most powerful machines. Moreover, we are able to perform this complicated processing in a few hundred milliseconds, which is approximately 100 times the firing rate of neurons. This means that such tasks must be accomplished in only 100 computational steps. It appears that the brain succeeds by using many computationally simple neuronal elements in parallel [1]. It is from this perspective that researchers have proposed ANN models capable of examining numerous competing hypotheses simultaneously through the use of massive interconnections among many simple nonlinear processing elements. They believe that these techniques are essential in order to achieve human-like performance in such areas as speech and image recognition.

A major problem involved with applying these networks is the difficulty of training them to perform appropriate tasks. Thus far only small networks (typically much less than 100 processing elements) have been successfully trained using ANN learning algorithms. To fully exploit the power of these models, they must be scaled up to much larger sizes. However, it is widely acknowledged that as the networks get larger and deeper the training time grows prohibitively. In fact, it has been shown that this training problem is NP-complete and therefore has no known efficient solution [2]. The issue then becomes one of finding ways to avoid the training problem in its full generality.

This paper presents simulation results for a number of currently popular ANN learning procedures. The performances of these algorithms are analyzed according to their convergence time and ease of implementation.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

2. A General Framework for Artificial Neural Network Research

The state of the art in ANN research is very broad, yet most of these efforts make use of the basic ANN model shown in Figure 1. Each node or processing element (PE) in the network has an activation value $a_i(k)$ which depends upon the net input to the i-th node after k iterations; this value is passed through an output function f_i to determine the output $o_i(k)$. Next, the output value is propagated through a network of unidirectional connections to other PEs in the system. Associated with each connection, there is a synaptic weight w_{ij} which determines the amount of effect the i-th node has on the j-th node. All of the inputs to PE_j at time k are combined according to some operator, along with an internal input θ_j, to produce the net input to PE_j [3].

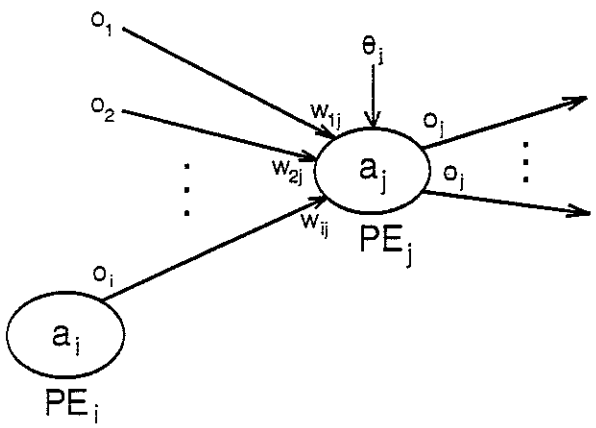


Figure 1. The basic model for ANN research

It is the pattern of activation levels in the nodes that determines what the system is representing at any given time. Thus, it is useful to view the processing in these systems as an evolution, through time, of a pattern of activity in the network.

All that remains to completely specify an ANN model is to determine a network topology and learning rule. The learning rule determines the method in which the connection weights between nodes are adjusted in order to improve performance.

3. ANN Learning Algorithms

3.1 Algorithms for Single Layer Networks

Learning algorithms for single layer neural networks have been available since the early 1960's. Rosenblatt introduced the famous Perceptron Convergence Algorithm with a corresponding proof that the algorithm could train a single layer network to distinguish between linearly separable pattern classes [4]. An adaptation of this algorithm was formulated that minimized the mean square error between the desired and actual outputs of the network. This procedure was termed the Least Mean Square (LMS) algorithm [5]. This algorithm also had the desirable property that weight changes were made in a manner that minimally disturbed the system. However, both of these approaches failed if the pattern classes under consideration were not linearly separable, in which case the weight values simply oscillated as long as the algorithms were allowed to execute.

The extension to multi-layered networks allowed researchers to represent nonlinearly separable functions, however reliable training algorithms were not available at the time [6]. Recently, learning algorithms for multi-layered networks have been derived from the LMS algorithm, a number of these are discussed below.

3.2 Algorithms for Multi Layered Networks

The Back-Propagation Algorithm

The back-propagation algorithm extends the error minimization aspect of the LMS algorithm to multi-layered networks. This algorithm accomplishes learning through the minimization of a cost function representing the mean square error between the desired and actual outputs. This is achieved using a gradient search which corresponds to a steepest descent on a surface representing the weight space; where the magnitude at any point in the weight space is equal to the error measure. These calculations require estimating the partial derivatives of the error with respect to each weight in the network. This means that the activation functions must be continuous and differentiable. In addition, since a multi-layer network with linear activation functions provides no advantage over a single layer network, the back-propagation algorithm uses nonlinear, continuous, differentiable activation functions [6].

The gradient search implemented in this algorithm involves two steps. A forward step and a backward step. In the forward step, a training pattern is presented and propagated forward through the network according to the following equations:

$$u_{jl}(k) = \sum_{i=1}^{N_l} w_{ijl}(k) o_{i,(l-1)}(k) + \theta_{jl}(k) \qquad (1)$$

$$o_{jl}(k) = f(u_{jl}(k)) \qquad (2)$$

$$l = 1, 2, ..., L \qquad j = 1, 2, ..., N_l$$

In the backward step the output values calculated in the forward step are compared to the desired outputs generating error signals. This error signal is propagated backward through the network to allow the recursive computation of weight and internal threshold adjustments in each layer as shown below:

$$w_{ijl}(k+1) = w_{ijl}(k) + \eta \delta_{jl}(k) o_{i,(l-1)}(k) \qquad (3)$$

$$\theta_{jl}(k+1) = \theta_{jl}(k) + \eta \delta_{jl}(k) \qquad (4)$$

where η is the learning rate, and δ_{jl} is the error signal at the j-th node on the layer l. These error signals are obtained recursively as follows:

$$\delta_{iL}(k) = (t_i^m - o_{iL}(k)) f'(u_{iL}(k)) \qquad (5)$$

$$\delta_{il}(k) = f'(u_{il}(k)) \sum_{j=1}^{N_{(l+1)}} \delta_{j,(l+1)}(k) w_{ij,(l+1)}(k) \qquad (6)$$

$$l = L-1, L-2, ..., 1.$$

where f' is the derivative of the activation function f, and t_i^m represents the desired or target value of the i-th output node on the m-th pattern presentation [6].

Specifically, if a sigmoid activation function is used, i.e.,

$$f_s(u_{jl}(k)) = \frac{1}{1 + e^{-u_{jl}(k)}}.$$

then the error signals become:

$$\delta_{iL}(k) = (t_i^m - o_{iL}(k)) o_{iL}(k)(1 - o_{iL}(k)) \qquad (7)$$

$$\delta_{il}(k) = o_{il}(k)(1 - o_{il}(k)) \sum_{j=1}^{N_{(l+1)}} \delta_{j,(l+1)}(k) w_{ij,(l+1)}(k) \qquad (8)$$

$$l = L-1, L-2, ..., 1 \qquad i = 1, 2, ..., N_l.$$

Although the back-propagation algorithm is guaranteed to search for a minima in the cost function, it may converge to a local minima in this function that is quite different from the global minima representing the desired solution. Another limitation involves the large amount of training time that is generally required before convergence occurs. Researchers have proposed a number of modifications to this algorithm intended to alleviate these limitations. One example is dynamic learning rate adjustment in which the learning rate used in weight adjustment is allowed to vary over time as the surface of the error function varies [7][8].

The MRII Algorithm

The MRII algorithm heuristically extends the minimal disturbance aspect of the LMS algorithm to multilayered networks. The training objective is to reduce the Hamming distance between the actual and desired responses. This algorithm performs the forward step as specified in equation (1) and (2) using a threshold activation function,

$$f_h(u_{jl}(k)) = \begin{cases} +1, & \text{if } u_{jl}(k) \geq 0 \\ -1, & \text{otherwise.} \end{cases}$$

The next step in the algorithm specifies the adjustment of weights in the network according to the following heuristics. Adapt the first-layer node j whose net input is closest to zero as follows:

$$w_{ijl}(k+1) = w_{ijl}(k) + \eta \frac{\delta_{jl}(k) o_{i,(l-1)}(k)}{\sum_{i=1}^{N_l} o_{i,(l-1)}^2(k)} \qquad (3)$$

where $w_{ijl}(k)$ is the value of weight w_{ijl} at iteration k, η is the learning rate, and δ_{jl} is the error at node j on layer l (i.e., the difference between the desired net input of the node and the actual net input of the node). The desired net input of an output node should be chosen so that it

produces the desired output. The desired net inputs of nodes other than the output nodes are chosen in such a manner as to change the sign of a particular node's output. In both cases, the magnitude of the resultsing desired net input should be small.

It can be shown that the weight adaptation presented above minimally disturbs the network weights while performing the desired error correction [9]. If this adaptation reduces the Hamming distance between the desired and actual output of the network (i.e., the Hamming error), then the weight changes are saved. Otherwise, the weight values are restored to their previous values and the first-layer node whose net input is next closest to zero receives a trial adaptation. This continues through all nodes on the first layer, accepting all weight changes that reduce the Hamming error in the network output. Next, first-layer nodes are trial adapted in combinations (i.e., pairs, triples, etc.) up to some predetermined limit in combination size. After this is completed, the second-layer weights are adapted in the same manner. If further error reduction is desired, then the weights of additional layers, including the output layer, may be trial adapted as well. A major limitation in the use of this algorithm is the difficulty in reasoning why the adaptations should be stopped at a particular combination of nodes or at a particular layer.

The various heuristics associated with this algorithm have been used with success on a number of learning problems [9,10]. However, a mathematical proof of the algorithm's convergence has not been forthcoming.

4. Simulations

The back-propagation algorithm and a number of its variants, along with the MRII algorithm were simulated on an exlcusive-or network. The exclusive-or network consisted of two input units connected to two hidden units which were connected to a single output unit. Four binary input patterns are presented to the network. The output unit is required to map these inputs into two classes, one class represents inputs that are identical and the other represents input that are different. Solution of this task occurs when the network is able to map each input pattern to the desired output pattern with an error tolerance of 0.01.

The time complexity of a particular simulation is given in terms of the number of iterations required to solve the problem. For each algorithm tested, twenty-five networks were simulated. The learning rate at which a particular algorithm performed best (in terms of the number of networks that reached solution) is presented.

The back-propagation algorithm reached solution on seventeen of the networks with a mean convergence time of 4280.23 iterations and a standard deviation of 5765.22. The best performance achieved using a variation of the back-propagation (the minimal disturbance back-propagation algorithm [11]) was solution on all twenty-five networks with a mean of 973.0 iterations and a standard deviation of 267.1. Finally, the MRII algorithm (with a maximum combination size of two) reached solution on all networks with a mean of 124.16 iterations and a standard deviation of 129.69.

5. Conclusion

Results show that the MRII algorithm has a much shorter training time and better convergence properties on the exclusive-or network than the back-propagation type algorithms. It is also worth noting that these convergence rates have much less variability than those of the back-propagation type algorithms. We are currently testing these algorithms on more complicated networks to see if the same results apply. Finally, since the MRII algorithm does not require nonlinear continuous differentiable activation functions, it should yield a much simpler hardware implementation.

6. References

[1] J.A. Feldman, Connectionist Models and Parallelism in High Level Vision in *Human and Machine Vision II* (A. Rosenfeld, Ed.), Academic Press, Orlando, FL, 1986.

[2] S. Judd, Learning in Networks is Hard, *Proc. IEEE Intl' Conf. on Neural Networks*, San Diego, CA., vol.II, pp.685-692, 1987.

[3] S.Y. Kung and J.N. Hwang, Parallel Architectures for Artificial Neural Nets, *Proc. IEEE Intl' Conf. on Neural Networks*, San Diego, CA., vol.II, pp.165-172, 1988.

[4] F. Rosenblatt, *Principles of Neurodynamics*, Spartan Books, Washington, D.C., 1962.

[5] B. Widrow and M.E. Hoff, Jr., Adaptive Switching Circuits, in *IRE WESCON Convention Record Part IV*, pp.96-104, 1960.

[6] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, Learning Internal Representations by Error Propagation, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol 1: Foundations* (D.E. Rumelhart and J.L. McClelland, Eds.), MIT Press, 1986.

[7] D.R. Hush and J.M. Salas, Improving the Learning Rate of Back-Propagation With the Gradient Reuse Algorithm, *Proc. IEEE Int'l' Conf. on Neural Networks*, San Diego, CA. vol.I, pp.441-447, 1988.

[8] Jacobs, Increased Rates of Convergence Through Learning Rate Adaptation, *Neural Networks* 1, pp. 295-307, 1988.

[9] B. Widrow, R.G. Winter and R.A. Baxter, Layered Neural Nets for Pattern Recognition *IEEE Trans. Acoustics, Speech, and Signal Processing 36(7)*, pp.1109-1118, 1988.

[10] R.G. Winter and B. Widrow, MADALINE RULE II: A Training Algorithm for Neural Networks, *Proc. IEEE Int'l' Conf. on Neural Networks*, vol.I, pp.401-408, 1988.

[11] G.L. Heileman, M. Georgiopoulos and H.K. Brown, The Minimal Disturbance Back-Propagation Algorithm, submitted to the *IEEE Int'l' Conf. on Neural Networks*, 1989.

TEXTURE ANALYSIS USING NEURAL NETWORKS

Stephan R. Yhann and Tzay Y. Young

Department of Electrical and Computer Engineering
University of Miami
P.O.Box. 24-8294, Coral Gables, FL 33124
Phone 305-284-3291

Abstract

In this paper a Neural Network design is presented for image texture analysis and classification. The network structure is hierarchal with the resolution decreasing from the lower levels of the network to the higher levels. The network makes use of a set of basic features called 'textons' which are orientation and size dependent. The feature information is available at various levels of the network and were used to classify random texture consisting of oriented lines of various sizes. The classification was also achieved via a neural network implemetation. Training of the neural network was done using a multi-layer back propagation algorithm.

Key words: Neural Networks, Texture Analysis Segmentation.

Introduction In this paper a hierarchal neural network structure suitable for image texture analysis is presented. This neural network consists of two functional stages; a feature extraction stage, whose outputs serve as the input to the second stage which classifies the image texture. The justification for using neural networks for textural analysis lies in the way these networks store and process information. Neural networks consist of a set of richly interconnected,simple computational elements, whose weighted interconnections store the networks information. The hierarchcal structure of the network allows independent design of each layer in a multilayer system: in this case each layer may consist of a specific neural sub-network. Neural networks also have the ability to 'learn' to perform specific tasks, and can be trained to analyze data that can not be expressed in precise mathematical quantities. This ability of neural networks to learn to analyze large amounts of input data and process them in parallel, and to form its own internal representations of the data, recommends neural networks for the task of image texture analysis. Neural networks present a medium through which the global relationships between the local features of an image can be expressed. This is very important, since in texture analysis it is necessary to obtain global features as well as local features over the image.

Most of the previous work done on texture analysis has been concerned with defining and obtaining measures for the quantitative description of various textural types such as: average gray levels, Fourier analysis and gradients, and probability co-occurrence matrices [1] These measures produce quantitative values, using mathematical operators over the image, to describe the image. Accordingly, a priori information of the source, or nature of image, is normally required before a choice of textural measure can be made. Methods using masks to extract gradients and edges in specified orientations and the outputs of these masks to determine a texture energy measure have been suggested by Laws [2]. These methods come somewhat closer to our approach.

In this paper, use is made of a set of basic texture features, termed textons, comprising of lines of varying orientation, width, and length, end-of-line terminations, line crossing, and spots. The existence of textons is collaborated by the discovery of receptors in animal visual systems that respond to such stimuli [3]. There is also physiological evidence indicating that the human visual system does use features similar to the textons described above, to perform analysis on texture at a pre-attentive level: the pre-attentive response is the response generated when the image is presented to the eye for a very short period of time (around 100msec) [4]. It is also believed that texture plays a very important role in the pre-attentive visual system and that the textons described, along with a few others such as color contrast, flicker rate and binocular and movement disparity, form a set of features by which the human visual system performs texture analysis. Supporting the physiological and psychological evidence justifying the use of the textons in texture analysis, is the added advantage that texton feature extraction is suitable for implementation in a neural network system [5,6].

A bottom up approach was used in the network development. This paper stresses the utility of this approach. The sub-networks were designed to detect features on a

This work was support in part by a NSF grant IRI-8711405 and by a grant from Florida High Technology and Industry Council.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

microscopic level, involving small neighborhoods about each pixel, and using very simple features. Each of the sub-networks were designed to detect a set of basic feature types. Hence, the network size depended on the number of features used. The fundamental subnets are shown to have learnt to detect the features on which they were trained. Larger networks were formed by combining the subnets, and adding additional layers, which were trained to detect larger features and had lower resolution. At the topmost level of the hierarchal structure a classification sub-network was trained and used to perform textural discrimination. The training was done using a multilayer back propagation algorithm.

In tests run on images containing differing textural patterns,the neural network presented was able to detect the presence of those features for which it was trained, and provide textural discrimination between regions of differing texture.

The neural network structure developed has the advantage of being able to build on itself and derive more complicated features composed of simpler forms. Also, because the structure is hierarchal and recursive, the training of larger networks is made simpler since these, for the most part, are derived from simpler forms which can be more easily trained. Furthermore, the network is flexible in the feature types used in the texture analysis. The feature set can be modified or updated quite easily, since this involves merely modifying the weights in the structure. Additional features can be added by increasing the size of the network, however no major changes need be made to the network components, since the network is composed of quite simple elements: nodes and their connections.

A description of the Network is presented in the next section of this paper. The design of the feature and classification stages of the network is then covered. This is followed by the results of simulations with the neural network to classify textural images. Finally some concluding remarks are made.

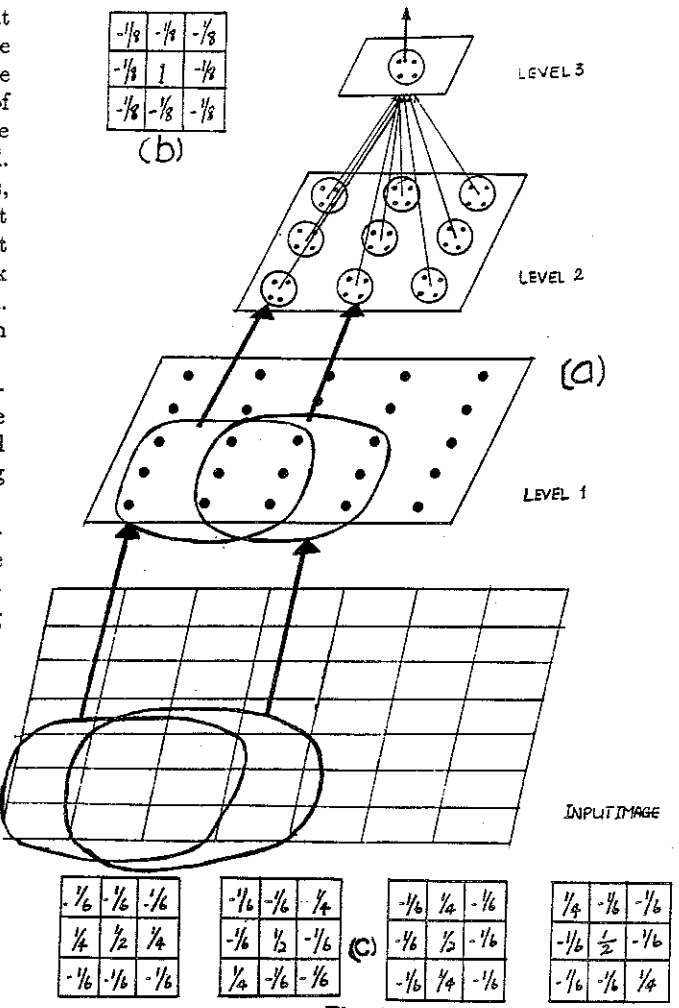


Fig. 1.

Feature Network Description and Design

Description of Feature Network: The elementary, more fundamental sub-nets in the feature extraction network were heuristically designed or individually trained to detect the presence of the desired features within a specified neighborhood. Input image field neighborhoods of 3 × 3, 5 × 5 and 7 × 7 pixels were used, with the smallest input field serving as input to the neurons in the lowest level in the network and the largest to the highest level. This increase in input field size corresponds to the decreasing resolution of the higher levels.

The general structure of the feature network is given in Fig. 1(a). In going from the lower levels of the network to the higher levels the scope of the input image field, to which the levels outputs respond, increases. The underlying philosophy is to have each lower level extract textural features which could be used as input to the higher levels as well as to provide textural information.

The objective of the first level of the network shown in Fig. 1(a) is to provide information on the 'spottedness' of the image and to increase the contrast between lines while suppressing the background. Therefore, the second level perceives lines of varying intensity with little regard to intensity variation. Each neuron of the first level of the network was designed to have a maximum response to a spot of size 1 pixel located at the center of its 3 × 3 input field and exhibited an isotropic response.

The second level provides information on the connectivity over a three pixel range in the four possible orientations within the 3 × 3 input field from level one. This information is in itself a useful feature since it also serves as the input to the third level of the network. The second level of the network contains four types of neuron detectors and provides information on the orientation of pixels over a 5 × 5 pixel area of the image. The outputs of the neurons in the first level from a 3 × 3 feature neighborhood are connected to each neuron in the second level. The neurons in the second level were designed to maximally respond to layer one inputs oriented at, 0,45,90 and 135 degrees, through the center of the of their input field.

The neurons in the third level of the network were trained to have a maximum response to lines of length 5 pixels and width 1 pixel, oriented through the center of the

input image field. The neurons in the third level have maximal responses at orientations 0,30,45,60,90,120,135,150 degrees. The input to the third level consists of the second level outputs in overlapping 3×3 fields contained within a 5×5 neighborhood of the second layer. This corresponds to a 7×7 neighborhood within the image field. The network was trained so that each neuron would respond to only one of the orientations listed above, and have a partial response to the orientations higher or lower than its maximal response orientation. The response for each neuron is symmetric about the maximal response, and decreases with increasing deviation from maximal response line orientation, length and width, as well as position within the input image field. For example, the neuron trained to respond to a 5 pixel line, oriented at 45 degrees through the center of the input image field, would have an equal response to a 30 or a 60 degree line through the center of the image field. However, the response to these lines would be reduced to 0.5 compared to 0.9 for a 45 degree line. If a 4 pixel line, oriented at 45 degrees were in the input field of the same neuron, a reduced response would also be experienced. The trained network will also respond partially to a line which has been translated with respect to the center of the input image field. The reduction in this response will increase with increasing distance from a like oriented line through the center of the image field.

To summarize this, the third level was trained to respond to lines of varying orientation and length. The neurons can then be said to be 'tuned' to a specific orientation and size. Lines of reduced length or of differing orientation will evoke a reduced response from the neurons. The magnitude of the response is dependent on the degree of difference between the input lines' size and orientation and that of the line to which the neuron is to respond. The neurons do not discriminate between positive and negative orientation differences. Thus a single line, with a non-tuned orientation, will produce a response in the neurons with tuned orientation angles larger and smaller than its orientation.

The feature networks were distributed over the entire image and grouped into neighborhoods about each pixel point. The feature outputs within each neighborhood serve as the the input to a network which determines the feature content within the neighborhood and classifies the region on this basis. The feature content measured in terms of the first order statistics (averaged outputs) gave an estimate of the concentration, or count, of each feature type in the neighborhoods, so that the neighborhoods with similar feature statistics could be grouped to form homogenous regions. The feature statistics collected over each neighborhood and homogenous region were used to describe the textural content of the image in the different regions. This averaging process was used to take into account the random nature of texture. The spatial relationship between the different features was implicitly taken into account by the spatial arrangement of the neurons over the image plane.

Considering the hierarchal structure of the network, it is seen that when the resolution of the first level is reduced, the network with the higher levels unchanged can be used to detect thicker and longer lines. Also, only minor modifications of the first level are required for the network to extract dark lines on bright back grounds and edges.

Network Design: The design of levels one and two of the network were fairly straight forward. The connection weights for the sub-networks in these two levels are based on the well known point and line templates [7] and are given in Fig. 1(b). The activation function

$$f(\bar{x}) = \frac{1}{1 + \exp \beta(\bar{w}^T \bar{x} - \theta)} \tag{1}$$

is used to decided the presence of a line or point within the input field of the template. The thresholds and beta parameters for the activation function were experimentally determined. These thresholds and beta parameters were selected to give both a sharp response to spots and a partial response to lines of size one pixel suitable as input to the second level of the network (in the case of level one activation functions), and to produce a favorable response to lines of the desired orientation while ignoring lines of other orientations (in the case of the second level: assuming as input the output of the first level).

The weights from the outputs of the level two neurons to inputs of the level three neurons were trained using the back propagation algorithm by Rumelhart [8], with some minor modifications.

The tuned orientations of the neurons in the third level were based on the resolution of the imaging system used. A sequence of training 7×7 pixel images consisting of lines of sizes and orientations, 5 to 3 pixels and 0 to 180 degrees respectively, were generated. The lines were positioned in the center of the training image field as well as off center. The desired outputs for the lines were heuristically assigned, based on the orientation,size, and position of the line within the training image field. The weights from the second level to the third level of the network for the 0 to 60 degree orientation neurons were learnt. These weights were assigned to the corresponding connections for the 90 to 150 degree neurons rotated through 90 degrees. Learning was done in the first quadrant (0-90 degrees) to ensure symmetrical weight setting in both quadrants (0-90 and 90 -180 degrees). If this symmetry were not present, then in the presence of two lines in the input image field, the networks response would be unpredictable. In this case the network may only respond to one line while totally ignoring the other. This effect is most noticeable when the lines are perpendicular to each other. This results because a line perpendicular to the tuned orientation of the neuron tends to have an inhibitory effect on the response. If the network is asymmetrical, then the amount of inhibition experienced by the neurons corresponding to the input two line orientations will not be equal, resulting in one neuron dominating the other, and an asymmetrical output. Forcing symmetry prevents this. A symmetrical weight assignment does not result if all of the weights in both quadrants are learned, since the initial weights are randomly assigned and the input training image sequence has a random ordering of line orientation types.

Results of Feature Extraction

The feature extraction stage of the network in Fig. 1(a) was first applied to 7×7 sub-images of the type used to train level three of the network. The output of the third level of the network was used to predict the input line type. It was found that using these outputs, it was possible to identify the presence of the different line types. When double parallel lines and intersecting lines were applied at the input field, the network was able to detect their presence with mixed success. It was noticed that when a pair of parallel lines were too close together, it was impossible to clearly distinguish between them and a like oriented line through the center of the image field, based solely on the outputs of the third layer. This is due to the two parallel off center lines providing a stimulus equivalent to that of a single line through the center. To distinguish the two events the outputs of level two are required.

With intersecting lines present in the input field, it was more difficult to predict the input line types present.The success of prediction depended on the relative orientation of the two lines to each other and their position within the image field. If the lines were nearly perpendicular and positioned near the center of the image field then detection was easy. However, with increasing offset from the center of field and reduced orthogonality it became increasingly difficult to classify the input line type; usually the the one nearest the center dominated.

Classification Stage

Description of Classification Stage: A neural network was trained to classify a textured image containing four texture types. The classifier neural network consisted of two layers. The input layer to the classifier stage contained sixteen neurons which were all connected to the four neurons in the output layer directly above. This network structure was associated with each point in the image field and was used to code information on the spatial distribution of the features in the neighborhood of each pixel. The primary function of the input layer was to provide enough neurons so that this spatial feature distribution information could be captured. It also served the secondary purpose of matching the large number of inputs from the feature field, 64 for each of the four feature types, to the relatively small number of neurons in the output layer. This was necessary to reduce the learning time. Each one of the four output neurons was trained to respond to one of the four textural types present in the input image field. A synthetic image containing lines of pixels in one of four orientations (0, 45,90,120) was generated.

The lines were randomly distributed within the four textural regions. The image used is shown in Fig. 2(a) with the four level three feature outputs in Fig. 2(b): image intensity coresponds to output activation level. This image was used as the input to the neural network simulated on a VAX 11/750 computer. The four level three orientation feature types present in the input image, were averaged over a 4×4 neighborhood, and used as input to the classification stage.

These four proved to be more than adequate to perform the textural classification using the neural classification stage.

The multi-layer back propagation learning algorithm was used to train the classification stage of the network. The weight update equations are given below

$$w_{ij}(t+1) = w_{ij}(t) + \nu \delta_j x_j \tag{2}$$

and

$$\delta_j = y_j(1-y_j)(\delta_j - y_j) \tag{3a}$$

for the top layer and

$$\delta_j = x_j(1-x_j)\Sigma_k \delta_k w_{jk} \tag{3b}$$

for the other layers. The learning algorithm was used with a variable ν factor, assigned to each layer. Initial ν values of 0.5 and 1.5 were chosen for the input and output layers respectively, to enable a rapid move towards a solution. After every 1000 iterations ν was decreased in steps of 0.26 and 0.51 until a further change would result in a negative value for these parameters, after which ν was held fixed. Learning of these weights for the second to third layer weights was achieved in 1500 iterations using a set of 250 14×14 training image samples. This corresponded to using an 8×8 field of output neurons in the third layer.

Once trained, the classification network was used in conjunction within the feature extraction network to classify the image in Fig. 2(a). The over all network is shown in Fig. 3. The performance of the classification network was compared with that of a simple algorithm which attempted to segment the image based solely on the average number of features within a given neighborhood. This comparison was done to illustrate that the spatial information stored by the neural classifier provided superior classification over methods using only first order statistics. The comparison algorithm assigned to each point a textural label determined from a maximum feature count. The maximum feature count gave the feature type with a maximum value at the most number of points within the sub-region centered about the labeled point.

Results and Discussion of Image Segmentation The results of the image segmentation on the image of Fig. 2(a), using the neural classifier and the maximum counting algorithm, are presented in Fig. 2(c) and (d) respectively. It is seen that the neural classifier performed an almost perfect segmentation with minimal error occurring at the region boundaries. This success can be attributed to the neural classifier being able to make use of spatial information on the spatial distribution of the features over the image plane. This information is necessary to discriminate between the regions in the top right and left of the image in Fig. 2(a), since the right side region contains a large amount of vertically oriented lines. Hence, it is subject to misclassification as a vertical region. This problem can be solved by using smaller neighborhoods in the maximum count algorithm. However, this imposes additional problems in distinguishing the 45 degree and horizontal region borders. To achieve better classification with the maximum count algorithm it is therefore necessary to use additional features from the

lower layer of the feature network. The neural classifier achieves a better segmentation with fewer features. However, this improvement is not free since additional time is required to train the network. When the image to be segmented is quite large and complex, the time required to train the classifier is small compared to the actual segmentation process. In such applications the network approach is definitely advantageous. Also, the neural network segmentation approach requires less user interaction which is desirable in an computer driven system.

Conclusion

A neural network was presented which was able to successfully detect the presence of lines and spots against a background. The hierarchal nature of the network allowed the various layers to be independently designed and trained. This approach was found to be useful in the developmental stages of the network. The network was able to encode orientations other than the tuned orientation by making use of partial responses. Using additional layers, it was shown how a neural network could be trained to successfully classify and segment an image based on its texture.

The features used in this paper are by no means complete and work is currently being done on extending the feature set. The application of more complex neural structures and models for the task of textural analysis are also being investigated.

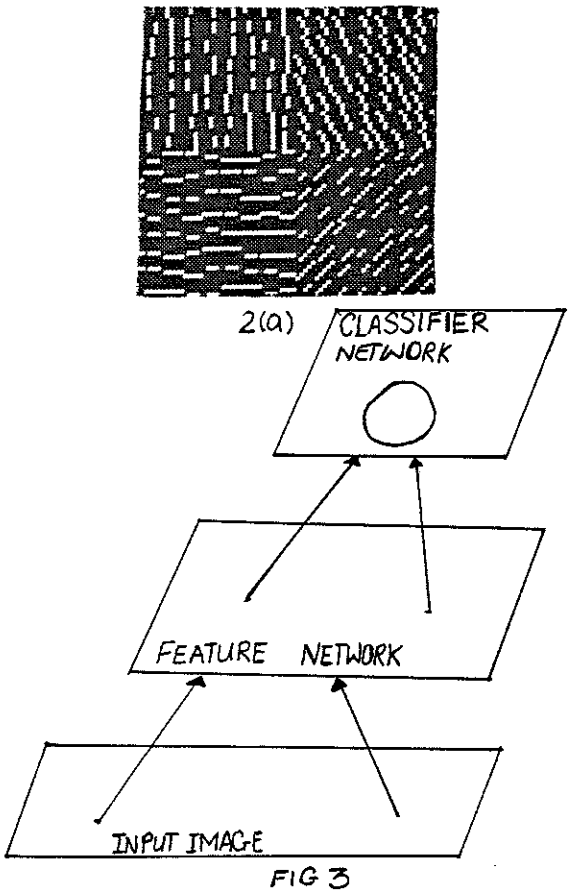


FIG 3

References

[1] Robert M. Haralick, "Statistical and Structural Approaches to Texture," *Proc. IEEE, Vol.67, NO. 5, May 1979.*

[3] Margaret Livingstone and David Hubel, "Segregation of Form, Color, Movement, and Depth: Anatomy, Physiology, and Perception," *Science, Vol. 240, NO. 6, May 1988.*

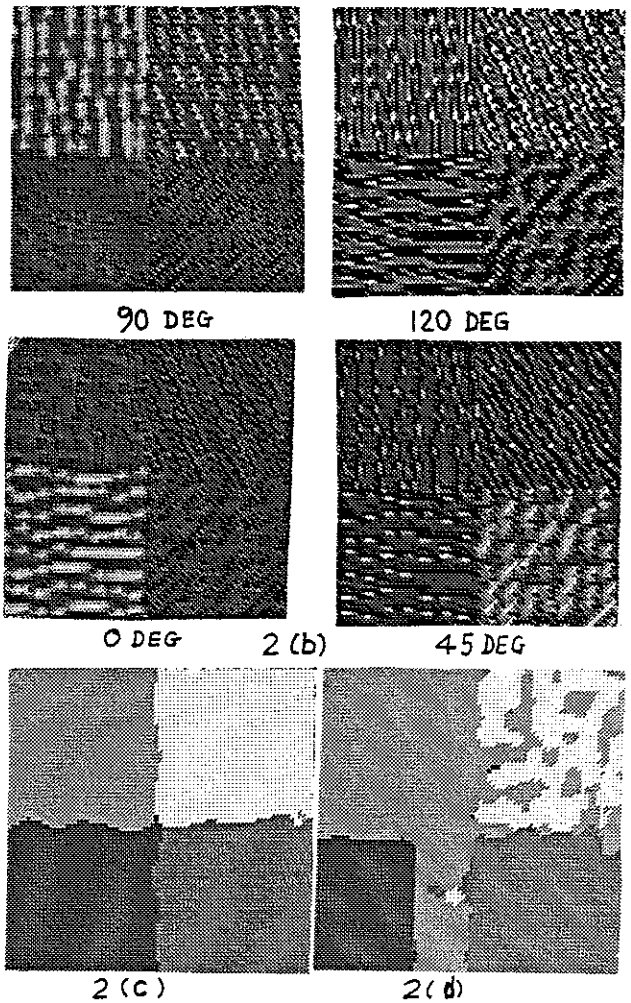
[4] Tara C. Callaghan, Maria I. Lasaga, and W. R. Garner, " Visual Texture Segregation Based on Orientation and Hue," *Perception and Psychophysics, Vol. 39 (1), pgs. 32-38, 1986.*

[5] Bela Julesz, "Textons, the Elements of Texture Perception, and Their Interactions," *Nature, Vol. 290, 12, March 1981.*

[6] B. Julesz and J. R. Bergen, "Textons, the Fundamental Elements in Preattentative Vision and Perception of Textures," *The Bell System Technical Journal, Vol. 62, NO. 6, July-August 1983.*

[7] Rafael C. Gonzalez and Paul Wintz, "Digital Image Processing," *Addison-Wesley, 1979.*

[8] Richard P. lippmann, "An introduction to Computing with Neural Nets," *IEEE ASSP Magazine, vol. 4 NO. 2 April 1987.*



AN EXPERT SYSTEM FOR COMPUTER AIDED DESIGN

UTILIZING A PATTERN RECOGNITION INTERPRETATION OF IMPLICATIONS.*

Dan E. Tamir, Daniel G. Schwartz, and Abraham Kandel
Department of Computer Science and the
SUS Center for Artificial Intelligence
Florida State University
Tallahassee, Florida 32306-4019, U.S.A.

ABSTRACT

A rule based expert system for computer aided design of man machine interface is implemented. The system applies a pattern recognition interpretation of a generalized one variable implication with linguistic variables (Zadeh 1975, Schwartz 1988).

The primary goal of the computer aided design system is to assist a human designer (typically an engineer) in the process of designing a controller to a process. The system receives general information about the expected interface operation and uses the information to design and allocate the visual tactile and audio controls and displays of the interface.

Introduction

In this paper we describe an expert system based on a pattern recognition interpretation for a generalized one variable implication. In this system, knowledge is represented by implication clauses, and the inference rule is a modified modus ponens. The syntax of an implication is: $P_0, P_1, ..., P_{n-1} \rightarrow P_n$, where
P_i $i = 0, 1, ..., n$ are logical (generally multi-valued) propositions.

The meaning of the assumed inference rule is as follows:
P_n is valid if and only if all the propositions
P_i, $i = 0, 1, ..., n-1$, are valid.

A class of decision procedures can be modeled by implications as described above, where P_i contains two types of variables and assumes the form: x is τ_i.

* This Work was Supported by the Office of Naval Research, Grant Number N00014-87-G-0219.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

The first type of variable, x, ranges over items of a set X and the decision process is performed on these items. The second type of variable is the linguistic variable τ_i. Informally a linguistic variable can assume values from expressions in natural or formal languages, for example, expressions such as "very stressed", "more or less experienced", or "creative".

1.1 Linguistic Variables

A linguistic variable Λ is characterized by a triple (T, U, M) where T is a set of linguistic terms ,U is a possibly empty set of measurements, and M is a function from T to U. The components of a linguistic variable are described as follows:

The set T contains the allowable linguistic values of the linguistic variable Λ. In the original definition (Zadeh 1975), the set T could be of infinite rank. This imposes a practical difficulty on the attempt to automate procedures which involve linguistic variables. For practical considerations Schwartz (1988) proposes to severly restrict the rank of T, and develops several plausible forms for T.

In this paper the following form is assumed: For each Λ the corresponding T contains a primary term λ. For example, the primary term of the variable 'Height' can be 'Tall'. Let ant(λ) denote the antonym of λ and let med(λ) represent the concept of an intermediate term. For convenience ml is used as an abbreviation for the linguistic term more or less and v is used as an abbreviation for the linguistic term very. The entire set T, is:
{ v-ant(λ), ant(λ), ml-ant(λ), med(λ), ml-λ, λ, v-λ }.
It is assumed that the terms in T are ordered by a relation $\leq$ in the manner shown.

Under these restrictions, the entire term-set of any linguistic variable is limited to seven grades of distinction. While this might appear to be an oversimplification, it may be argued on practical grounds that in natural language discourse one seldom needs more than seven grades of distinction and linguistic terms which do not reduce to elementary terms. On the other hand, this interpretation is computationally economical, thus it can be implemented in automated reasoning systems.

The set U, contains a range of possible results of measurements of an attribute A in a set of individuals X.

As in Schwartz (1989), three basic ways of forming the functiom M from T to U are allowed. For $\tau \in T$, $M(\tau)$ is either: (1) an interval in U, (2) a probability distribution in U, (3) a possibility distribution in U.

In the decision process the function M can be used to ascribe an attribute $\tau \in T$ to an individual $x \in X$, given some measurement $u \in U$ for x.

Some linguistic variables ('Motivation', 'Kindness', etc.) do not have a natural numerical measurement scale. In this case the common practice is to assume that the linguistic variable Λ is mapped into an artificial measurement scale, and this scale constitutes U. For example, IQ test scores may constitute the set U for the linguistic variable 'Intelligence'. We propose another approach, where in this case U is considered to be empty, hence M is undefined, and operations such as inference are performed directly on the linguistic terms, without appeal to their underlying meaning.

Thus for every individual x, an attribute of x with respect to a variable Λ is specified either in a numerical way, as denoting a measurement $u \in U_\Lambda$, or, if $U_\Lambda = \Phi$, then that attribute is given directly as an item of T. Moreover, M is restricted to be a one to one function from T onto U. Thus given an item $u \in U_\Lambda$, the function M^{-1} maps u onto $\tau \in T$, that is $M^{-1}(u) = \tau \in T$. Hence, given a measurement of an attribute of an individual $x \in X$, if this measurement relates to a linguistic variable Λ, then one can use M_Λ, to determine $\tau \in T_\Lambda$

1.2 Pattern Recognition Interpretation of Implications

Consider the implication $P_0, P_1, ..., P_{n-1} \rightarrow P_n$ as described above. This form of implication can be interpreted as classification of the objects $x \in X$ to equivalent classes. Thus a decision procedure is reinterpreted as a classification procedure. The classification is performed with accordance to the validity of the propositions P_i, and the different classes comply with different truth values of the proposition P_n.

The decision theoretic method of pattern recognition is basically a method of classification of patterns, hence it is possible to interpret the implication clause as recognition of patterns. Actually an implication clause is interpreted as a clustering operation, on the objects $x \in X$.

Definition: Let x be an individual and let $\alpha \in T_\Lambda$ be the actual attribute of x with respect to the linguistic variable Λ, then $\forall \tau \in T_\Lambda$, the truth value of the proposition "x is τ", is considered to be α.

As a result of the model of linguistic variables assumed, the P_i's, can accept one of the seven truth values from v-ant(λ) to v-λ. Actually, in order to enable arithmetical formulation of the decision function, terms such as v-ant(λ), v-λ, are replaced by the numbers 0 to 6, according to their order. A function ρ which is a one to one function from values of linguistic variables to the integer interval [0,6], is defined in the following manner: Let Λ be a name of a linguistic variable, and λ be the primary linguistic term in T_Λ - the term-set of Λ, as defined in Section 1.1.
The following terms receive the following values:
ρ(v-ant(λ)) $\mapsto$ 0, ρ(ant(λ)) $\mapsto$ 1, ρ(ml-ant(λ)) $\mapsto$ 2, ρ(med(λ)) $\mapsto$ 3, ρ(ml-(λ)) $\mapsto$ 4, $\rho(\lambda) \mapsto 5$, and
ρ(v-(λ)) $\mapsto$ 6.

Given the clause $P_0, P_1, ...P_{n-1} \rightarrow P_n$, where the proposition P_i, $i = 0, 1, .., n-1$ is "x is τ_i" and $\tau_i \in T_{\Lambda_i}$. The goal is to represent this implication as a part of a pattern recognition rask. The advantage of the pattern recognition interpretation is that it enables applying well tested methods of clustering, feature extraction and selection and adaptive recognition, in automated reasoning systems. As follows from the decision theoretic approach to pattern recognition two stages should be considered: learning and classification. Where the learning stage consists of feature extraction and selection, partitioning of the feature space, and determination of a decision function. The classification stage consists of feature extraction, and application of the decision function. Generally, the learning and the classification are performed on two distinct sets; let these sets be X and Y respectively.

Learning:

1) Feature extraction: Let X be a set of individuals. The item $x \in X$ denotes a pattern, and Λ_i the linguistic variable in the proposition P_i, denotes the i's feature of x, let this feature be f_i. Each pattern x is represented as a point x' in an N-dimensional feature space F^n, where each axis complies with one feature (one linguistic variable), and has the coordinates 0 to 6, i.e.,
$F = f_0 \times f_1, ..., \times f_{n-1}$. The i's coordinates of a point x' in the feature space F^n is the numerical equivalent of the truth value of the proposition P_i.

For example, suppose that the attribute of x with respect to Λ_i is $\alpha \in T_{\lambda_i}$. This implies that the truth value of the proposition "x is τ_i" ($\tau_i \in T_{\Lambda_i}$) is α. Hence, the numerical equivalent is $\rho(\alpha)$, and the i's coordinate of x' is $\rho(\alpha)$.

The result of representing all the items $x \in X$ in the feature space is a set of points, let this set be X'.

2) Partitioning of the feature space: the feature space is divided into seven regions and each region is associated with a value of P_n. Let these regions be
$R = R_0, R_1, ..., R_6$.

3) Determination of a decision function: a decision function $d : F^n \rightarrow R$ is defined. This function maps patterns to decision regions in the feature space. While an arbitrary selection of regions and decisions functions is allowed, there is one restriction which is stated as the following: assuming that the implication clause is given by: x is τ_0, x is τ_1,...,x is $\tau_{n-1} \rightarrow x$ is τ_n. Let $\rho(\tau)$ be the numerical value that correspond to τ, the function d is restricted to map points in the neighbourhood of the point $(\rho(\tau_0), \rho(\tau_1), ..., \rho(\tau_{n-1})$ to the region $R_{\rho(\tau_n)}$. Note that this applies that up to seven implication clauses can be mapped to the same pattern recognition task.

Classification:

At this stage the decision function d is applied on members of a set Y where generally $Y \neq X$, and each $y \in Y$ is classified into one of the 7 decision regions.

1) Feature extraction: Let y be a pattern which has to be recognized. The values of each of the features of y are measured or evaluated, and thus y is represented as a point in the feature space, let y' denote this point.

2) Application of the decision function: The value of $d(y')$ is computed, and based on this value y' is mapped into a decision region.

It is possible to extend the pattern recognition interpretation so that it is applied directly to the decision process, before this process has been formulated in terms of implication clauses. The new idea here is that the interpretation can be applied to decision process that involve linguistic variables, while traditionally pattern recognition is applied to decision process's which assumes only numerical parameters.

The pattern recognition interpretation of decision procedures (where the decision procedure is not formulated by implication clauses) is as the following:

Learning:

1) Feature extraction: Let X be the set on which the learning process is performed and let D denote the set of different decisions that can be accepted. It is assumed that $|D| \leq 7$ where $|D|$ is the cardinality of D. By inspection of the decision problem a set of features of the items $x \in X$ which is relevant to the decisions D is identified. This set can be large, and generally, different features in the feature set has different importance in the decision process. Moreover, some of the features may relate to linguistic variables that do not have a naturally corresponding measurement scale.

2) Transformation to numerical representation: Each feature is treated as denoting one linguistic variable. Let the i's feature be f_i and suppose that this feature complies with the linguistic variable Λ_i. Using the model of linguistic variables as presented in Section 1.1, the term set T, of the linguistic variable Λ_i is identified. The items of T denotes the allowable linguistic values of Λ.

3) Feature selection: The stage of feature selection is optional, however, if the set of extracted features contains nonmanageable amount of items, or seems to contain redundancies, then feature selection is an essential stage.
Let RF denote the set of relevant features, and let $|RF| = M$. In this stage, a subset $EF \subseteq RF$ is selected and thus $|EF| \leq |RF|$. The sub-set EF is the most efficient sub-set of features for the decision problem according to a given efficiency criteria. After selecting the efficient features the recognition is continued in the space of the efficient features, that is, in EF.

4) Partitioning of the feature space: the feature space is divided into seven regions $R = \{R_0, ..., R_6\}$. Each region complies with one of the possible decisions, and generally the points in a region represent individuals with attributes that complies with that decision.

Stages 3, and 4, are implemented using any method of pattern recognition, e.g., clustering analysis, dynamic programming etc.

5) Determination of a decision function: A function $d : EF \rightarrow R$, is defined.

Form this point the decision process is continued as described above, i.e., as in the case where the decision procedure is already formulated in the form of implication clauses.

2. An expert system for man machine interface

We design and implement a C based expert system shell which utilizes the pattern recognition interpretation of implications and applied this shell in a computer aided design system. The core of the computer aided design system is a cooperative expert system composed of two expert systems where the first emulates an electrical engineer and the second emulates an ergonomics expert.

The Internal Structure of The system

The main components of each expert system are:
1) the knowledge base which is a set of implication clauses, 2) the fact base, 3) the inference rule, and
4) the inference engine.

Implication Clauses:

The knowledge-base of each expert is a set of implication clauses of the form:
P_0 'prompt'$_0$, P_1 'prompt'$_1$, ..., P_{n-1} 'prompt'$_{n-1}$ $\rightarrow$ P_n."explanation",
where: 1) 'prompt' is an optional string of characters which can be used to prompt the user to input a value for a linguistic variable,

2) "explanation" is an optional string of characters which can be used to explain the decision process to the user,

3) P_i $i = 0, ..., n-1$ is of the form $\omega_i x$ is τ_i, where ω is the weight of the preposition ($\omega \geq 0$), x is an individual and τ_i is a linguistic variable,

4) P_n is either of the form "x is τ_n" or it is a name of a predefined C function.

An implication is interpreted as a part of a pattern recognition task. Following the pattern recognition interpretation, the set of implications that constitutes the knowledge base determines the feature spaces and the decision regions of each recognition task.

A decision function $d : F^n \rightarrow R$ is defined. This function maps patterns (points in the feature space), to decision regions and is defined as the following:

a) assumptions: Let $E_l = d(y', C_l)$ where, $y' \in F^n$, is a point in the n-dimensional feature space (F^n), C_l, is the center of the l's region of decision. and, $d(\alpha, \beta)$ is the weighted Euclidean distance between α and β, defined to be:
$d(\alpha, \beta) = \sqrt{\sum_{f \in F^n} [\omega_f \times (\alpha_f - \beta_f)^2]}$.
And let j be the index such that $E = \overset{Min}{j} E_j, j = 0..6$.

b) Decision: If $E = E_j$, then assign y' to the region R_j. If $E = E_k = E_j$ and $k > j$, then assign y' to the region R_k.
Note that this applies that up to seven implication clauses can be mapped to the same pattern recognition task.

The fact base:

The fact base is composed of the initial fact base and the final fact base. Syntactically, the initial and the final fact base are optional files that contain a list of items of the form: x is τ. The initial fact base is used to supply initial values, to some of the propositions in the implications and the final fact base contains the final truth value of all the propositions that appear in the knowledge base.

The fact base is used to implement a schema of cooperative expert system where expert system A, operates, creates a final fact base and transmits it to the expert system B as the initial fact base of B.

The Inference Rule:

The system is using one inference rule which is a multiple premise single conclusion modus ponens. Semantically, The inference rule is the classification part of the pattern recognition task, where the decision function is applied on patterns, classifies them to different decision regions and assigns a truth value to the conclusion of the implications that corresponds to that task.

The Inference Engine:

In these expert systems, the construction of the knowledge base is performed using the C program rule_to_C.c, and the inference is done by a shell which is another C program (shell.X.c), yet this program is automatically created by rule_to_C.c.

Phase 1 - Construction of the knowledge base and the inference engine:

The program rule_to_C.c accepts text written in the syntax of the implications converts it to a set of C function, each function represents one recognition task, and creates the following tables:
1) an inference table, 2) a recognition table, 3) an explanation table.

Phase 2 - Inference:

The shell, generated by rule_to_C.c is thus a main program along with a set of functions, three tables, and a stack that contains names of tasks. The shell is capable of performing inference, and optionally write an explanation of the inference process into a file. The shell receives as parameters the names of two files the first is assumed to contain the initial fact base and the second file is created by the inference engin and contains the final fact base.

Truth values are assigned to propositions that appear in the initial fact base, and the task stack is initialized to contain the list of the recognition tasks which can be executed. The inference is done by a loop of fetch and execute of these tasks.

When task$_j$ is executed it implements a pattern recognition procedure and as the result of the recognition it updates the truth value of the conclusion or activates a C function. The information for the pattern recognition task (i.e, the feature space and the regions of decisions for each task), is stored in the recognition table.

In addition to updating the truth value of the conclusion, after being fired, task$_j$ can return an entry to the task table, this entry contains the names of all the tasks that are affected by task$_j$. The name of these tasks is being pushed into the task stack.

The execution ends when the stack is empty, or if n tasks were fired with out updating the truth value of any proposition. In the EXPLAIN mode the inference engine perform all the previous operations, yet when a task is fired it can return an entry to the explain-table as well as an entry to the task table. The names of that task and the name of the tasks that were fired by it, as well as the contents of the entry in the explain-table returned by that task are appended into a file and the user can view this file using a read only editor.

The Conceptual Component of the System

The primary goal of the system is to assist a human designer in the process of designing a controller to a specific process.

This application is selected since it has some important aspects, the main of them are:
1) The problem of "intelligent" support for a computer aided design system is of great interest.
2) Man machine interface is an essential issue.
3) The design of man machine interface is a process

which involves knowledge from some domains of expertise such as ergonomics, electrical engineering, mechanical engineering, etc. Hence this application can be used, to study other aspects of knowledge representation such as cooperation between experts, communication of knowledge, and standards of knowledge representation.

The objective of the user of the computer aided design system is to design a controller to a process. One of the sub-objectives of the user is to design the man machine interface between the controller and the operator. Currently the focus is set on the design of the audio portion of the man machine interface.

Each interface operation is represented as a signal, it is considered to be an input signal if it is a control operation performed by the operator, (thus the controller receives input signals from the operator), it is considered to be an output signal if it is displaying some information on the state of the controlled process to the operator (thus the operator receives output signals from the controller).

The computer aided design system receives general information about signals passed from the operator to the controller (input signals) and from the controller to the operator (output signals). For example, the information that the system receives can be: 1) The level of the ambient noise, 2) The amount and the rate of information which flows from the operator to the controller and from the controller to the operator, 3) The expected degree of stress of the operator.

The system goal is to find the best way to implement input signals by controls, and output signals by displays, where a control can be either a tactile or an audio control, and display can be a visual display or an audio display.

The cooperative expert system is the core of the computer aided design system it uses the information received from the user to design and allocate the visual tactile and audio controls and displays of the interface. Moreover, the system proposes an implementation to the audio part of the interface.

The expert system has to examine some branches of a decision tree for each given interface operation. For example, one of the nodes in the decision tree is the decision whether certain operation of the operator should be performed by a tactile control or by an audio control. Another node is the decision whether to display information by visual display or by audio display and another node is the node of decision whether to display certain audio information by a tonal display, e.g., alarm or by a vocal display, e.g., by a pre-recorded message.

Following the pattern recognition approach, each node in the decision tree is interpreted as a pattern recognition (classification) task, thus one of the tasks of the system is to classify the input signal (passed from the operator to the controller) and the output signals (passed from the controller to the operator) to tactile visual or audio controls and displays, and to recognize which of these classes is the best for a given scenario. Another task is to classify audio displays to tonal and vocal messages and to determine (recognize) what kind of audio display best fits a specific case.

The following is a list of the main recognition tasks of the system:
1) Classification of signals to displays and controls.
2) Classification of displays to visual displays, and audio displays.
3) Classification of controls to tactile controls, and audio controls.
4) Classification of audio displays to vocal, and tonal messages.
5) Classification of audio controls to automatic speech recognition systems, and voice operated switches.
6) Classification of tonal systems.
7) Implementation of an automatic speech recognition system.
8) Implementation of a tonal system.

Note that although some of the nodes in the decision tree are called classification nodes, and others are called implementation nodes, both denote a decision process, and are interpreted as a pattern recognition task. The difference is that the implementation nodes are "leafs" in the decision tree, i.e., the decision process is always terminated in an implementation decision, On the other hand, classification nodes are branches of the decision tree, and can lead either to classification nodes or to implementation nodes.

REFERENCES

Schwartz D.G. and Lee M.J. 1988, " A Scheme for Logical Inference with Linguistic Attributes", *Proceedings of the first Florida Artificial Intelligence Research Symposium, FLAIRS-88*, Orlando, Florida, (May), pp. 208 - 212.

Schwartz D.G. 1989, " A New Approach to Representing Imprecise Linguistic Inference", *Proceedings of the second Florida Artificial Intelligence Research Symposium, FLAIRS-89*, Orlando, Florida, (this volume).

Zadeh, L. A. 1975, "The Concept of a Linguistic Variable and its Application to Approximate Reasoning", Part I: *Inf. Sci. 8*, pp. 199-249; Part II: *Inf. Sci. 8*, pp. 301-357; Part III: *Inf. Sci. 8*, pp. 9 43-80.

AN ONCOLOGICAL EXPERT SYSTEM FOR GYNECOLOGY

Isabel Stabile and Ladislav J. Kohout
Center for Biomedical and Toxicological
Research and Department of Computer Science,
Florida State University,
Tallahassee, Florida, 32306
and John Anderson
King's College Medical and Dental School,
University of London, U.K.

ABSTRACT

The expert system with "deep knowledge" discussed in this paper is aimed at the identification of women at risk from ovarian cancer, based on their past and present medical history since no reliable clinical or laboratory screening tests are as yet available.

In analyzing the past and present medical history of asymptomatic women, it may be possible to identify factors, which though insignificant on their own, may, if present in some particular specific combination, be highly significant risk indicators for cancer. The knowledge of these risk factors (to be elicited from the literature) has to be embedded as the "deep knowledge" into the system. Such a knowledge based system can scrutinize context dependent combinations of many factors and hence significantly contribute to the identification of women at risk from ovarian cancer. A Knowledge based system, as described here, could be of considerable help in clinical practice, firstly because the identification of highly complex combinations of many, in some cases insignificant factors, is time consuming; in order to link the factors appropriately and explore comprehensively all the possible correlations between them a computer is essential. Secondly, not all gynecologists are experts in oncology. In the at-risk patient, the unusual significance of some risk factors may not immediately alert the non specialist in oncology and any delay in diagnosis may reduce the chances of cure.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

STATEMENT OF THE PROBLEM AND REQUIREMENT ANALYSIS FOR THE DESIGN OF THE SYSTEM.

Medical Aspects Of The Problem: Ovarian Cancer

Ovarian cancer is a disease of western nations and is 3 to 5 times more common in industrialized populations than in developing countries. The age adjusted annual incidence rate in the USA is 10 per 100,000 women compared with 3.5 for Spain (Green, Clark and Blayney 1984).

Despite developments in the management of ovarian cancer, there are no prospects for cure in advanced stage disease. The one area of hope for progress is the possibility of early diagnosis. Support for the concept that ovarian cancer has a strong environmental and/or cultural component comes from observations on high- and low-incidence populations in various parts of the world. The incidence rate per 100,000 per year in the US ranges from 10-13, whereas in Japan the rate is 2.2. It is noteworthy that first generation Japanese immigrants to the US have an increased risk of developing ovarian cancer and that these immigrant populations ultimately develop ovarian cancer at a rate approaching that of their host country.

The incidence of ovarian cancer in the US is increasing. It is a disease which presents late, approximately 75% of cases having Stage III or Stage IV disease at the initial operation

(Kottmeier 1982). This combination of late presentation and poor response to treatment accounts for the fact that, although less common than endometrial and cervical cancer, ovarian cancer causes more deaths than both combined, and has a diagnosis to death ratio of 1.6:1 (American Cancer Society 1982).

Although this appalling background cannot be disputed, there is evidence to suggest that the pattern of the disease can be changed and survival figures improved. In spite of these improvements (detailed below), women still suffer unnecessarily because established knowledge has not been transformed into established practice. The opportunity for improvement exists in several areas:

1. Earlier diagnosis. This involves greater awareness of the at-risk woman, prompt investigation of suspicious symptoms and further development of screening techniques. No reliable clinical or laboratory screening tests for ovarian cancer are as yet available, radioimmune scanning being largely experimental while ultrasound and peritoneal cytology are not of proven value. In addition, although preliminary data suggest that CA 125 may be of value as a tumor marker (Jacobs et al. 1987), it may not possess sufficient specificity to be useful in screening asymptomatic women.

2. Prevention. This involves removal of both ovaries at surgery for benign disease, particularly in women with a strong family history of ovarian cancer.

3. Improved use of existing therapeutic techniques, by accurate staging, maximal surgical effort and optimal adjuvant therapy.

4. Further development of experimental therapies.

Our work addresses the first of these four options with the specific aims of identification of the patients likely to be at risk and early diagnosis of ovarian cancer in those high-risk patients.

TECHNOLOGICAL ASPECTS OF THE PROBLEM

The Use Of Appropriate Knowledge Engineering Techniques And Expert Systems Architecture

The specific knowledge domains of this field lead to new technological problems. Although these domains may appear less involved than those in other specialist branches of medicine, the clinical information, decisions and actions are rather complex and not easily amenable to formalizing by rule-based knowledge representation methods. In addition, dealing with individual data items and pieces of knowledge is strongly context dependent. Context dependency of this knowledge can be expressed adequately only within a wider framework of deep knowledge characterizing not only the domain of Gynecology, but also including some deeper foundational aspects of the interaction between Medicine and Surgery.

Our analysis of the problem has shown that a gynecological oncological knowledge-based system should have the following features:

1) The system must be capable of dealing with a multiplicity of contexts. In the identification and interpretation of these contexts a deep knowledge framework is essential.
2) It must also utilize the deep knowledge concerning cancer risk factors which may involve representation and utilization of possibilistic and fuzzy uncertainty measures.
3) The patient's signs, symptoms and syndromes must be interpreted in the context of the patient's present history. This may involve deep knowledge from other medical specialties.
4) The system must also utilize the patient's past history (medical, family, etc.) in its inferential process.

Correct interpretation of patient histories (both past and present) necessarily involves the deep knowledge concerning the conceptual and logical nature of time dependency (Kohout et al. 1989e).

THE METHOD OF ATTACK: NEW CLINAID KNOWLEDGE BASED SYSTEM ARCHITECTURES

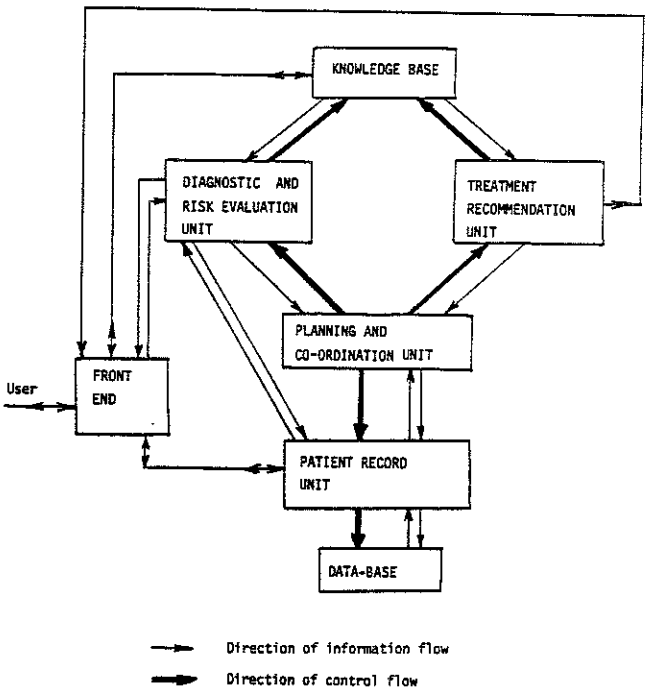
Our novel CLINAID architecture (Anderson et al. 1985; Kohout et al. 1989c), supported by a well-developed knowledge engineering methodology based on Activity Structures (Kohout 1989a), was devised to deal with problems occuring in multi-context and multi-environmental situations (Kohout, Anderson and Bandler 1989b). In a series of papers, we have described the conceptual structures as well as the basic units of CLINAID. Its architecture is aimed at supporting not only diagnosis but also other types of clinical activity and decision making in diverse medical, clinical and/or hospital environments. The aim of CLINAID is to support knowledge-based decision making involving risk, uncertainty and incompleteness of data. Such a system has to operate in a multi-environmental situation and make decisions within a multiplicity of contexts (Kohout, Anderson and Bandler 1989a). We have utilized the Activity Structures approach (Kohout 1987; 1989a) in the design methodology in order to furnish the necessary foundations and framework for our construction.

The first demonstration of the prototype of the CLINAID diagnostic unit was given by Trayner and colleagues in 1983. A recent addition to the CLINAID architecture is the adaptive learning unit (Kohout et al. 1989c), which can monitor changes in the nature of both diagnostic and clinical data, as well as the rate of mis-diagnosis of the system, due to changes in the nature of environments, and other factors. Another recent addition was the design blueprint for DEXTERON, a real-time fuzzy expert system with deep knowledge for the assessment of movement performance of neurological patients by classification of the dynamic features of their handwriting (Kohout and Kallala 1986).

AN OVERVIEW OF THE FUNCTIONS AND ACTIVITIES OF CLINAID

The basic architecture consists of the co-operating units connected to each other by appropriately matched interfaces (Kohout et al. 1985). This Basic Shell Substratum is composed of four Units ie., the Diagnostic and Risk Identification Unit, the Treatment Recommendation Unit, the Patient Clinical Record Unit and the Co-ordination and Planning Unit. The global configuration of CLINAID, including the data and control flow interaction between the units is shown in Figure 1.

Figure 1: The Basic Configuration of CLINAID



Activities Of The Diagnostic Unit

The purpose of this unit is to assist in the diagnosis of diseases. During this activity it accepts the information required in a dynamic manner. It is not connected to the patient but relies on requests to the user (clinician) for as long as it lacks sufficient information to suggest the working diagnosis. "Working Diagnosis" is defined as a set of possible diseases, together with the degree of possibility of appearance for each disease in this set.

During the inference process, the Diagnostic Unit requires some deep knowledge, which is obtained from the Knowledge Base.

When working towards the diagnosis the unit may also require the use of the patient's medical record. All relevant information (e.g. the signs and symptoms, the findings of physical examination, the results of laboratory tests, as well as the conclusion), is communicated to the Patient Clinical Record Unit to be embedded within the context of the patient's present history.

Activities Of The Treatment Recommendation Unit

The purpose of the second unit is to advise about treatment taking into account the working diagnosis, various risk factors, and the possible adverse effects of each element of the class of treatments available (Kohout et al. 1984)

Activities Of The Patient Clinical Record Unit

It is essential to incorporate a clinical record system that keeps track of all the data concerning the patient, because treatment recommendation requires knowledge of prior and current therapy in order to avoid adverse cross-interactions. Also, prognosis of the patient's clinical state requires knowledge of both present and past histories.

Activities Of The Co-ordinator and Planning Unit

This unit provides the overall control and planning of the clinical management activities as well as all the necessary auxillary matching functions such as generating and unifying the
communication protocols.

CONCEPTUAL DESIGN OF KNOWLEDGE STRUCTURES FOR THE ONCOLOGICAL SYSTEM WITH THE DEEP KNOWLEDGE

The deep knowledge elicited from the literature and embedded into the system's knowledge base consists of the following major components:

a) The risk factors linked to the specific and non-specific symptomatology of the individual patient, in particular knowledge of the significant combinations of various risk factors in this category.

b) The risk factors linked to the individual's past history (medical, obstetric, gynecological, environmental, etc).

c) The risk factors linked to the family history (e.g. mother etc).

d) The knowledge structures relating to the significance of coincidences of the risk factor groups (a), (b) and (c) above.

The knowledge structures (a), (b) and (c) are each placed in a separate Diagnostic and Risk Identification Unit. The knowledge structure (d) is located in the Planning and Co-ordination Unit of CLINAID. These units are interfaced to co-operate with the Patient Record Unit of the patient being assessed.

At present, knowledge elicitation from various sources including medical texts is in progress. Because of the diversity of medical sources that are utilized in the knowledge acquisition process, the heterogenous combination of probabilistic, possibilistic and linguistic knowledge is essential. The methods previously developed by the authors are used for this purpose (Kohout et al. 1988).

CONCLUSION

As the identification of cancer risk depends on rare combinations of many common factors, an expert system with the deep knowledge of these combinations could be of considerable help in a busy gynecological practice. In view of the increasing litigation facing practitioners in Florida State the validated system would find commercial application.

A Knowledge-Based System capable of monitoring unusual combinations of various factors and of giving a warning of potential risks may have important applications in manufacturing industries, insurance and reinsurance underwriting (Kohout, Anderson and Bandler 1989b), and elsewhere. Thus the system empty shell and the knowledge elicitation methodology may find significant commercial applications outside gynecological oncology.

REFERENCES

American Cancer Society 1982. "Cancer facts and figures." New York.

Anderson, J.; Kohout, L.J.; Bandler, W. and Trayner, C. 1985. "A knowledge-based clinical decision support system using new techniques: CLINAID." In *Proceedings of the AAMSI Congress* 1985 (San Francisco, Calif. May 20-22). American Association for Medical Systems and Informatics, A.H Levy, B.T Williams, eds, 187-192.

Green, M.H., Clark, J.W., Blayney, D.W. 1984. "The epidemiology of ovarian cancer". *Seminars in Oncology*, 11: 209-226.

Jacobs, I., Stabile, I., Bridges, J., Reynolds, P., Oram, D. and Grudzinskas, J.G. 1988. "A Multimodal approach to screening for ovarian cancer." *Lancet* ii: 268-271.

Kohout, L.J., Keravnou, E., Bandler, W., Trayner, C., and Anderson, J. 1984. "Construction of an expert therapy advisor as a special case of a general systems protection design." In *Cybernetics and Systems Research 2*, R. Trappl, ed. North-Holland, Amsterdam, 97-104.

Kohout, L.J; Bandler, W.; Anderson, J.; and Trayner, C. 1985. "Knowledge based decision support system for use in medicine. In *Computer models for decision making*. G. Mitra. ed. North-Holland, Amsterdam, 348-352.

Kohout, L.J., Kallala, M. 1986. "Evaluator of neurological patients' dexterity based on relational fuzzy products." In *Proceedings of the 2nd Expert Systems International Conference*, Learned Information. Oxford, U.K. and New Jersey, 1-12.

Kohout, L.J. 1987. "Activity structures: a general systems construct for design of technological artifacts." *System and Cybernetics: An International Journal*, 18: 27-34.

Kohout, L.J., Behrooz, A., Anderson, J., Gao, S., Trayner, C. and Bandler, W. 1987. "Dynamics of localised inference and its embedding in activity structures architectures." In *Proceedings of the Second IFSA Congress* (Tokyo, Japan, July, 20-25). IFSA, 2: 740-743.

Kohout, L.J., Ben-Ahmeida, M., Stabile, I., Anderson, J. 1988. "Extracting knowledge from texts." In *Proceedings of the Workshop on Knowledge Based Systems and Models of Logical Reasoning* (Cairo, Egypt Dec 27-31). In Press.

Kohout, L.J. 1989a. *"Perspectives on Intelligent Systems: A Framework for Analysis and Design.""*. Kogan Page, U.K., In Press.

Kohout, L.J., Anderson, J.; and Bandler, W. 1989b. *Multienvironmental Knowledge Based Systems*. Technical Press (Grower), Aldershot, U.K. In Press.

Kohout, L.J., Anderson, J.; Bandler, W., Gao, S., Trayner, C. 1989c ."An Overview of CLINAID" In *Kohout et al, 1989b.*

Kohout, L.J., Behrooz, A. and Anderson, J. 1989d "Problems of knowledge elicitation in insurance". In *Kohout et al, 1989b.*

Kohout, L.J., Anderson, J., Bandler, W., Behrooz, A., Gao, S., Trayner, C. 1989e. "Activity structures-based architectures for knowledge based systems. Part 1: Dynamics of localized fuzzy inference and its interaction with planning. *Fuzzy Sets and Systems*, In Press.

Kottmeier, H. ed. 1982. *Annual Report on the Results of Treatment in Gynaecological Cancer*, FIGO, Stockholm. Vol 18.

Trayner, C., Anderson, J., Bandler W. and Kohout, L.J. 1983. " Public Demonstration of a Diagnostic Unit of CLINAID" *International Workshop on Fuzzy Sets and Knowledge-Based Systems,* (Queen Mary College, University of London, March 1983.)

Expert Systems and Music: Translating Piano Music to Guitar chords

by

William B. Feild Jr., Gisela Fraguio, Jainendra K. Navlakha and Mark A. Weiss

Florida International University

ABSTRACT:

Our implementation of an expert system to translate piano music to guitar chords and fingerings is an effort to expand the rapidly growing domain of computer assisted music. The expert system is a rule based system, implemented using the VAX version of OPS5. Analysis of the expert heuristics led to the discovery of more efficient procedure oriented methods.

1. INTRODUCTION:

The paths of the music world and the computer world have been on a collision course for several years. The available literature suggests that most of the overlap has occurred in the areas of musical interfaces and computer assisted composition. A Musical Instrument Digital Interface (MIDI) allows the user to connect a synthesizer to a computer and analyse/control the timbre of various sounds. Experiments in Music Intelligence (EMI) is a Lisp based system that allows the user to construct 'dictionaries of styles' and use these dictionaries to assist in the composition of music [4].

Another application for computer assisted music is in the area of music translation. It is often useful to convert music written for one instrument into music for another instrument. A particular example is the translation of piano music, which is written to be played by playing notes, into guitar music, which is written to be played by fingering chords.
This particular example is a rather complex one. It not only requires many years of practice and experience in music, but also requires expertise with the piano and guitar specifically. Additionally, the problem seems to be one with many 'correct' answers since most experts would come up with different translations of the same piece of music. However, a high degree of precision is still required since the different translations must still sound essentially the same. For these reasons an expert system would be an excellent approach to solving this problem [9].

The translation process requires several distinct steps. Initially, the piece of piano music must be encoded into a suitable format for access by the expert system and then the music must be read into the system. Next, the translation process begins. A key signature is determined for the piece of music and the notes are read in. The notes are then scanned to remove any extraneous notes. Then the remaining notes are grouped together to form the most suitable chords. These chords are then considered in order to select the easiest sequence of fingerings for the given sequence of chords.

In this paper, we describe a rule based expert system for the translation of piano music to guitar chords and fingerings. We examine the design decisions concerning the representation of the problem and the knowledge base. We summarize the rules included in the system to illustrate the chord and fingering selection algorithms. We also discuss the criteria used to decide when to use the rule based system versus the use of external procedure oriented methods. Finally, we examine the scope of the current system and suggest future extentions. For a non-music oriented reader, Appendix contains a glossary of music terms used in the paper.

2. DESIGN DECISIONS:

The Piano-to-Guitar (P2G) expert system is implemented on a VAX 8800 computer using the VAX version of OPS5 and external support procedures written in Pascal. The language details of OPS5 are described in [5,6].
In the design of any system, one of the first areas examined is the problem

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

resolution strategy. Many expert systems are called upon to solve problems that involve searching from among many possible solutions and coming up with the best (or only) possible solution. In some cases, the problem has a relatively small solution space. For those problems it is easy to implement an exhaustive search through the possible solutions and select the correct one. However, very often the problem has a large solution space and time or system constraints demand a more selective search process.

In our problem, we have a very large number of possible solutions for each of the problem's component subproblems and this dictates the latter approach. The major subproblem entails finding the best match for a sequence of input notes from among many sequences of notes that constitute the various knowledge base (KB) chords. As more input notes are examined, the number of KB chords capable of being 'best match' can be reduced until we have left only a few chords, often only one, for the best match. Based upon this part of the problem, we selected a 'tree' as our model for the solution scheme.

Another subproblem entails finding the best sequence of fingerings to match with the sequence of chords arrived at previously. Each chord has several possible fingerings that could be selected. Each fingering is played at a different string, or fret, on the guitar stem. The general goal is to reduce the amount of movement the guitar player's hand and fingers must make while they are playing. Our initial choice was to reuse the model in the chord selection subproblem since the processes seemed, at first, to be very similiar. The task turned out to be more like traversing a graph with the goal being to minimize the distance traveled between nodes, representing the chord fingerings.

Another important design decision concerns the representation of knowledge in the KB. As in any other data base, one important decision is how much redundacy of data is to be allowed or accepted. Minimizing the physical data kept in the KB would reduce the amount of memory required to maintain it. The trade off is that this requires more computation, in the form of rules, to infer the data that is not explicitely available. For example, the notes that comprise any given chord are explicitely given in our KB. However, given the type and root of a chord, the remaining notes could be computed using a trivial algorithm. In our problem, we choose to have the extra knowledge readily available rather than having to make the computation each and every time we needed to match chords since this computation would be done frequently and would be different for each number of chord tones. In most instances, we choose to have redundant data because memory turned out to be less critical than the overall running time.

3. RULES FOR THE EXPERT SYSTEM:

The next phase in designing the P2G expert system was to examine and implement the algorithm that would take the particular music presented to the system and translate it into acceptable guitar fingerings. This process was broken down into several steps:

Step 1: Read in the 'a priori' Knowledge Base:
The first step accepts the data from data files and establishes an 'a priori' KB for all music that is to be considered [1,2,3]. This 'a priori' KB is restored each time that a new chord is being considered.

Step 2: Determine the key signature:
This step determines the key signature. This is done only during the first measure examined and considers the key that was read and the first note read in. Figure 1 gives the detailed algorithm to determine the key signature.

```
Read in the key input value
Examine the first note of the music
Match the key input value to the input
 value of the key signatures in the KB
Match the first note read to the first
 note of the tonic chord of the key
 signatures in the KB
If there exists a Match among the key
 signatures then
 If key signature type = MAJOR then
  key signature name is MAJOR
                or
 If key signature type = MINOR then
 Match note 6 & note 7 of minor key to
  the notes in the measure
  If Match is on note 6 and note 7
  then key signature is MELODIC MINOR
                or
  If Match is on note 7 only then
  key signature is HARMONIC MINOR
                or
  If Match is not on note 7 at all then
  key signature is NATURAL MINOR
          OR
If there exists no Match among the key
 signatures then key signature is
 MAJOR
```

Algorithm to determine a key signature
Figure - 1

Step 3: Read the music to be translated:
This step begins the actual translation of piano music into guitar chords. It begins by reading in the piece of music to be translated. Notes are read one measure at a time, removing embellishing notes as they are encountered.

```
While not end of music
  While not end of measure
     Read in the notes of the music
          until end of measure
     Perform the removal of all
          embellishing notes
     Perform Step 4.
```

Step 4: Examine the notes to determine chords:
This step determines the chords to be selected. As rules are applied, the KB is pruned to avoid unnecessary searching. The KB is first pruned using knowledge about the notes as they are examined. Once the examination of notes has ceased for the current chord, further pruning of the chords remaining in the KB is done by a heuristic process and continues until only one chord remains.

Beginning with the first note of the measure

{Examine a note}
If the note has already been examined for the current chord then save that note for the next chord and go to {Continue chord purge}

or

If the note is not in at least one chord in the current KB then save that note for the next chord and go to {Continue chord purge}

or

Else purge all chords in the current KB that do not contain that note and go to {Examine a note}

{Continue chord purge}
If the number of notes examined for this chord = 2 and there exists chords in the current KB of type TRIAD and SEVENTH then purge all chords of type SEVENTH

Purge all chords in the current KB that contain notes that Match notes in the key input (note 1 - note 14) for the key signature

If the number of notes examined for this chord > 2 and there exists a chord in the current KB of type SEVENTH and there does not exist chords of type TRIAD then the CHORD is found and go to {Reset KB}

If the number of notes examined for this chord = 3 and there exists a chord in the current KB of type TRIAD then the CHORD is found and go to {Reset KB}

If the number of notes examined for this chord > 1 and there exists chords in the current KB of type TRIAD then the CHORD is found to be one such TRIAD - choose one at random and go to {Reset KB}

If the number of notes examined for this chord = 1 and there exists chords in the current KB of type TRIAD then Match the root note of the chords to the note examined the CHORD is found to be one such TRIAD - choose one at random and go to {Reset KB}

If no CHORD has been found and there exists a chord of type SEVENTH in the current KB then the CHORD is found to be one such SEVENTH - choose one at random and go to {Reset KB}

{Reset KB}
When the CHORD has been found, various lists are updated and the KB is reset to it's 'a priori' state to allow for continuation of the translation of the piano music.

Step 5: Examine the selected chords to determine fingerings:
The final step is the selection of fingerings for the chosen chords. The chords are examined sequentially to determine which fingering for a given chord requires the minimum movement along the frets from the previous chord.

```
Initialize Movable Table with appropriate
   fret values
Initialize Chord Table with chord names
Initialize starting fret position at fret
   5; Read in the chord names from the
   translated music
Beginning with the first chord
While not end of music
   Locate the next chord in the Chord
        Table and record the index
   Locate the fret values in the Movable
        Table based upon the index
   Calculate the difference between each
        fret value and the previous fret
        position
   Select the minimum difference and
        assign the appropriate movable
        fingering from the Movable Table
Endwhile
```

Much of the music theory and related details are beyond the scope of this paper and hence are omitted here. More details on the algorithm and theory behind the above rules can be found in [7].

4. RESULTS OF THE P2G SYSTEM:

The P2G system was implemented using the OPS5 rule based system on VAX computer.

Experiments were run with P2G to determine just how effective the system was. Several different musical pieces were transcribed and presented to the system for translation. Some of the pieces translated include parts of "Eine Klein Nachtmusik", "Scarborough Fair", "Sailor", and "Terms of Endearment". An example portion of the latter is presented. The '|' mark the end of a measure and '...' represent gaps for convenience. This piece had no sharps or flats for the key signature and all notes are in the same or an adjacent octave. Figure 2 contains the original, the intermediate and the translated music.

Original Music - before transcription into data file

```
| E C D C E D C | ... | C B A E E | E A B C C D |...
```

Intermediate Music - after removal of embellishing notes

```
Key signature = C Major
| E C C E B C | ... | C A E | E A C D |...
```

Translated Music - chords and fingerings

```
Am ------  Cmaj7 ------   C ------     Am ------   Am ------ Dm------
   ------        ------     ------        ------     ------   ------
   1--111 5      -1---1 3   -1---1 3 ..   1--111 5   1--111 5 -1---1 5
   ------        ---2--     ------        ------     ------   ----2-
   -34---        --3-4-     --333-        -34---     -34---   --34--
   ------        ------     ------        ------     ------   ------
```

Example of original and translated music
Figure - 2

The P2G system was rated by our music experts to be performing at about a moderate level of expertise. There were no obvious errors detected in any of the trial music we ran, however, as forewarned, the experts came up with slightly different chords and fingerings in some cases.

5. CONCLUSIONS:

Examining the expert heuristics obtained during the knowledge acquisition phase of the system design led to some interesting discoveries. We had obtained a number of rules for how to best obtain the optimal fingering sequence. The rules followed our intuition that the problem was best modeled as a 'tree' and the rules allowed us to traverse the possible fingerings at a given position and select the overall optimal branch. However, careful analysis of the rules revealed that a table-driven algorithm, similiar in intent to the one that would help traverse a graph, not only mimicked the results from the rules but also handled the frequent 'exceptions' that were constantly being added to the KB. This led to the implementation of external Pascal routines to handle these tasks and return the results to the P2G KB.

Upon completion of this phase of the system, we observed that we had essentially created an algorithmic solution to this problem. We believe that the knowledge acquisition step as well as the careful refinement common in any software engineering methodology helped us in identifying a simplier, conventional implementation of a solution to this problem. In other words, it should be feasable and much easier to construct a conventional program to handle this problem by combining methodologies from conventional and expert system engineering.

In the development of some expert systems, certain components of the problem may obviously lend themselves to conventional algorithmic methods and these should be exploited as such in order to improve system performance. We concluded that some (if not all) components, which at first appeared to be only suited for a rule oriented approach, may in fact be conventional algorithms in clever disguise.

6. FUTURE RESEARCH:

Although the P2G expert system performed fairly well, it is still very much in its infancy. P2G has several limitations at present which, when overcome, will enhance the performance and make it a more useful tool.

One such limitation concerns 'styles' of music. It was found that the notes and chords used frequently in jazz music are different than those frequently used in pop music. And both differ from the ones used in classical music. It would be useful to create distinct parts of the KB that would be included once the style of music is given or determined.

Another current limitation is that P2G only considers TRIADS and SEVENTHS. Modifications to the KB and the inclusion of additional rules to handle them will allow more sophisticated chords to be included in the selection process. Also, broader recognition of embellishing notes (i.e. notes of short duration) will further enhance the overall selection process. Finally, during chord selection, further refinement of the rules to assure that no 'random choice' need be made will allow for not only correct chords but also the most desirable.

In its present form, P2G does not consider the base clef. Also, it does not handle cases where the key signature modulates in the middle of a piece. Both of these enhancements will increase the complexity of P2G considerably.

Finally, the rule based approach taken with the P2G expert system is fairly standard. A more radical approach to expert system design involving an entirely new architecture called neural nets is being considered as an alternate method for the knowledge acquisition phase of expert system design.

REFERENCES:

[1] Jay Arnold, 7488 Guitar Chords, Hansen House, January 1984.

[2] Bruce Benward, Music in Theory and Practice, Second edition, Vol 1, 1981.

[3] Bruce Benward, Music in Theory and Practice, Second edition, Vol 2, 1982.

[4] David Cope, Music and Lisp, AI Expert, pp 26-34, March 1988.

[5] VAX OPS5 Reference Manual, Digital Equipment Corporation, September 1985.

[6] VAX OPS5 User's Guide, Digital Equipment Corporation, September 1985.

[7] William B Feild Jr, Gisela Fraguio, Jainendra K Navklakha and Mark Weiss, Smart Computer assisted Music: Translating Piano Music to Guitar Chords, Submitted for publication.

[8] The International Library of Piano Music: Sounds of Our Time, University Society Inc., 1973.

[9] Donald A Waterman, A Guide to Expert Systems, Addison Wesley, 1986.

APPENDIX
GLOSSARY OF MUSIC TERMS.

Chord. A unit of a minimum of three different tones sounding simultaneously.

Embellishing Tones. Any non-chord tones that occur simultaneously with the harmony and in some way enhance one or more chords.

Fingering. The arrangement of fingers upon the guitar strings and frets used to produce the various chords. The amount of movement upon the frets and the amount of movement of the fingers is ideally minimized.

Flat. An accidental that lowers the pitch one half step (e.g. 'b').

Fret. A guitar has between sixteen and twenty-four frets. Pressing a string against a fret shortens the string and therefore changes the pitch. A played, but unfretted, string is called an open string.

Interval. The difference in the pitch between two notes. One half step is the smallest interval and is used to denote two adjacent notes, i.e. C - C# is one half step, C - D is a whole step.

Key. A system of tones related to one central tone (tonic).

Key Signature. An arrangement of necessary sharps or flats which appear at the beginning of a composition.

Measure. A series of notes delineated by a bar signifying the end of one measure and the beginning of a new measure.

Movable Chord. A chord played with no open strings is a movable chord. Playing the same formation at different frets produces different chords, i.e. A chord formation at fret 0 represents an E Major chord while the same formation played at fret 1 is F Major.

Note. A tone to which a letter is associated (A B C D E F G).

Octave. Since the pitch spectrum is so wide, notes are identified by the octave they are in. Each octave consists of the entire scale.

Root. The chord root is the lowest pitch of the chord and is used to name the chord.

Scale. A series of ascending and descending pitches. The entire scale consists of: C C# D D# E F F# G G# A A# B (Db = C#, Eb = D#, Fb = E, Gb = F#, Ab = G#, Bb = A#).

Seventh. A four note chord (root, third, fifth, seventh): Major-Major is a Major triad with a major seventh; Major-Minor is a major triad with a minor seventh; Minor-Minor is a minor triad with a minor seventh; Minor-Major is a minor triad with a major seventh; Dim-Minor is a diminshed triad with a minor seventh.

Sharp. An accidental that raises the pitch one half step (e.g. '#').

String. A standard guitar has six strings, each of a different thickness, with the tension giving different notes.

Tone. A musical sound of definite pitch.

Tonic. The note which comprises the first scale note - tonal center.

Triad. A three note chord (root, third, fifth): Major Triad has a major third and a perfect fifth; Minor Triad has a minor third and a perfect fifth; Diminished Triad has a minor third and a diminished fifth; Augmented Triad has a major third and an augmented fifth.

REMOTE MAINTENANCE MONITORING SYSTEM

Martin J. Lougheed, NASA

Donn Rochette, GTSI
TE-LPS-13
Kennedy Space Center, FL
32899

In its current configuration, the Launch Processing System (LPS) at the Kennedy Space Center (KSC) relies on over 200 ModComp Computers to perform launch critical monitor and control functions between the space shuttle and Ground Support Equipment (GSE). While hard failures seldom present troubleshooting challenges, operational ModComp computer history shows that 17.4% of the intermittent failures are never found and cannot be duplicated in a trouble shooting environment. Due to the system nature, maintenance of these 200 computers is performed in an off-line environment, thus making the task of troubleshooting and repairing the system both manpower and time intensive. When fully integrated into the LPS, the Remote Maintenance Monitor System (RMMS) will function as an automatic maintenance diagnostics system capable of anticipating approaching failures, and troubleshooting both hard and intermittent failures down to the Line Replaceable Unit (LRU) level while expending minimum time and manpower.

Spanning a five year development life cycle, the Remote Maintenance Monitoring System program schedule includes the development of the RMMS Phase I prototype, RMMS Phase II design and specification work, and RMMS Phase III demonstration and concept validation.

The Phase I prototype focused on the development of a "proof-of-concept" evaluation unit which demonstrated the ability to implant data acquisition hardware into the ModComp computer. Various operating parameters are monitored for error conditions, and automatically diagnose the cause of the failure using the Artificial Intelligence built into a knowledge based maintenance diagnostics expert system.

Following the successful completion of Phase I, the start of Phase II represented the beginning of full scale design and development toward the goal of producing a deliverable maintenance monitor. In its final configuration the maintenance monitor will be able to capture ModComp data in real-time and subsequently use this information to anticipate failures, diagnose irregular or spurious system conditions, automatically analyze ModComp memory dump information to determine the source or sources of failure, and provide system maintenance engineers with an interface for performing on-line background testing of ModComp parameters and processes. Design and development efforts required during Phase II include the specification of a dedicated LISP processor to host the Remote Maintenance Monitoring System Expert System, and Expert System Shell to assist in the development of expert system software, and a standalone workstation to provide communication and control between the various modules within the entire system.

The Remote Maintenance Monitoring System architecture in Phase II (please see Figure 1) consists of a Data Manager, Data Analyst, Logged Data Monitor (LDM), ModComp CPU implant, Common Data Buffer implant and a Communication Network. The Data Manager serves as the interface between the implants and the Expert Diagnostic System, as well as the system archiver. The Data Analyst controls the implants and serves as a workstation for diagnosing ModComp failures. The Logged Data Monitor captures ModComp system messages and data dumps from the Launch Processing System data communications network. The ModComp CPU implant monitors and evaluates ModComp memory and register accesses, CPU/I/O processor handshaking operations, I/O processor microcode execution, peripheral transactions and all regulated voltages. The implant also provides failure mode information through real-time evaluation of register and memory values. The Common Data Buffer implant detects and captures errors in the Launch Processing System network controller subsystem. The Communication Network links the CPU implants and the Data Manager.

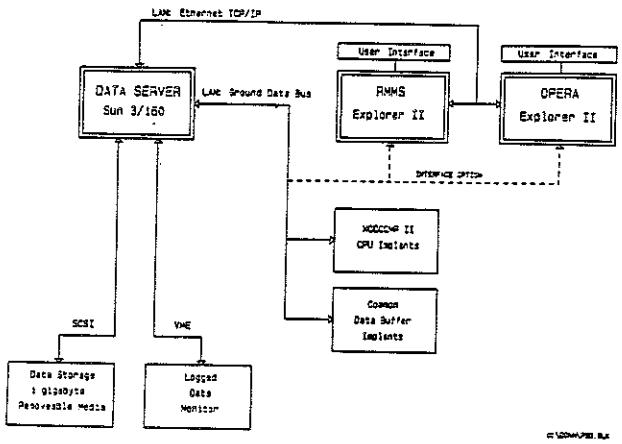
In 1988, RMMS engineers finalized the system software architecture for the Data Manager and initialized software development on a SUN 3/160 workstation. Major software tasks included the LAN interface design, file handling, user interface and Logged Data Monitor simulation. System architecture and hardware design for the Logged Data Monitor has also been completed. This two printed circuit board package is designed for the host SUN 3/160 Data Manager and includes an 11

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

Figure 1

RMMS

System Architecture



MIP 32 bit microcoded CPU with a 96 bit Control Store and a standalone I/O card that allows for high speed interfacing with the Launch Processing System network data.

The preliminary version of the expert system tool has been implemented with Intellicorp's Knowledge Engineering Environment (KEE). Functioning as an information analyst, this tool provides system engineers access to ModComp data information recorded by the Data Manager through the Logged Data Monitor, and subsequently performs automated diagnostic analysis on the retrieved information.

Additionally Remote Maintenance Monitoring System engineers have accomplished the downloading of information to the Data Analyst, as well as testing the user interface which is being functionally enhanced as a result of inputs from Kennedy Space Center System Engineers.

With its advanced designs the Remote Maintenance Monitoring System will be capable of providing enhanced support to the Launch Processing System through 1995. The Remote Maintenance Monitoring System concept efficiently incorporates continuous real-time monitoring of the Checkout, Control and Monitor Subsystem (CCMS) hardware, the centralizing of all diagnostic data acquisition processes and the automation of the diagnostic process via expert system technology.

Presently within the firing rooms at the Kennedy Space Center, the first line of fault isolation when a CPU anomaly or failure occures is the "Memory Dump". Numerous failure modes are identified by the main memory automatic dump process. The Launch Processing System Operating System failure mechanisms will store unique identifiers tagged to the failure type into main memory before the main memory dump is initiated. The binary contents of CPU main memory (typically 128K bytes to 390K bytes) are printed out to a line printer in ASCII HEX format. This main memory data contains information about the state of the machine at the time of the failure (ie. state of the interrupt queues and the identification of the task that was active at the time of the failure). The Launch Processing System Engineer must then manually search this paper listing to find specific data to determine the cause of the failure. This process can take weeks or months for system experts. Among the goals of Remote Maintenance Monitoring System is the elimination of this tedious process through automation and enhancement of fault isolation via an expert system aid.

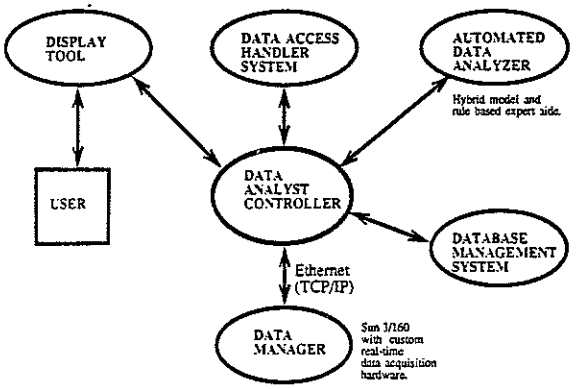
As described above the Remote Maintenance Monitoring System is a combination of Expert System techniques and real-time data capture and evaluation hardware. The Expert System within the Remote Mainenance Moniroting System is known as the Data Analyst Tool. By using custom hardware, the tool examines the contents of the failing CPU's main memory after real-time capture of said data. The Data Analyst Tool can be subdivided into two parts. The first part is known as the Data DIsplay Tool. This Tool elevates the analysis of failure data from a paper listing to an interactive graphical computer interface. Each "dump" is automatically re-organized to provide the user access to the salient memory information necessary to begin failure analysis. This reorganization also allows the correct formatting of this information to be loaded into a relational database of failure histories. Similar failures can then be compared for common modes, trends and symptoms.

The second part of the Data Analyst Tool is the Expert AId for automating fault isolation. This aid is a hybrid of rule based and model based techniques. Through knowledge engineering the memory dump analysis expertise from a number of Launch Processing System Engineers has been incorporated into the Analyst Tool. THe Expert Aid uses a specific model for each failure mode identified by the Launch Processing System software. The LISP model defines the accepted approach that the System Engineers use to trouble shoot ModComp subsystem failures. The model controls which rules are applicable to the given failure mechanism parameters. The architecture for this expert system software is illustrated in Figure 2.

Typically, when a critical Launch Processing System failure occurs the subsystem CPU will identify the failure mode, and then transmit the contents of its main memory to a Shared Peripheral Area (SPA) via a high speed parallel interface. Depending on the Launch Processing System network traffic, this data can be transferred at a rate of one 32 bit word every five microseconds. During this transfer of information, the Remote Monitoring System hardware will capture this data, and store it to disk within the Data Manager. The Data Manager will then automatically create a relational database entry from this data dump as well as archiving it to optical disk. At this point the Launch Processing System Engineer will initiate an automated analysis of the failure data. The Data Analyzer will provide an indication of

Figure 2.

RMMS
DATA ANALYST SYSTEM



The Data Analyzer will provide an indication of the source of that particular failure. Further analysis is available to the System Engineer through the various display tools built into the Remote Maintenance Monitoring System.

The process described above is only a portion of the total Remote Maintenance Monitoring System utilities. Additional custom hardware has been designed to facilitate the Real-Time capture and evaluation of the low level Launch Processing System CPU hardware functions. This Kennedy Space Center designed hardware is in the form of Diagnostic Processor implants that will be integrated into the Launch Processing System's ModComp minicomputers. The data accessed by these implants includes significant failure information that has previously been unavailable to Kennedy Space Center engineers in any shape or form. These implants will monitor and evaluate microcode execution while the ModComp performs at its full rated system speed; which is one microcode cycle every 267 nonaseconds. Due to this computer's high performance rate the implant's design utilitizes two 80 bit microprogrammable processors and a 16 bit microprocessor. In order to evaluate the ModComp's instruction execution and handle the networking communications with the Data Manager, the Remote Maintenance Monitoring System's implants must operate at a speed that is at least one order of magnitude faster than the ModComp. These unique devices support this performance requirement. The design specifications and system architecture have been finalized for the ModComp CPU implant. Conceived as a large-scale data acquisition system, the implant monitors the execution of various ModComp processes to detect impending failures. Subsequently, this enables the implant to create a "snapshot" look at the state of the machine prior to ModComp detected errors and sub-system autodumps. The implant consists of a Zilog Z8002 microprocessor, an 80 bit microcoded memory Image Processor with a 4K writable control store, a second 80 bit microcoded Register Evaluation Processor design with 4K writable control store, amin memory trace logic, a Microcode Address Trace, Microword Comparison Logic, a Communications Interface, +5V and +12V voltage monitors, isolation logic and a local terminal interface.

The initial implementation of the Remote Maintenance Monitoring System at the Kennedy Space Center will allow for this data analysis to be performed by the System Engineer at the Data Manager with graphical displays and ModComp language disassemblers. Mechanisms for configuration management that will control the implant's operational parameters will also be available to the user. Future development plans include an expert system that will model the fault isolation process by focusing the analysis efforts on this wealth of new failure information accessed to the Remote Maintenance Monitoring System by these implants.

The Remote Maintenance Monitoring System is being developed because the maintenance of the Launch Processing System's major components has been an expensive effort requiring many skilled and apecialized people. Kennedy Space Center management recognized the need to enhance the process by capturing expert knowledge. The Remote Maintenance Monitoring System represents the state-of-the-art in automation of maintenance techniques as well as hardware and system design. While not being an end in itself, the Remote Maintenance Monitoring System will continue to evolve and become further integrated within future Kennedy Space Center operational as well as maintenance environments.

A Neural Net Approach to Knowledge Acquisition

Jeremy C. Jones / Kenneth M. Ford
Division of Computer Science
University of West Florida
Pensacola, Florida 32514

INTRODUCTION

At the most fundamental level, the knowledge acquisition phase of expert system development consists primarily of eliciting knowledge from a skilled person. The literature of clinical and experimental psychology contains much work relevant to this attempt to improve communication between the knowledge engineer and the domain expert (Bainbridge, 1979; Hoffman, 1987).

This paper proposes an added tool to the knowledge acquisition process, in a manner tightly coupled to the process of the expert's rule elicitation. This close coupling of the method to the expert's self-examination attempts to deal with a central AI problem, that of assuring semantic relevance to the data in use.

BACKGROUND

Several recent attempts to alleviate the knowledge acquisition bottleneck have been based either loosely or directly on Personal Construct Psychology. Specifically, repertory grid analysis is becoming an increasingly useful tool in knowledge acquisition for expert systems (Boose, 1987; Ford, 1989). The root of this relative success seems to stem from the method's ability to at least partially bridge the gulf between the semantic depth of the domain expert's knowledge and the limited nature of our entirely syntactic tools (i.e., computers). That is, repertory grids may assist the knowledge engineer in eliciting deep knowledge that the expert would not have been able to express otherwise.

Kelly's Role Construct Repertory Grid, commonly called the 'repertory grid', is essentially a complex sorting test in which a list of elements are judged successively on the basis of a set of bipolar constructs (cf. Adams-Webber, 1987). The elements (for example, personal acquaintances), can be either concrete or abstract entities, and may be said to operationally define the repertory grid's universe of discourse. These elements should be chosen carefully to represent the topic of interest as well as possible, and yet be of roughly the same type and level of complexity. Constructs represent the ways in which elements are judged to be similar or different from each other. In other words, they permit the subject to make relevant distinctions among the elements. Thus, a repertory grid may be considered a mapping of elements onto constructs. The data generated by each subject are entered into a separate two-dimensional table, or 'grid', in which there is a column for every element and a row for every construct. Each row-column intersect in this table contains a number indicating how a given construct was applied to a particular figure. For example, the data represented in Figure 1 were elicited from a subject upon the evaluation of acquaintances.

Rate elements on a scale from 1 -> 5, a '1' is at the LHP, a '5' is at the RHP, while the rest fall between the poles.			Elements											
			1	2	3	4	5	6	7	8	9	10	11	12
C1	Somber	Humorous	2	4	3	4	4	4	2	4	3	4	1	5
C2	Individualistic	Group oriented	3	2	4	3	1	4	4	2	2	1	1	3
C3	Generous	Greedy	2	4	3	2	1	4	2	3	2	5	2	4
C4	Driven	Unmotivated	1	2	4	3	4	2	3	2	2	1	2	3
C5	Daring	Cautious	2	2	4	4	1	2	4	2	1	2	3	2
C6	Simple	Complex	4	3	2	3	4	4	4	2	4	4	4	2
C7	Honest	Dishonest	2	4	2	1	1	3	2	1	2	4	2	4
C8	Boisterous	Quiet	4	2	5	4	3	2	5	1	4	4	5	1

Figure 1 : An example of a multivalued repertory grid.

The act of the respondent's assigning a rating to an element on a given construct has been interpreted in a variety of ways, and has thus lead to several different approaches to grid analysis, including information measures, non-parametric factor analysis, conventional factor analysis, principal component analysis, multidimensional scaling, and hierarchical cluster analysis, among others (Adams-Webber, 1987). Recently, some researchers have developed procedures for the logical analysis of construct entailment (Gaines & Shaw, 1980; Ford, 1987). In such approaches, the grid data are viewed not as a set of vectors, but instead as an assignment of truth values to logical predicates. The left and right-hand poles of each construct are considered logical predicates that may be applied to the elements.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

It is these 'logical' approaches to construct entailment that are used to generate expert system rules from repertory grid data.

Hidden Redundancy and Meta-Constructs

Although the repertory grid method of knowledge acquisition has been reasonably successful, there remains room for considerable improvement. One difficulty facing repertory grid oriented knowledge acquisition tools is the expert's inability to provide a reasonably complete set of constructs. It is somewhat ironic that a knowledge acquisition tool should suffer from this problem. The repertory grid method is adept at finding relations (entailments) between constructs and formulating them as expert system rules, but offers little assistance in the elicitation of the constructs themselves.

Typically, the constructs are elicited by asking the expert to compare the grid's elements (which are usually elicited first) with each other in groups of two or three. When groups of two are used, the expert is asked to name a construct (an attribute) that distinguishes each from the other. When groups of three are used, the expert is asked to name a construct that two of them have in common that distinguishes the pair from the third element. As noted by Butler and Corter (1985), there are logical problems for construct elicitation methods that rely exclusively on questions about grouped elements. Contrasting two elements with a third can fail to discover an important construct (feature) that distinguishes the pair. Conversely, if two objects are contrasted with one another, an important common feature may be overlooked. These objections can be met if the knowledge engineer asks the expert to (1) give the common features of a pair of elements A and B, and (2) give the features that A has that B does not (distinctive features of A) and that B has that A does not (distinctive features of B) (cf. Tversky, 1977). However, this technique still suffers from the obstacle that the number of pairs and triples increases exponentially with the number of elements. Thus, even if the interviewing method could in principle discover the construct structure of the set of elements, the process can be impractical.

Meta-Construct Feature Identification

The work described in this paper is concerned with assisting the expert to uncover non-linear patterns, as they occur, during the construction of a repertory grid, suggesting higher level, meta-constructs. Kelly's theory suggests that such super-constructs might act as pivotal invariants when searching for alternative construct systems. When such hidden construct (attribute) abstractions or superordinate constructs are found, the suggested construct will be tentatively presented to the expert for approval, thus providing added insight in extending, or revising, the grid. The expert's preliminary repertory grid data will serve as input to a multi-level artificial neural net, in which the non-linear abstractions formed by error back propagation learning at intermediate nodes, called "hidden constructs", will be identified with the meta-constructs. Since the hidden constructs are formed from semantic primitives (constructs), it seems likely that they correspond to unconscious superordinate and/or subordinate constructs. These artificial neural net abstractions are expected to serve as suggestive nudges to the expert in redefining, reducing, and sharpening self-knowledge. Artificial neural net learning algorithms seem especially suited to this discovery of semantically imbued patterns, as they are capable of forming a hierarchy of abstractions from training data. For example, the Boltzmann machine has been shown to minimize the sum of the squares error on a training set, given sufficiently slow annealing (Geman & Geman, 1984).

Necessity of Feature Identification

We expect the patterns in a repertory grid to be manifestly non-trivial, in the sense of not being linearly separable. This means that the grid structure can not be captured by node-to-node correlations, represented as connection weights. This makes it necessary to include abstractions, represented by hidden units, which must then be added as extra non-explicit nodes in the net. By detecting certain of an allowable class of linearly separable features in the data, and adding these features to the data, the pattern becomes linearly separable, and the structure easily detected as a simple correlation. These feature detectors are implemented as input layer hidden-units.

Minsky and Papert point out that such features always exist. That is, they show that there is always some set of features, defined and detected by the addition of simple perceptron-like, linear threshold detectors, acting on the inputs to the net, which augment the net to permit reproduction of any required Boolean mapping via simple linear threshold elements (Minsky & Papert, 1969).

This is most clearly understood by thinking of these feature detectors as defining decision hyperplanes, i.e., appropriate linear combinations which can isolate an arbitrary point of decision space from any finite set of distinct points -- similar to drawing a box around any point. Since the strength into a unit is the scalar product of the input vector with the weight vector to the unit (plus the bias), the decision plane defined by such a hidden unit is easily described as the plane perpendicular to the connection strength weight vector, translated along the vector by the threshold distance (i.e., bias) at that node.

Boxing in a set of feature points by these hyperplanes may be thought of as implementing a kind of statistical cluster analysis, but one in which the space metric is defined to reflect the particular pattern error measure in use. This error measure is typically taken to

be the sum of the square errors, taken over both the set of training patterns, and all outputs, as is the case in our computer net simulations.

Thus Minsky and Papert have shown us both the necessity and generality of detecting meaningful features, in the form of useful linear combinations of input patterns, when capturing a non-trivial (non linearly separable) pattern with an otherwise linear, thresholded system.

Several tools are available for such feature detection. Error back propagation, and simulated annealing, are both methods for automatic identification of heuristic (hence generally non-unique), working sets of such features, which are needed to capture non-trivial (i.e., third order) patterns.

NON-UNIQUE FEATURES: XOR EXAMPLE

A difficulty of this study is that the discovered features may be somewhat arbitrary. This means that they may not exactly match the syntactic form in which the expert has become used to seeing them. The exclusive-or is used to show that there may sometimes be two equally attractive sets of features.

The simplest example of a pattern which is not linearly separable is the exclusive-or function. The behavior of the exclusive-or cannot be statistically captured by the delta learning rule; therfore, this pattern is described by third order information.

That the exclusive-or pattern cannot be recovered from only first and second order information, is seen by noting that both the XOR and XNOR functions have the same first and second order descriptions. That is, the delta learning rule will adjust the threshold cutoff biases (first order) and the connection strengths (second order correlations) to the same values, for both the XOR and XNOR function. The average of (0 0 0) (0 1 1) (1 0 1) (1 1 0), is (.5 .5 .5), and the mutual correlations of (0 0 0) (0 1 1) (1 0 1) and (1 1 0) are all zero, as with the XNOR.

In the case of the exclusive-or, the added features are usually taken to be the logical and, with the logical or, which linearly isolates points (0 0 0) and (1 1 0). But an equally efficacious choice of feature detectors would be x^y', with y^x'. This distinct set of primitive feature detectors serves to isolate points (0 1 1) and (1 0 1).

SYMMETRY BREAKING & NON-UNIQUE RESULTS

Because back propagation is a steepest descent method, symmetry in the net initial conditions can lead to a metastable state. Therefore the algorithm must be started by breaking any such symmetry. This is done by initializing connection weights and thresholds with random values. This is appropriate for such a heuristic method, and accounts for the range of results produced by successive runs of the same net configuration.

EPISTEMOLOGICAL NATURE OF OUR DATA

A primary difficulty of this study is probably its greatest strength, i.e., that we are dealing with the deeply compiled (over a professional lifetime), epistemological beliefs of an expert. Our purpose is not to engineer a pattern replicating engine, but to suggest to the expert that some of the features we have detected in the repertory grid may be implicit 'chunks' of their unconscious model building. Thus we intend to aid the expert's self-searching by nudging him or her with hints of deeper questions while he or she is attempting to add new constructs to the repertory grid.

Therefore, any detected features must be filtered to match characteristics of human model building because of their assumed epistemologic content. For example, the features detected in our application of the above methods to expert repertory grids will be made: discrete (-1, 0, +1); small in number of constructs; Boolean, to reflect the qualitative nature of the expert's self-knowledge and our desired rule base; and truncated to no more than five items, to reflect the short-term memory limitations of humans.

SEMANTIC RELEVANCE

While admitting the above difficulty, we gain the advantage of assured relevance. By close interaction with the expert (i.e., subjecting each tentative feature for approval) we are avoiding what should be a dread danger to the cautious researcher, that of crossing Godel's abyss, and of making the usual chasmic assumption of the relevance of our mechanically derived syntax vs. the human expert's generally hermetic and deeply held semantics.

Because this method is intended for use with an interactive knowledge acquisition shell (see Ford), its output to the expert is qualitative, as described above. The system will simply suggest groups of constructs (each with a + or -) which the expert should consider as a potential abstraction to be distinguished by the choice of the next construct.

THE METHOD

The method had to be fast enough to be used with a real-time, interactive, expert system construction shell. With this requirement, simulated annealing in the Boltzman machine was considered too slow, therefore the faster and simpler Back Propagation learning method was used. The fundamental back propagation method uses simple gradient descent to reduce output error (with the chain rule), and a differentiable squashing function (used to 'spread the blame' of any output error over the entire net). This method distributes the connection strength changes throughout the net to all nodes contributing, however indirectly, to the error at each output node. The software is straightforward, and is taken from McClelland and Rumelhart's handbook of PDP modeling (McClelland, 1988).

RESULTS

The following is the result of applying this back propagation method to the earlier acquaintance-evaluation repertory grid. The honest/dishonest construct is taken to be the output pattern to be generated from the other constructs. Two hidden unit layers of abstraction were used, rather than one layer. This is because it was conjectured that the required small 'chunks' might be more discoverable by the net if there were more than one layer in which to build abstractions.

The following is the result of 5,000 training iterations, each using the twelve cases (elements), given in the previous acquaintance evaluation repertory grid. The pattern output was encoded as a binary vector to make results non-parametric. Thus, a 'very' honest element would be encoded with the binary vector (1, 0, 0, 0, 0); a 'fairly' honest acquaintance would be encoded with (0, 1, 0, 0, 0); and a 'very' dishonest grid element would be encoded as (0, 0, 0, 0, 1), etc.

Because of the learning algorithm's theoretical capacity to adjust the input decision hyperplane (Boolean feature detector), which defines the binary cutoffs as required, no generality seemed lost in using the numeric construct values (1, 2, 3, 4, 5) as inputs to the learning algorithm. This assumption is implicit in the following work, and is only justifiable if (1) sufficiently many hidden units were provided in the net to capture as many Boolean features, as were needed to capture the pattern, and (2) the steepest descent learning algorithm succeeded in finding these features, i.e., did not become trapped in a 'bad' local minimum.

The sum of the square errors over all five binary outputs, and all twelve elements (i.e., over a total of sixty outputs), was only 2.5002, or an average error per output of only 0.04, with two seven-unit successive hidden layers. Given the squashing function approximation to the cutoff step function, this is virtually a perfect fit. This was the best fit achieved over about twelve net configurations which were tried. The other total error values for different net configurations ranged between about 3 and 8. Again, the two hidden layer net was used because it was felt that a deeper net might correspond better to human abstraction, in the sense of permitting smaller chunking (it also yielded a better fit).

In interpreting the data below, we should notice that each value is expressed in 100ths, and that each row vector is perpendicular to the decision hyperplane which characterizes the feature detected by its hidden unit, with one caution -- back propagation requires a differentiable sigmoid smearing function. A large vector magnitude represents sharpening the trigger break on the threshold function, and the bias should be considered relative to the vector magnitude. Stated more precisely:

$$1/[1+\exp(-(w.i - \text{bias})/T)]$$

is the hidden unit response, using the scalar product of its weight vector, with the input data vector. The point is that it's the relative size of each weight vector component which determines the direction of the decision plane, and hence the relative emphasis given to each input to that hidden unit. This is nicely consistent with the qualitative simplifications we perform, to be meaningful to the expert's semantic introspection.

Below is the raw data from one of the runs, where the first seven columns represent the following grid constructs (abbreviated at the top of each column):

SH: Somber vs. Humorous
IG: Individualistic vs. Group oriented
GG: Generous vs. Greedy
DU: Driven vs. Unmotivated
DC: Daring vs. Cautious
SC: Simple vs. Complex
(Honest vs Dishonest, used as binary training output)
BQ: Boisterous vs. Quiet

Each row represents a hidden unit of the second, feature-detection layer of the net. Any row, which strongly influences the net output, is a candidate for identifying as one of the expert's meta-constructs, since forming this pattern of constructs was part of the feature construction which helped minimize output error on the twelve training patterns.

weights(Inputs=Col To Hidden1=row), weights(Hidden1=Col To Hidden2=Row)

SH	IG	GG	DU	DC	SC	BQ	bias	h1	h2	h3	h4	h5	h6	h7	bias
497	66	1462	-852	140	-285	XXXX	-4	-89	40	278	-619	49	-646	-250	-640
-43	-18	-111	-67	-15	-57	-77	-68	-763	-19	-20	129	338	192	-168	131
-99	-25	642	-540	-153	-381	-457	-174	-69	-141	-73	-129	-72	-58	-119	-80
484	287	51	250	306	72	-176	157	-48	-51	-81	-135	-38	-157	-34	-136
-222	-289	-487	-338	76	-104	-262	18	-28	-85	25	6	-71	65	-316	18
89	75	81	18	46	113	62	25	-404	2	41	112	18	98	-843	157
479	-500	47	-24	246	-139	-146	29	-81	-39	-36	36	34	84	-392	65

weights (Hidden2=Columns to Outputs=Rows)

pc1	pc2	pc3	pc4	pc5	pc6	pc7	bias	
-580	722	-88	-45	-224	-959	-189	-88	++ H = very honest
-849	437	90	-42	113	954	173	XXX	+ H = fairly honest
-281	-473	-263	-434	-170	-500	-375	-802	0 H = neutral
606	-690	-124	-151	-222	-739	-411	155	- H = fairly dishonest
-276	-462	-177	-348	-128	-445	-313	-766	-- H = very dishonest

INTERPRETATION OF RESULTS FOR THE EXPERT

Using cutoffs to make these results easier to read, the first row vector of weights, defining the first feature detector, becomes:

1 Somber-Humorous:	4.97
0 Individualistic-Group oriented:	.66
1 Generous/Greedy:	14.62
-1 Driven/Unmotivated:	-8.52
0 Daring/Cautious:	1.40
-1 Simple/Complex:	-2.85
1 Boisterous/Quiet:	XX.XX (large weight)
0 [bias=	-.04]

Doing this same qualitative translation for all seven features:

	h1	h2	h3	h4	h5	h6	h7
Somber-Humorous:	1	0	0	1	-1	0	1
Individualistic-Group oriented:	0	0	0	1	-1	1	-1
Generous/Greedy:	1	-1	1	0	-1	1	0
Driven/Unmotivated:	-1	-1	-1	1	-1	0	0
Daring/Cautious:	0	0	0	1	0	0	1
Simple/Complex:	-1	-1	-1	0	0	1	0
Boisterous/Quiet:	1	-1	-1	-1	-1	1	0
bias =	0	-1	-1	1	0	0	0
degree of Influence:	!	?	?	!	?	!	!

(where ! denotes relatively strong influence, and ? denotes weaker influence, as defined by the columns of the upper right connection strength matrix, i.e., each hidden layer-1 unit's influence on hidden layer-2).

The more influential of these columns, as determined by their maximum connection strengths to successive hidden units, can be presented to the expert as possible meta-constructs, influencing his modeling, and being of possible use as pivoting, super-constructs, when considering construct revisions.

The above example yields four possible meta-constructs, relative to judgments of honesty. They are the following:

chunk1 = +Somber +Generous -Driven -Simple +Boisterous
chunk4 = +Somber +Individualistic +Driven +Daring -Boisterous
chunk6 = +Individualistic +Generous +Simple +Boisterous
chunk7 = +Somber -Individualistic +Daring

These are all tentative super-constructs, available to the expert to discard or adopt for his self-insight while constructing an expert system. It is hoped that some combinations might give rise to an 'aha!', and stimulate introspection, contributing to semantically deeper production rules in the construction of the expert system.

Since, in eliciting such a grid from an expert, we are dealing with a lifetime of deeply compiled knowledge, the patterns captured are epistemologic, and so the features extracted might be very unique to the individual expert, and not very recognizable as useful by another person.

If any of the above combinations give rise to insightful abstractions, then it would become appropriate to examine the connection strengths into the next layer for any influential penultimate constructs. This gives such chunks ordinary statistical or functional meaning, in terms of the net outputs, in addition to their earlier described deeper epistemological values. In our above data we find the following:

PC1 = -chunk4 -chunk6
PC3 = -chunk4 -chunk7
PC4 = -chunk4 -chunk6 (= PC1)
PC6 = -chunk1 -chunk7

And finally, the discretized honesty/dishonesty construct is qualitatively describable in terms of these second level meta-constructs:

very honest = -PC1 (+PC2) -PC6
fairly honest = -PC1 +PC6
neutral = -PC6
fairly dishonest = +PC1 (-PC2) -PC6
very dishonest = (-PC2) -PC4 -PC6

SUMMARY

The method described above may be particularly effective in finding the superordinate constructs upon which the expert relies, but is not consciously aware, or has no developed vocabulary to describe. Improvements in the expert's ability to recognize the constructs he uses to make distinctions about a set of elements is an advance in both knowledge acquisition and psychology.

CONCLUSION

The four level artificial neural net shown, with a width in hidden units chosen equal to the number of constructs, detected manageably small and simple repertory grid meta-construct features, while capturing the grid structure remarkably well, when one of the constructs was chosen as an output to be matched. By staying tightly bound to the semantics of the expert, in his own terms, we seem to cope with the invariable underestimation of modeling difficulty, from semantics to syntax. This distance seems especially vast when attempting to capture the unique quality that distinguishes the world-class expert from competent players.

REFERENCES

Adams-Webber, J.R. 1987. Repertory grid technique: Research and assessment. Invited lecture presented to Seventh International Congress on Personal Construct Psychology, Memphis, Tennessee.

Bainbridge, L. 1979. Verbal reports as evidence of the process operator's knowledge. *International Journal of Man-Machine Studies*, 11, 411-436.

Boose, J.H. 1987. A knowledge acquisition program for expert systems based on personal construct psychology. *Int'l. J. of Man-Machine Studies*, 26, 3-28.

Butler, K.A. and J.E. Corter. 1986. Use of psychometric tools for knowledge acquisition: A case study. In *Artificial Intelligence and Statistics*, W.A. Gale, ed. Addison-Wesley Publishing Co., Reading, MA, 295-319.

Ford, K.M. 1989. A constructivist view of the frame problem in artificial intelligence. An Invited Commentary. *Canadian Journal of Psychology*.

Ford, K.M. 1987. An approach to the automated acquisition of production rules from repertory grid data. Ph.D. diss., Tulane University.

Gaines, B.R. and M.L.G. Shaw. 1980. New directions in the analysis and interactive elicitation of personal construct systems. *Int'l. J. of Man-Machine Studies*, 13, 81-116.

Geman, S. and D. Geman. 1984. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6, 721-741.

Hoffman, R.R. 1987. The problem of extracting the knowledge of experts from the perspective of experimental psychology. *The AI Magazine* (Summer).

McClelland, J.L. and D.E. Rumelhart. 1988. *Explorations in Parallel Distributed Processing*. The MIT Press.

Minsky, M. and S. Papert. 1969. *Perceptrons*. MIT Press.

Tversky, A. 1977. Features of similarity. *Psychological Review*, 84, 372-352.

KNOWLEDGE ENGINEERING DOMAIN KNOWLEDGE FOR AN INTELLIGENT COMPUTER AIDED INSTRUCTION SYSTEM

Kent E. Williams
Institute for Simulation & Training
University of Central Florida
12424 Research Parkway
Orlando, Florida 32826

Richard E. Reynolds
Naval Training Systems Center
Orlando, Florida 32826

Thomas F. Carolan
Institute for Simulation & Training
University of Central Florida
12424 Research Parkway
Orlando, Florida 32826

Empirical evidence from cognitive learning research (see Williams & Reynolds, 1989 for a review) suggest that cognitively engineered instructional exercises will improve the efficiency of learning compared to less rigorously formulated approaches to instructional development. A set of methodological guidelines for specifying the instructional content for an intelligent tutoring system are discussed and applied to the training of tactical console operations.

The approach taken is based on a cognitive simulation modeling (Kieras, 1987a, 1989b) or explicit rule formulation approach to learning. Two assumptions underlie this approach. The first is that learning to perform a task such as operating a tactical console involves the development of a mental model or cognitive simulation model in which the essential information and operational procedures required for task performance are represented. This model is then used to guide subsequent interaction with the device. The second is that the underlying architecture of cognition can essentially be defined within a production system framework (Anderson, 1983, 1986). The cognitive simulation model then, is assumed to be structured by a production system architecture. From this perspective, one goal of training is to guide the development of a cognitive simulation model which will provide an accurate representation of the tasks to be performed and the information necessary for their performance.

Williams and Reynolds (1989) discuss the evidence for the various processes and components involved in a cognitive simulation model formulation. These components include: processes for selecting or encoding information, processes for organizing facts into a rule-like structure, processes for organizing rules into a hierarchy of goals and subgoals, and mechanisms for directing the processing of information employing this network of rules. By presenting the instructional content in a production system format both the acquisition of knowledge and its subsequent application will be facilitated.

The methodological guidelines presented provide a way to direct the structuring of information in accordance with the production system model of cognitive learning. It involves a form of information analysis which is generic and can be applied to any domain. Once the information is structured, it in turn drives the construction of exercises for the intelligent tutoring system. The exercises developed as a result of this knowledge engineering process (the process is essentially consistent with that which is required in knowledge engineering when developing an expert system knowledge base) specify explicitly the cognitive simulation model which the trainee must acquire to effectively operate the tactical console. The correspondence between the cognitive simulation model developed through training and the production system formatting of the instructional material is essentially isomorphic. The goal of the training program is achieved when the trainee works through the exercises and develops an accurate representation of the operation of the console.

Information Analysis

Following Kieras (1987a, 1987b), an initial analysis of the information required to effectively operate the tactical console yields a device explanation hierarchy. This analysis can provide information at a number of different levels, not all of which will be necessary for the effective performance of the required tasks. In the current operational task, information about the internal workings of the device will not provide the trainee with additional power to infer correct procedures. The mental model is developed from an explicit set of procedures, with no requirement for inference based reasoning. What is required is a model which includes: the overall functions of the console, the physical layout of the console with locations of controls and outputs, the external console operating procedures, the functional relation-

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

ships between console parametersand task parameters, and how to use the console parameters to perform the relevant tasks. Figure 1 is a portion of the explanation hierarchy for the tactical console. It consists of the information that would be used to explain or describe the relationships among the input and output components of the console. Figure 2 is a portion the overall functional hierarchy of the console. It consists of the information that would be used to explain what the console is to be used for.

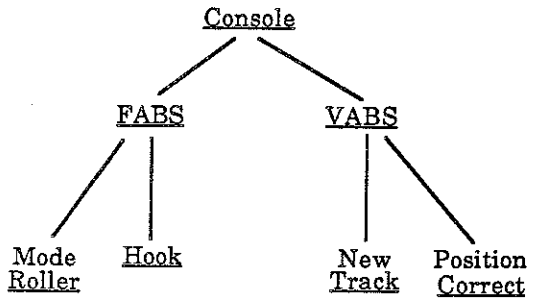


Fig. 1. Explanation hierarchy

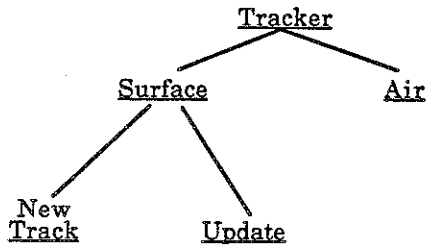


Fig. 2. Function (Goal) hierarchy

A task-goal control hierarchy is then formulated with elements which are linked to the elements that make up the explanation hierarchy. The hierarchy of goals is expanded to include subgoals related to the tasks to be learned and performed. This task-goal hierarchy assists in guiding the inclusion of relatedness and sequencing information into the design of the instructional content. In figure 3 some of the links between the goal hierarchy and the explanation hierarchy are illustrated. These links yield a task-goal hierarchy as shown in figure 4.

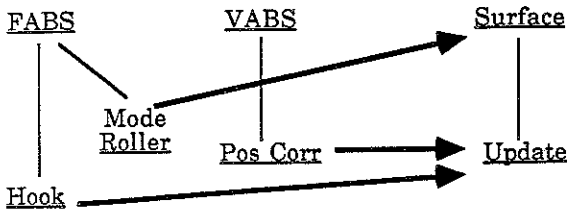


Fig. 3. Links between explanation and goal-task hierarchies.

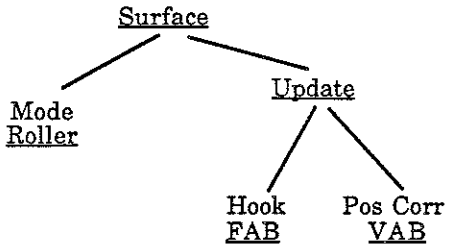


Fig. 4. Task-Goal hierarchy

A detailed cognitive analysis of the domain knowledge is then completed to organize the goals and subgoals into an AND/OR graph hierarchy. This analysis is similar to what Kieras has called a rational analysis, meaning the logical requirements for conducting the task, not a psychological analysis of what the skilled user seems to know. All responses of the individual required in order to perform the to be learned tasks are specified and associated with each task goal and subgoal in the hierarchy. If the goal or subgoal requires a sequence of responses, that sequence must also be specified. All stimuli or conditions associated with each individual response is identified and linked to that response. This requires the identification of all declarative knowledge (i.e, names, locations, symbols, etc). All constraints (i.e, time) required by console operating procedures are identified and linked to the appropriate responses. A task-goal hierarchy in AND/OR graph form is illustrated in figure 5.

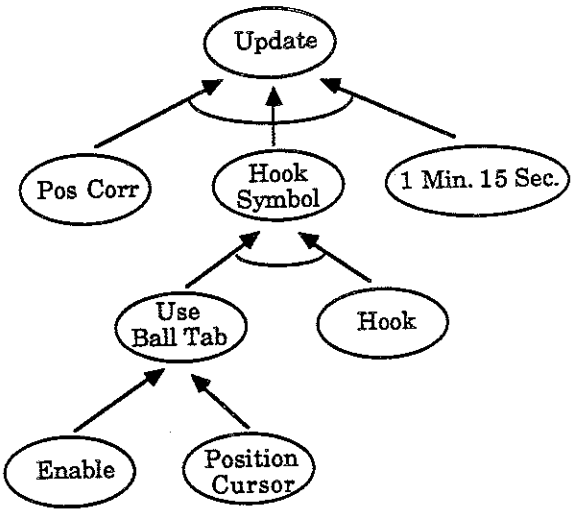


Fig. 5. AND/OR graph of the task-goal hierarchy

From the task-goal hierarchy a set of rules and declarative facts is developed which completely define system operation. The hierarchy of rules (IF-THEN statements) is associated with each goal and subgoal in the AND/OR graph of the task-goal hierarchy. Each rule combines the relevant pieces of knowledge into a chunk. Goals and subgoals embedded in the hierarchy of rules are important as a design principle for

structuring to be learned information. The specification of these IF/THEN rules is essentially the development of the production system which serves as a simulation of system functions and operations. Following is a sample of the rules which describe in part the operating of the tactical console.

Rule 14 IF the subgoal is to update a tentative tract
THEN hook the track symbol
AND use your position correction variable action button (VAB)
AND complete this procedure in 1 minute and 15 seconds

Rule 15 IF the subgoal is to hook the track symbol
THEN use the ball tab
AND press the hook button

Rule 5 IF subgoal is to use the ball tab
THEN locate and press the ball tab enable button
AND manipulate the cursor with the ball tab

Rule 17 IF the subgoal is to use your position correction VAB
AND hook track symbol has been accomplished
THEN use your ball tab
AND press position correction VAB

DCLR 8 Locate and press ball tab enable button

DCLR 9 Locate and manipulate cursor with ball tab

DCLR 16 Locate and press hook button

DCLR 18 Locate and press position correction VAB

DCLR 10 Describe and identify a tentative track symbol

These rules serve to develop the production system which models the operation of the tactical console in order to update a tentative track. Given this model, all relevant knowledge has been specified and related such that the operation of updating a tentative track can take place. The student must learn the model as a basis for becoming skilled at console operation. Consequently, in constructing an exercise for a training system which teaches console operation, the student must first learn all of the declarative facts such as that from 8, 9, 16, 18, and 10 (or, locate the ball tab enable, hook and position correction buttons, manipulate the cursor and identify a tentative track symbol). He must then learn how to compose or compile these facts such that when certain cues or subgoals are set certain actions or sequences of actions are triggered. In this example, the appearance of a tentative track sets the subgoal to update the track. This in turn requires that the operator hook the track symbol, know the proper sequence of actions to take in order to use the position correction variable action button, and to do all this within 1 minute and 15 seconds (the time constraint is a system feature). To implement this procedure however, the operator must know how to hook the track symbol, Rule 15; know how to use the ball tab, Rule 5; and, know how to use the position correction VAB, Rule 17. In order to enable the actions required of these rules he must also be knowledgeable about specific declarative facts such as DCLR 8, 9, 16, 18, and 10. In order for rules 14, 15, 5, and 17 to be learned, all of the facts making up these rules must also be learned.

Instructional Content

The instructional content of the intelligent computer aided instruction system is created from the chunking of the declarative facts and the formation and composition of rules which trigger the required actions. The ICAI system is then in essence the information required for a simulation of the cognitive operations required to operate the console.

A large network of exercises is developed from this production system. An exercise is associated with each rule and declarative fact in the system. Each exercise in turn represents an underlying rule which combines relevant facts and/or lower level rules. Trainees are presented with these exercises and the process of acquiring knowledge continues until all facts and rules are learned. The trainee works his way through this network to acquire expertise.

In working through the exercise set for the training system which teaches tactical console operation, the student learns all the declarative facts and how to compose or compile these facts such that when certain cues or subgoals are set certain actions or sequences of actions are triggered. Each exercise explicitly describes the subgoal or goal state associated with the production represented by the exercise. The exercises are tightly related to the rules and facts which they represent.

Each exercise includes an exposition of the information represented by the rule, an example which requires a response from the student to test the students ability to use the knowledge addressed by the exposition, and a diagnostic which determines what part or parts of the rule have not yet been mastered. The student is presented with the expository information and asked to respond to a question or perform a procedure. If the response is incorrect, the trainee is presented with a set of diagnostics to isolate the source of the error in terms of a specific rule or fact.

The student is diagnosed in terms of the conditions or actions for which he has weak knowledge. Since each exercise created employing this methodology can be explicitly linked to pieces of knowledge in the form of facts or rules, the training program queries the trainee if he fails an exercise to determine what rules or facts have not been learned by the student. Each exercise has one diagnostic that is directly associated with the new content of that exercise, and additional diagnostics for each rule and declarative fact from which it is composed. The system then selects another exercise for the student which addresses these unmastered facts or rules and which will serve to strengthen the cognitive model of console operations. For

example, if a student fails an exercise which addresses Rule 14, the instructor can then diagnose the student in terms of the conditions or actions for which the student has weak knowledge. The diagnosis may reveal that the student doesn't understand "how to hook the track symbol". Consequently, an exercise dealing with hooking the track symbol, Rule 15, would be selected next for the student.

Advantages of coupling instructional content to a rule based cognitive simulation model include efficient delivery of exercise content and effective diagnosis of errors in the development of the cognitive simulation model in terms of specific rules, declarative facts, or combinations of rules. Through this assessment of student strengths and weaknesses a determination of what lesson content should be presented next is made.

Instructional Strategy

Learning then progresses with the formation and acquisition of a cognitive simulation model. When all rules are learned the student has acquired a complete cognitive model of console operations which is used when interacting with the console.

Having a set of exercises in which the instructional content is engineered according to cognitive learning principles is not in itself sufficient. An instructional strategy, grounded in sound theory and empirical research, must be implemented.

The instructional strategy employed involves adaptive exercise sequencing capitalizing on the trainee's prior knowledge. Sequencing of exercises plays an important part in the process of new rule assimilation and composition, and, as Kieras and Bovair (1986) have shown, is conducive to training efficiency and effectiveness.

This instructional approach is based upon the heuristic that one learns from relationships with what one already knows. New rules are related to rule sets which exist in part or in whole in the trainee's knowledge base so that they can be assimilated effectively and efficiently. The importance of relating to existing knowledge and of overlapping new knowledge with old knowledge is critical to guiding the lesson sequencing. New rules have a better chance of entering the system when they can be associated with strong existing clusters of rules (Thargard & Holyoak, 1985). Exercise sequencing is then dependent on the strengthening of existing rules to which that information can be related.

In the case at hand, exercises representing lower levels in the hierarchy of related facts and rules should be strengthened first. This will serve to strengthen related rules (composed rules) further up in the hierarchy and assist in their assimilation. Appropriate exercise sequencing will optimize the overlap between productions, which will reduce training time (Kieras & Bovair, 1986).

Results of the diagnostic tests associated with each exercise becomes the input to an exercise selection heuristic. Sequences of exercises are selected by the heuristic such that the overlap between productions represented by each exercise is optimized, and prior declarative knowledge and production knowledge is exploited. In this way existing partial knowledge is used to guide learning.

The capability to select a set of exercises which overlap with an exercise that the student has failed is built into the instructional system through the selection heuristic. The heuristic selects the exercise with the greatest overlap in partial knowledge with the exercises the student has already mastered and the least new or previously failed elements. When no appropriate new lesson is found, the exercise just failed may be presented for additional practice.

References

Anderson, J.R. 1983. *The Architecture of Cognition.* Harvard University Press, Cambridge, Massachusetts.

Anderson, J.R. 1986. "Knowledge Compilation: The General Learning Mechanism." In *Machine Learning an Artificial Intelligence Approach Volume II*, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, eds. Los Altos, California.

Kieras, D.E. 1987a. "The Role of Cognitive Simulation Models in the Development of Advanced Training and Testing Systems." Technical Report TR-87/ONR-23. Office of Naval Research.

Kieras, D.E. 1987b. "What Model Should be Taught: Choosing Instructional Content for Complex Engineered Systems." Technical Report TR-87-ONR-23. Office of Naval Research Technical Report.

Kieras, D.E. and S. Bovair. 1986. "The Acquisition of Procedures from Text: A Production-System Analysis of Transfer of Training." *Journal of Memory and Language* 25: 507-524.

Thagard, P. and K.J. Holyoak. 1985. "Discovering the Wave Theory of Sound." In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence.* Morgan Kaufmann, Los Altos, California.

Williams, K.E. and R. Reynolds. 1989, in press. "The Acquisition of Cognitive Simulation Models: A Knowledge Based Training Approach." In *Advances in Simulation*, P.A. Luker and B. Schmidt, eds. New York.

A SUPPORT TOOL FOR EVALUATION OF KNOWLEDGE ENGINEERING STRUCTURES

L.J. Kohout, S.M.A. Mohamad and W. Bandler
Institute for Expert Systems and Robotics and
Department of Computer Science
The Florida State University
Tallahassee 32306-4019, U.S.A.

Keywords

Design of concurrent computing systems, Knowledge-based systems, Dynamic protection, Performance evaluation, Activity structures, Simulation, Statistical models, Human-computer interaction.

Abstract.

To deal successfully with the challenge of the Japanese Fifth and Sixth Generation AI architectures, it is essential to appraise the effect of a variety of factors on the performance of AI Knowledge- Based Systems early in the prototyping stage.

A software tool than can support building of a variety of well- protected knowledge-based systems with optimalised performance and well-protected knowledge structures for a multiplicity of user environments is thus important for successful and competitive commercial knowledge engineering design. The paper discusses some issues involved in developing and use of a support design tool based on the Activity Structures methodology.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

1 DEVELOPMENT OF THE SUPPORT TOOLS AND METHODOLOGY FOR DESIGN AND VALIDATION OF COMPUTER AI ARCHITECTURES WITH DYNAMICALLY PROTECTED KNOWLEDGE STRUCTURES

Progress in Artificial Intelligence on the software side of computer science, combined with spectacular advances in hardware, involving Very Large Scale Integration and parallel processing, leads to the development of radically different computer architectures for Intelligent Knowledge-Based Systems and multi-center expert systems utilising deep knowledge. Such AI Systems will play increasingly important roles in design and manufacturing in the High Technology industries.

Practical considerations and the quest for high cost-efficiency requires that the AI Knowledge-Based Systems be reliable and perform well. The crucial factors determining the quality of their performance and cost are the following:

1) factors determining speed and computational power;

2) the quality of the human-computer interface;

3) availability of dynamic protection mechanisms imposed on the knowledge structures, guarding against conflicts of concurrent processes that would lead to degradation or disastrous shutdowns of the whole system, as well as against the corruption of the knowledge structures themselves;

4) protection mechanisms against theft and malicious damage as well as accidental destruction of the knowledge structures and data;

5) capability of the system to deal with the multiplicity of environments and contexts that appear in the applications in which it is utilised (such as manufacturing, insurance underwriting or medical decision making).

The design methods for tuning the performance of computer architectures are usually dealt with in a completely separate way from the methods for the design of well protected computing systems. Unfortunately, it has been shown that a computer architecture tuned for high performance may show severe degradation of its performance if the protection features are only added after the completion of the design (as is the case e.g. with the Cambridge CAP machine). This makes it necessary to develop design methods that allow us to combine, and evaluate the effects of various combinations and interactions of the factors (1) - (5) above in an efficient way. This has to be done at the stage of prototyping, prior to the completion of the design, as most realistic systems require a number of man-years for their completion.

Thus this paper addresses the results achieved as well as the problems encountered, in development of an efficient tool for computer- aided simulation and evaluation of the effects of interaction of the following factors:

1) structural features of architectures;

2) dynamic protection features;

3) the effect of change of various user environments on the knowledge bases as well as on the inference structures;

4) the effect of various contexts in which the knowledge structures are utilised.

The approach described here is based on Activities Structures design methodology (Kohout 1987), the suitability of which for highly constrained multiprocessor architectures has been demonstrated previously by the authors. This approach has developed from the work on dynamic protection of descriptor oriented computer architectures (Kohout and Gaines 1975; Kohout and Bandler 1981), that was integrated within the framework of global human-computer interaction (Kohout, January 1978), and strongly influenced by work on neuro-computational structures (Kohout, 1976). It has been successfully applied in several fields of computing, including Knowledge Engineering (Kohout 1989; Kohout, Anderson and Bandler 1989).

2 SUPPORTING METHODOLOGY

The design of a system – an *Information Processing Machine* or IPM – is viewed as a process of building a representation of a composite object that is to be constructed according to some requirement specification. Since software is not a physical object, the role of representation is especially important. Our design paradigm consists of the following fundamental representation of the design situation (Kohout and Mohamad 1989):

DESIGN PARADIGM ::=
<SYSTEM MODEL, DESIGN PROBLEM, DESIGNER>.

SYSTEM MODEL ::=
<MULTI-ENVIRONMENT, IPM, ME-IPM INTERACTION>

DESIGN PROBLEM::=
<REQUIREMENTS KNOWLEDGE ELICITATION, CREATIVE KNOWLEDGE REPRESENTATION, DESIGN OF IPM>

CREATIVE KNOWLEDGE REPRESENTATION::=
<BASE KNOWLEDGE TRANSFORMATION, NEW KNOWLEDGE ACQUISITION>

Multi-Environmental SYSTEM MODEL, the DESIGN PROBLEM and the DESIGNER'S activities strongly interact. The specification (stated above) of the elements of the SYSTEM MODEL used in the design of the architecture (i.e. Substratum Structure) that could support multi- environmental activities indicates that some kind of Functional Environmental Activity Structures have to be included in the design sequence.

Thus, in terms of Activity Structures Methodology (explained in detail in Kohout et al. 1989; and in the monograph of Kohout, 1989), we have the following sequence of transformation of Activity Structures describing the construction of an IPM:

FEAS ==> FAS -> SUBSTR -> IPM-AS,

where the meaning of the abbreviations is as follows:

FEAS .... Functional Environmental Activity Structure.

FAS .... Functional Activity Structure of IPM.

SUBSTR .... Substratum structure of the architecture (IPM) to be designed.

IPM-AS .... Overall activity structure of the SUBSTR, (i.e. the hierarchy of hardware, firmware and software virtual machines).

The above listed four Activity Structures are used as the framework in which the dynamics of the design features to be simulated is captured. It should be noted that *Requirement knowledge elicitation* stage yields partial FEAS. The purpose of each Activity Structure is defined as follows:

FEAS ... captures the characteristics of the environments, their dynamic features and their changes

FAS ... captures the relevant characteristics of the functions of architectures being designed, and the changes of

its dynamics induced by the designer

SUBSTR .. represents the hardware, firmware and software elements of the architecture and their interactions.

IPM-AS .. captures the relevant characteristic activities and dynamic descriptors of the resulting design alternatives.

These four Activity Structures represent the design knowledge that is to be embedded into the proposed software tool – the simulator.

Suitability of the Activity Structures approach for prototyping and projection of performance of descriptor oriented architectures, as well as of more general well-protected Highly Constrained Multi-Processor Hardware was already demonstrated (Kohout and Mohamad, 1989). Mohamad (1986, unpublished Ph.D) performed a detailed study concerned with the performance of well-protected multi-processor architectures.

Our current objective is to make applicable and further extend the previous work on Activity Structures-based performance evaluation to AI Knowledge-Based Systems (KBS) Architectures. In order to achieve this, it is essential to incorporate into the tool the appropriate knowledge structures, capable of capturing the specific features of the KBS parallel architectures. For this purpose, high level Relational-product based virtual machine structures (Kohout and Bandler 1985) are employed.

The overall aim of our project is to advance the development of a tool for rapid evaluation of performance and protectability of Knowledge-based system architectures during the stage of rapid prototyping. The tool and the supporting methodology are intended to support the following functions:

(a) Performance simulation of a well-protected multi environmental system model which can serve as a basis for the construction of secure and efficient Knowledge-Based Systems for a variety of applications.

(b) Means of representation and design for creating a variety of protected multi-environmental Architectures with ease.

(c) A way of expressing the user requirements of each environment in a way that allows easy evaluation of its influence on the overall performance.

(d) A way of expressing the hardware constraints imposed upon the multi-environmental designs.

(e) A method of evaluating the protected system while in the design stage before commencing its implementation.

It is clear that these objectives are relevant to a progressive and cost effective utilisation of knowledge-based AI techniques in manufacturing industries and financial business.

3 DESCRIPTION OF THE SOFTWARE TOOL

The Activity Structures based approach described in the previous section is supported by the tool consisting of several subsystems. The software tool itself consists of the following:

1. PREPROCESSOR for eliciting the design information from the designers wishing to construct computer architectures based on Activity Structures.

2. EMBEDDED SYSTEM, designed to function as a very general simulation system, that is measurable, portable, with extensible abstract object structures, with functional adjustability.

3. The POSTPROCESSOR, that collects the statistics produced by the embedded simulator system. The statistical data is summarised by this unit, and the details irelevant to a particular purpose are filtered out.

The EMBEDDED SYSTEM consists of:

2.1. A subsystem simulating the machine activities and learning for the purpose of generating the machine environment.

2.2. A highly parametrised shell which provides the designer with the essential design and functionality constraint modules, which embed the functional structures quoted above.

These four modules are linked in a distributed fashion (i.e. by message-passing). For reasons of efficiency, C was chosen as the implementation language of the prototype of the simulator.

3 DESIGN GUIDELINES FOR THE USE OF THE TOOL

The software tool provides the means for effective evaluation of the following factors on the potential or actual design of a Knowledge-Based System:

1) the chosen architectural features
2) the effects of the knowledge base complexity
3) the effects of the dynamic protection chosen
4) the effects of the various user environments
5) the effects of various knowledge structures and contexts.

The simulator thus also allows the the Knowledge Engineer to input and evaluate the essential qualitative differences of various knowledge structures utilised in AI Knowledge-Based Systems, in addition to the differences induced by the multiprocessing features of the distributed architectures alone.

A central part of the ability of the designer to reason from first principles is the skill of using structural and functional descriptions at various levels of abstraction. Hence, the necessary first step in the process of constructing a modern concurrent knowledge-based system, or indeed any well-protected distributed architecture, is to create adequate representations for the system. *Adequate* in our terms are only such representations as can also be used as the input for dynamic simulation of the system interacting with its environment. This should also apply to all the subsequent transformations and creative extensions of the design model, until an effective and verified realization of the design model can be achieved, in the form of the final architecture.

Fig. 1 below shows a typical input from the part two of the tool – the embedded system, which is subsequently used as a raw item of data for the postprocessor. This figure presents the effect of the the type of dynamic protection model of distributed protected objects working in the mode of apparent concurrency on a system architecture supported by the NUKE operating system. Three-level protection policy represents the capability – pass – permit model based on Kohout and Gaines (1975, 1976); Kohout (1978, 1989). It is obvious that the distributed system of dynamically protected objects will give different performance indexes, if the simulation depicts a truly parallel supporting hardware architecture running a distributed operating system instead of NUKE. The simulator can of course be used to compare the differences between the performance of a chosen multi-centre knowledge based system when run in the real concurency mode on parallel hardware, with the case when it is run only in the apparent concurrency mode.

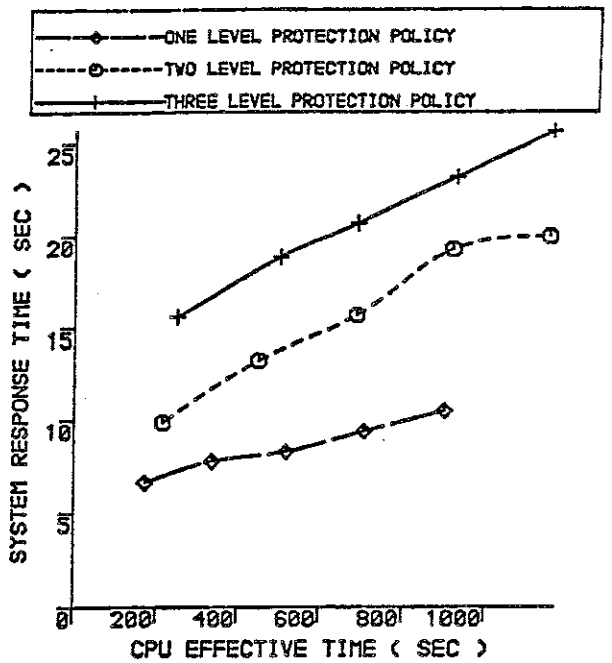


Figure 1:
Effects of changing the protection policy

The strength of our concentration upon functionality stems from the fact that at any stage of the Activity Structures based design process, only the issues of immediate importance are considered. At any stage of construction, the design activity at the current level of abstraction (which we call *current design zone*) serves to identifiy the current design goals and express these by the so called *task descriptors* (Kohout 1978). The design activity in a particular design zone is thus expressed as an object of a certain degree of abstraction. The objects of a particular zone then identify the functional design goals which form the task descriptors of the next, lower design zone.

3 CONCLUSION

We have briefly described the structure of a new design tool that supports utilization of a new methodology for construction of well-protected distributed architectures for knowledge-based systems. A unique point has been taken in our approach, by adopting Activity Structures methodology.

4 REFERENCES

Kohout, Ladislav J., (1986). On functional structures of behaviour. In Kohout, L.J. and Bandler, W., editors, *Knowledge Representation in Medicine and Clinical Behavioural Science*, chapter 7, pages 69–94, Abacus Press, Cambridge MA, U.S.A. and Tunbridge Wells, U.K.

Kohout, L.J., (1989). Activity Structures: A methodology for design of multi-environment and multi-context knowledge-based systems. In Kohout, L.J., Anderson, J., and Bandler, W., editors, *Multi-Environmental Knowledge-Based Systems*, chapter 5, Technical Press (Grower), Aldershot, U.K. in press.

Kohout, L.J., (1987). Activity structures as a tool for design of technological artifacts. *Systems and Cybernetics: An International Journal*, 18(1):27–34.

Kohout, L.J., (1978). Analysis of computer protection structures by means of multiple-valued logics. In *Proc. of the 8th Internat. Symposium on Multiple-Valued Logic*, pages 260–268, IEEE, New York.

Kohout, L.J., (January 1978). *Methodological Foundations of the Study of Action.* Ph.D. Thesis, University of Essex, U.K.

Kohout, L.J., (1976). Representation of functional hierarchies of movement in the brain. *Internat. Journal of Man-Machine Studies*, 8:699–709.

Kohout, L.J., Anderson, J., and Bandler, W., (1989). *Multi-Environmental Knowledge-Based Systems.* Technical Press (Grower), Aldershot, U.K. in press.

Kohout, L.J. and Bandler, W., (1981). Analysis of capability-based computer protection models by means of fuzzy logics. In *Proc. of Eleventh Internat. Symposium on Multiple-Valued Logic*, pages 95–99, IEEE, New York.

Kohout, L.J. and Bandler, W., (1985). Relational-product architectures for information processing. *Information Science*, 37:25–37.

Kohout, L.J. and Gaines, B.R., (1975). The logic of protection. In *Lecture Notes in Computer Science vol. 94*, pages 736–751, Springer Verlag, Berlin – New York.

Kohout, L.J. and Gaines, B.R., (1976). Protection as a general systems problem. *Internat. Journal of General Systems*, 3:1–21.

Kohout, L.J. and Mohamad, S.M.A., (1989). Development of support tools and methodology for design and validation of multi-environmental computer architectures. In Kohout, L.J., Anderson, J., and Bandler, W., editors, *Multi-Environmental Knowledge Based Systems*, chapter 17, Technical Press (Grower), Aldershot, U.K. in press.

Mohamad, S.M.A. and Cavouras, J.C., (1984). Performance study of descriptor oriented architectures. *Computer Performance*, 5:14–22.

A THEORETICAL FRAMEWORK FOR MODELING CHAINS-OF-THOUGHT: AUTOMATING FAULT DETECTION AND ERROR DIAGNOSIS IN SCIENTIFIC PROBLEM SOLVING*

Bienvenido Jose A. Juliano, Jr.** and Wyllis Bandler

Department of Computer Science
and Institute for Cognitive Sciences
The Florida State University
Tallahassee, Florida 32306-4019

ABSTRACT

This paper presents an on-going exploratory study concerning the cognitive processes involved during scientific problem solving. It is conceived that by ascertaining the novice's (in this case, the student user's) *chain-of-thought* , developed while solving a particular problem, a cognitive perspective aiming for a better understanding of the mental processes involved could be achieved. The approximated cognitive structure is correlated with correct or expected versions supplied by the expert (tutor or educator). Such a comparison is asserted to generate valuable information for diagnosis, direction, and eventual correction and improvement of the novice. A theoretical framework, based on a *cognitive map* implementation, is discussed. It attempts to capture the underlying cognitive mechanisms that govern successful fault detection and error diagnosis in problem solving. These formulations are surmised to be significant for possible implementation of diagnostic modules for intelligent tutoring systems that embody the proposed paradigm. We humbly emphasize that this endeavor aims to contribute to a better understanding of the functional aspects of the human mind.

* Part of NSF research project 1327507-41
** Research supported by NSF grant MDR-8609356

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

INTRODUCTION

A growing interest towards problem solving has been evolving among cognitive scientists (Good *et al.* 1986). The understanding and cultivation of successful problem solving techniques has been the primary concern of related studies. These not only involve the instruction of various methodologies and their underlying notions and mechanisms. The diagnosis and possible improvement of effectively utilized approaches are considered as well. Feedback from such diagnoses could aid in the development of more challenging or befitting problems that could be categorized for ameliorating both critical and analytical skills concerned.

In corroboration with the growing interest in cognitive studies related to education, the Florida State University's Department of Computer Science and Institute for Cognitive Sciences launched a research project centering on computerized diagnostic methods for improved instruction. The main objective was to come up with an expert system called DIPS, which stands for Diagnosis for Instruction in Problem Solving. Initially, an effort was made to apply the study in the field of physics (Good *et al.* 1986; Szabo 1987). But with the emergence of valuable data on classical genetics problem solving (Smith 1983; Smith and Good 1984) and the National Science Foundation (NSF) research project, "Computer Diagnosis of Faults in Genetics Problem Solving", a new approach was considered. Details of earlier work pertaining to DIPS can be found in (Good *et al.* 1986; Szabo 1987).

This study is somewhat related to current research (Bandler *et al.* 1988; Mancini and Bandler 1988), that develop and compare various constructs involved in cognitive processes. Other interesting approaches are suggested in various literature (Minsky 1986; Sowa 1984). Some methods for computer coaching (*e.g.* as that in *Wumpus* (Goldstein 1980)) are also available.

CHAINS-OF-THOUGHT

The importance of the *chain-of-thought* concept is made apparent within the context of an idealized *Tutor-Novice Model.* Relative to this model, an analogous system of conceptual structures could be formulated for embodying the concept in a computer. This is the current focus of this research.

Perhaps the most challenging aspect of this study is the fact that numerous problem solving systems have previously been developed, but only a few have emerged for diagnosing the problem solving task (Feigenbaum and Barr 1981; Feigenbaum and Cohen 1982; Goldstein 1980; Newell and Simon 1972). Indeed, developing a system capable of solving a specified class of problems is, in itself, a difficult undertaking. What about a program that assesses problem solving by determining how the problem solver is thinking? Or perhaps a program that approximates the concept formation and manipulation employed by a problem solver in attempting to solve a particular problem? This is exactly how a tutor goes along in directing the course of a session with a novice. A problem is presented for solving. The problem is familiar to the tutor, who associates to it a *solution map* that embodies various solutions known to him. By observing both verbal and non–verbal actions (Ericsson and Simon 1984; Gagne 1966) exhibited by the novice (*e.g.* combinations of written partial solutions, visual contact, bodily movements and others), the tutor detects any faults made by the novice. These are actually deviations from the solution map in his mind and could therefore be diagnosed to generate some appropriate advisory information. The perceived protocol is then noted, synthesized and manipulated by the tutor. The novice, on the other hand, attempts to finally solve the problem successfully based on the hints and comments provided by the tutor.

So, the chain-of-thought referred to here pertains to the line of thinking, or successive linking of concepts utilized to solve a problem. By approximating this structure, the tutor is able to effectively diagnose the problem solving skills of the novice. It is in this manner that our system develops and maintains a *model of the novice*, an important feature provided by our approach.

Cognitive Maps

We now focus our attention on the use of *cognitive maps* (Axelrod 1972; Axelrod 1976; Sowa 1984) in order to capture the essence of the approach just presented. These conceptual structures, introduced by cognitive psychologist E. C. Tolman in 1932 , could be viewed as a collection of nodes semantically linked in a fastidious manner (Hayes 1979; Minsky 1981; Quillian 1967; Schank and Abelson 1977) . Axelrod utilized these structures to represent causality in social scientific knowledge (Axelrod 1976). They facilitated documentary coding by providing symbolic representations of expert documents.

Correspondence with the Tutor-Novice Model

The scientific knowledge with which Axelrod utilized cognitive maps for representation allowed the simple utilization of the (adjacency) matrix to measure causal chaining information and for performing causal reasoning. But in order to capture the mental processes involved in problem solving, as exemplified by the Tutor-Novice Model, a different approach is used here. By taking into consideration the role of the tutor for this model, it is presumed that functionally distinct cognitive maps are maintained within the tutor's mind to successfully carry out diagnostic functions. Furthermore, the representation utilized by Axelrod is insufficient for the evolving or constructive approach outlined earlier. This would result to sparsity.

We conjecture the sufficiency of three cognitive maps. Firstly, a *Domain-Specific cognitive map* that embodies the *generic* (Sowa 1984) conceptual formulations characteristic of the field of discourse under which the problem solving task is to take place (*e.g.* physics, classical genetics, algebra). This represents the tutor's knowledge and expertise in the subject area being considered. Secondly, a *Problem-Specific cognitive map* that incorporates both generic and *referent* (Sowa 1984) concepts involved with the successful solution of a particular problem. This depicts the solution map associated to each of the individual problems presented to the novice. Lastly, the approximation of how the novice is thinking relative to the successful development of a solution to the problem is embodied in what is referred to as the *Novice's Chain-of-Thought cognitive map.*

This approach considers an *overlay* (Goldstein 1980) or *subset* relation between the tutor's expertise and the student's knowledge. This is further exhibited by the structural equivalence of the cognitive maps. The major difference lies in the information encoded within the nodes of each conceptual structure as well as the fact that the model of the student's knowledge is generated during a particular session.

The possibility of utilizing the matrix representation in conjunction with the digraph representation utilizing nodes and arcs is being considered. But current attempts of implementing the system consider only the latter scheme in some specified abstract data type.

Fuzzy Structures

When the complexity of a particular system increases, uncertainty is bound to prevail. This phenomenon applies to society, as well as specific tasks like decision-making and planning, and others. It is also true with problem solving. More complex problems have a higher degree of difficulty, thereby demanding a higher level of intelligence. Hence, when a student is faced with such problems, partial and complete solutions are usually presented with some measure of uncertainty. In such situations, classical bi-valued logic becomes insufficient, if not inappropriate, to capture the intricacies involved.

To capture such imprecision, Zadeh introduced the theory of *fuzzy sets* (Zadeh 1965) . Since then, research on *fuzzification* of formerly *crisp* systems have sprung, particularly on knowledge representation (*e.g.* fuzzy frames (Graham and Jones 1987)) and inductive methods for expert systems. Some recent studies have also shown the plausibility of incorporating causal symbolic inferences with numeric *probabilistic* inference. But we are more interested here in combining symbolic inferences with fuzzy or *possibilistic* inferences or inferences under uncertainty. This latter composition would be appropriate to more situations than the traditional probabilistic or *Bayesian* methods (Bandler and Kohout 1985; Bhatnagar and Kanal 1986) .

Now, consider the scenario when a novice presents some partial or complete solution, preceding the answer with *Maybe* or *I think*. The tutor should readily notice a lower degree of certainty associated with such responses. But how is this to be represented within the cognitive maps? It is asserted that cognitive maps could be fuzzified by introducing strength measures on relational links connecting the semantically rich conceptual nodes of a map. A corresponding *fuzzy cognitive map algebra* (Kosko 1986) could be developed to precisely define inference modalities within the structures.

In connection with the conceptual structures used in this study, only the *Novice's Chain-of-Thought* cognitive map will be fuzzified. The other two maps remain crisp since they are already established facts and are expected to be precise and exact. The measures of strength are to be assigned by some linguistic variable (Zadeh 1979) interpreter that would approximate these values. Certainly, a problem arises here — the various ways to implement such an interpreter to capture the very great variety of discriminant principles utilized by tutors (Bandler and Kohout 1980a; Bandler and Kohout 1980b) . This issue is crucial for implementing these cognitive structures, most specially for the map-matching algorithm that would generate diagnostic information based on the performance of the novice.

We point out the issue of formulating a plausible measure of the *degree of subsethood* relation between pairs of cognitive maps under consideration. Such a measure of uncertainty has been suggested (Bandler and Kohout 1980) in terms of the possibility, π, and is defined as

$$\pi(A \subseteq B) = \mu_{\mathcal{P}(B)}A$$

where $\mu_{\mathcal{P}(B)}A$ denotes the *membership function* that maps A to the *power set* of B. The value returned by $\mu_{\mathcal{P}(B)}A$ would depend on the *fuzzy implication* operator utilized to embody the said mapping.

The Tri-Map Configuration

Relative to the cognitive maps discussed earlier, this study further asserts that successful automation of fault detection and error diagnosis of problem solving skills greatly depends on some operational and relational manipulations invoked by a predetermined map-matching algorithm. Theoretically, the cognitive maps must be accessible to the algorithm to allow the performance of specific functions necessary to generate the appropriate diagnostic results. It is envisioned that, at least conceptually, some form of *Tri-Map Configuration* should be established to facilitate overlay-like comparisons between the distinct cognitive maps involved.

Fault Detection

If the cognitive maps were to be intuitively characterized as opaque nodes interlaced on a translucent plane and the algorithmic manipulations as a light source, then fault detection becomes straightforward. Passing this "light" vertically through the cognitive structures, positioned as in a Tri-Map Configuration, would detect deviations that represent faults. These *negative* or *anti*-nodes must be noted for interpretation. This suggests a correspondence with *differential* models that

recognize and summarize the student's behavior (Dede 1986). Some mathematical formulations could be made for *discrepancy* operations that embody this approach, and these are currently being considered.

Alternate solutions presented by the student user are considered. We emphasize that these deviations from the methodology expected by the expert, which may actually represent better solutions, could be investigated by systems utilizing this approach via backward chaining guided by the *Domain-Specific* and *Problem-Specific* cognitive maps.This provides a more user-friendly system behavior geared towards motivation.

Error Diagnosis

Equally important to the function of fault detection is the diagnosis of the error detected. The conglomeration of negative nodes, representing deviations from the idealized paths, may be categorized for error classification. Since the nodes are semantically rich, diagnosis is performed by extracting the required information from the nodes concerned. It is pointed out that the tutoring capability of the system is derived from this information. The problem of credit/blame appointment is also resolved, specially with the fuzzified approach, similarly with the so–called *perturbation* models (Dede 1986). This also suggests an effective pedagogical mechanism for the system that could hopefully induce learning within the subject domain concerned.

AN OPEN INVITATION

What useful information could be generated by implementing such an approach? Firstly, from a psychological viewpoint, it is presumed that a better understanding of a specific class of mental processes could be achieved (Greeno 1980; Minsky 1986; Newell and Simon 1972; Schoenfield 1984) . This should prove to be helpful for the unending quest to gain a deeper understanding of the human mind and its intricacies. Secondly, from the educator's viewpoint, it could provide information regarding measures of strengths of currently existing curricula (Reif 1980; Simon 1980). This could also indicate certain flaws in the method of instruction being utilized. Last, but not the least, we cannot overlook its inherent tutoring capability. This is important for improving an individual's problem solving skills.

We have remained silent in suggesting other areas where this framework could be applied. Surely, pattern recognition, fault detection and error diagnosis are areas of interest. We have also been considering abductive systems, as well as other plausible possibilistic formulations for the fuzzy case. Such conceptions necessitate further investigation of methods for integrating symbolic causal inference with fuzzy inference in the context of the Tri-Map configuration discussed in this study. All these, and perhaps more, provide an open invitation for future research.

REFERENCES

Axelrod, R. M. 1972. *Framework for a General Theory of Cognition and Choice.* U.C. Berkeley Institute of International Studies, Berkeley, California.

Axelrod, R. M. 1976. *Structure of Decision: The Cognitive Maps of Political Elites.* Princeton University Press, Princeton, New Jersey.

Bandler, W. and L. J. Kohout. 1980. "Fuzzy Power Sets and Fuzzy Implication Operators". *Fuzzy Sets and Systems* 4: 13–30.

Bandler, W. and L. J. Kohout. 1980. "Semantics of Implication Operators and Fuzzy Relational Products". *Int. J. Man-Machine Studies* 12: 89–116.

Bandler, W. and L. J. Kohout. 1985. "Probabilistic vs. Fuzzy Production Rules in Expert Systems". *Int. J. Man-Machine Studies* 22: 347–351.

Bandler, W., V. Mancini, and L. J. Kohout. 1988. "Investigating the Structures of Knowledge and Thought". Presented at ICSRIC '88 (4^{th} International Conference on Systems Research, Informatics and Cybernetics), August 16-22, 1988, Baden Baden, DBR, and to appear in *Proceedings* thereof.

Bhatnagar, R. K. and L. N. Kanal. 1986. "Handling Uncertain Information: A Review of Numeric and Non-numeric Methods". In *Uncertainty in Artificial Intelligence*, L. N. Kanal and J. F. Lemmer, eds. North-Holland Publishers, New York, 3–26.

Dede, C. 1986. "A Review and Synthesis of Recent Research in Intelligent Computer-Assisted Instruction". *Int. J. Man-Machine Studies* 24: 329–353.

Ericsson, K. A. and H. A. Simon. 1984. *Protocol Analysis: Verbal Reports as Data.* The MIT Press, Cambridge, Massachusetts.

Feigenbaum, E. A. and A. Barr. 1981. *The Handbook of Artificial Intelligence, Vol. I.* Addison-Wesley Publishing Co., Reading, Massachusetts.

Feigenbaum, E. A. and P. R. Cohen. 1982. *The Handbook of Artificial Intelligence, Vol. II.* William Kaufmann, Inc., Los Altos, California.

Gagne, R. M. 1966. "Human Problem Solving: Internal and External Events". In *Problem Solving: Research, Method and Theory*, B. Kleinmutz, ed. John Wiley & Sons, Inc., New York.

Goldstein, I. 1980. "Developing a Computational Representation of Problem-Solving Skills". In *Problem Solving and Education: Issues in Teaching and Research*, D. T. Tuma and F. Reif, eds. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 53–79.

Good, R., R. Kromhout and W. Bandler. 1986. "Human and Machine Diagnosis of Scientific Problem–Solving Abilities". *Journal of Research in Science Teaching* 23: 263–275.

Graham, I. and P. K. L. Jones. 1988. "A Theory of Fuzzy Frames". In *Research and Development in Expert Systems IV*, D. S. Moralee, ed. Cambridge University Press, Cambridge, England, 76–88.

Greeno, J. G. 1980. "Trends in the theory of knowledge for problem solving". In *Problem Solving and Education: Issues in Teaching and Research*, D. T. Tuma and F. Reif, eds. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 9–24.

Hayes, P. J. 1979. "The Logic of Frames". In *Frame Conceptions and Text Understanding*, D. Metzing, ed. Walter de Gruyter and Co., Berlin, 46–61.

Kosko, B. 1986. "Fuzzy Cognitive Maps". *Int. J. Man-Machine Studies* 24: 65–75.

Mancini, B. and W. Bandler. 1988. "Congruence of Structures in Urban Knowledge Representation". Presented at IPMU '88 (International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems), July 4-7, 1988, Urbino, Italy, and to appear in *Proceedings* thereof.

Minsky, M. 1981. "A Framework in Representing Knowledge". In *Mind Design*, J. Haugeland, ed. The MIT Press, Cambridge, Massachusets, 95–128.

Minsky, M. 1986. *The Society of Mind.* Simon and Schuster, New York, New York.

Newell, A. and H. A. Simon. 1972. *Human Problem Solving.* Prentice-Hall, Englewood Cliffs, New Jersey.

Quillian, M. R. 1967. "Word Concepts: A Theory and Simulation of Some Basic Semantic Capabilities". *Behavioral Science* 12: 410–430.

Reif, F. 1980. "Theoretical and Educational Concerns with Problem Solving: Bridging the Gaps with Human Cognitive Engineering". In *Problem Solving and Education: Issues in Teaching and Research*, D. T. Tuma and F. Reif, eds. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 39–50.

Schank, R. C. and R. Abelson. 1977. *Scripts, Plans, Goals and Understanding.* Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Schoenfield, A. H. 1984. *Cognitive Science and Mathematical Education.* Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Simon, H. A. 1980. "Problem Solving and Education". In *Problem Solving and Education: Issues in Teaching and Research*, D. T. Tuma and F. Reif, eds. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 81–96.

Smith, M. U. 1983. *Comparative Analysis of the Performance of Experts and Novices While Solving Selected Classical Genetics Problems.* PhD thesis, Florida State University, Tallahassee, Florida.

Smith, M. U. and R. Good. 1984. "Problem Solving and Classical Genetics: Successful vs. Unsuccessful Performance". *Journal of Research in Science Teaching* 21: 895–912.

Sowa, J. F. 1984. *Conceptual Structures: Information Processing in Mind and Machine.* Addison-Wesley Publishing Co., Reading, Massachusetts.

Szabo, S. 1987. *A Design for a Diagnostic Expert System in the Field of Education.* Master's thesis, Florida State University, Tallahassee, Florida.

Tolman, E. C. 1967. *Purposive Behavior in Animals and Men.* Appleton-Century-Crofts, New York, New York.

Zadeh, L. A. 1965. "Fuzzy Sets". *Information and Control* 8: 338–353.

Zadeh, L. A. 1979. "A Theory of Approximate Reasoning". In *Machine Intelligence* Vol. 9, J. Hayes, D. Michie, and L. I. Mikulich, eds. Halstead Press, New York, New York, 149–194.

AN APPROXIMATE REASONING SYSTEM BUILDING TOOL[1]

Abraham Kandel, Zhiqiang Cao and Jie Feng
The SUS Center for Artifical Intelligence
and The Department of Computer Science,
Flordia State University
Tallahassee, Florida 32306-4019

1. Introduction

Over the past twenty years, the seminal work of Zadeh[7,8] has stimulated widespread interest in the modeling of human reasoning process by means of fuzzy mathematics. One of the most important developments is approximate reasoning [5,?], which was defined by Zadeh[12] as "the process or processes by which a possibly imprecise conclusion is deduced from a collection of imprecise premises". In this work we present an approximate reasoning system building tool which cooperates with the human expert to learn and improve the definition of linguistic descriptions and fuzzy inference rules. Since a small diversity in the meaning of these definitions can result in totally different reasoning result, the precision of fuzzy definition is crucial. To achieve a reasonably accurate precision, a new inference model[2] with better precision is employed; analysis on the fuzzy definition using the concept of fuzzy expected value (FEV) [4] is provided to assist the user in finding a better representation; a test execution against simulated data is also provided for evaluating the performance; an interactive graphics interface is designed for visual verification of the definition and performance; the process is repeated until the user is satisfied.

[1]This research was sopported in part by NSF grant IST 8405953 and by the Florida High Technology and Industry Council grant UPN 85100316

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

2. ARSBT, An Approximate Reasoning System Building Tool

The ARSBT system is an approximate reasoning system building tool based on the new model of fuzzy reasoning [?]. It provides menu-driven and restricted natural language interface and graphics to users for easy, flexible consultation and accurate feedback on fuzzy definitions and overall system performance. The system is designed to provide the following functions:

- 1. learning the definition of a fuzzy relation, the linguistic descriptions, the semantics of these linguistic descriptions through the definition of fuzzy set, the linguistic inference rule in the form of "*IF premise THEN conclusion WITH certainty factor*". It also provides a simulated run of the approximate reasoning over the entire interval which is covered by the fuzzy set definitions of those linguistic inference rules.
- 2. modifying the definition of the semantics of those linguistic descriptions, updating the linguistic inference rules and their certainty factors.
- 3. reasoning according to the knowledge learned. It provides quantitative output as well as graphical output.

There are three subsystems in this ARSBT system. They are Knowledge Acquisition Subsystem, Knowledge Modification Subsystem and Knowledge Reasoning Subsystem. The ARSBT system architecture is shown in Figure 1.

In the design and implementation of the ARSBT user interface, several safety checks are provided to prevent an unpleasant break down of the system. Whenever user enters input data, the system automatically performs a typing check and ranging check to avoid any possible typing mistakes and out of range data. A warning message

like "Illegal Input, "Input data out of range" is issued if a typing or ranging error were to occur. Color graphics are utilized to provide the best visual representation of the fuzzy definition and reasoning results. System messages, system warnings, user entries and graphics are shown in different colors for clearer visual identification.

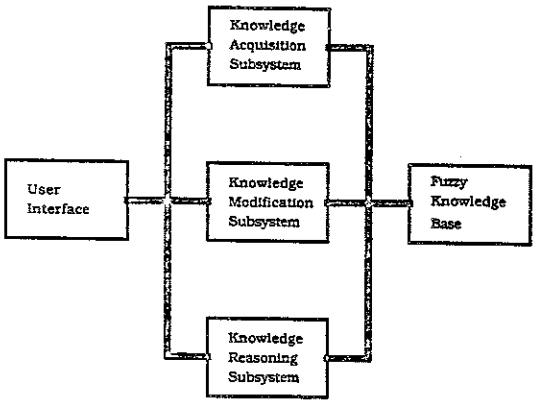


Figure 1. The ARSBT system architecture

In what follows, the functions and designs of each subsystem and the user interface are discussed.

The Knowledge Acquisition Subsystem

The Knowledge Acquisition Subsystem (KAS) is designed to have the capability of learning fuzzy relations and fuzzy descriptions as well as testing the knowledge learned. The functions provided by KAS are

- 1. learn the number of variables, their names, their relation and their ranges.
- 2. learn the linguistic descriptions for each variable.
- 3. provide graphics tool for user to enter the coordinates and membership to define membership function, or
- 4. provide graphics tool for user to compose membership function to define the meaning of the linguistic descriptions.
- 5. show the definition for a linguistic description in graphics with analysis of the fuzzy set.
- 6. show the definitions of all the linguistic descriptions in graphics.

The learning is carried out by an interactive process between user and the system. Feed back on the information learned and the reasoning result is provided in the form of both quantitative measures and graphics representation to assist the human expert in finding a reasonably accurate representation of his or her knowledge. System loops to any of the above functions are also provided for modification until the human expert is satisfied with the representation. In building an approximate reasoning system, all parameters associated with the system, such as definition of relations, definition of fuzzy meaning of a term and fuzzy inference knowledge, need to be determined before the structure of the system is established and tested. This task is accomplished by interactive learning between human expert and the ARSBT system. The information need to be learned from human expert is:

- 1. the names and ranges of variables involved in the functional relation $y = f(x_1, x_2, \ldots, x_n)$.
- 2. the linguistic descriptions associated with variable Y and X_i, where $i = 1, \ldots, n$.
- 3. the fuzzy set definitions of these linguistic descriptions for y and $x_i, i = 1, \ldots, n$.
- 4. the linguistic inference rules on the relation between y and x_i, where $i = 1, \ldots, n$ and their certainty factors.

The following is a sample dialogue between a human expert and the system:

How many variables do you have in the system ?
2
What is the name of variable number 1 ?
x
What is the name of variable number 2 ?
y
Do you mean $y = f(x)$? [Y/N]
y
What is the range of x ?
Range Starts at : *0.0*
Range Ends at : *10.0*
How many verbal descriptions do you have for x ?
7
Please enter your verbal description-1 for x.
small
Please enter your verbal description-2 for x.
very close to 5
...

After learning the information about the functional relation and the linguistic descriptions for each variable, the ARSBT system proceeds to ask for the fuzzy definition for each verbal description. To define the semantics of a linguistic description, a fuzzy membership function is needed. The ARSBT system has provided the user with two methods to choose from. In the first approach, a user may choose to describe the membership function by coordinates and memberships, i.e., describe the function point by point. In the second approach, a user may select parts of a membership function and then assemble them. The shape of a part is determined by several parameters. By modifying these parameters, a user may change the distribution of the function, which is illustrated in graphics, to achieve the best representation of his or her knowledge. The eight membership function parts are shown in Figure 2. The assembly method also starts with a screen of eight function parts and then a dialogue:

Your membership function can be constructed by some of the above curves. A membership function in this system has three parts: LEFT, MIDDLE and RIGHT.

Enter [start, end] interval for part - 1.
It starts at : 0
It ends at : 3

Enter your choice of function for part -1 :2
Enter [start, end] interval for part - 2.
...

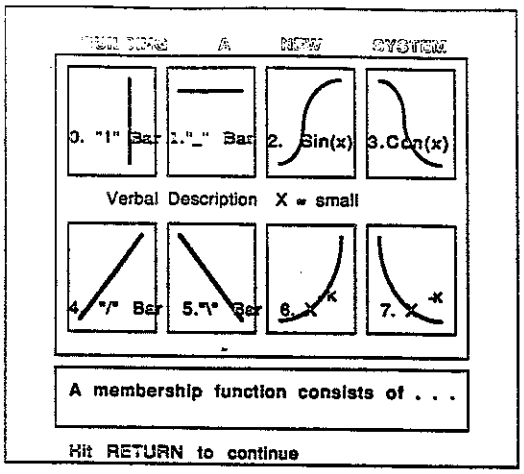


Figure 2. The function parts in the ARSBT system for composing a membership function

After the dialogue, the system will demonstrate the membership function in graphics together with the analysis and will leave the evaluation of the new function to the human expert. At this time the expert can visually verify the actual distribution of the function and the three semantic measures indicating important characteristics of the meaning. Now the expert is giving a choice to modify if he is not satisfied. Otherwise he may choose to proceed to the definition of the next term.

The learning of inference rules is carried out through an interactive learning module. Since, in general, the number of verbal descriptions for each variable is finite, the relation between verbal descriptions of two variables x and y can be described by a matrix. The learning of the relation matrix is through an interaction with a menu-driven interface like the follows:

IF x=very large, then
In what degree y=very small :0.00
In what degree y=small: 0.00
In what degree y=medium: 0.95
In what degree y=large :0.05
In what degree y=very large:0.00
In what degree y=very very large:0.00

Once the knowledge is learned, it will be stored in the ARSBT's knowledge base. A test on the knowledge is provided for the human expert to examine the performance of the knowledge learned.

Semantic Analysis

The system provides a certain semantic analysis. The semantic analysis is carried out by calculating three quantitative measures and by illustrating the fuzzy set distribution through graphics. The three quantitative measures are : the defined interval or the range of the membership function, the typical element of the function, the peak point. The qualitative measure mentioned above is a graphics representation of the population distribution of the fuzzy set. With these measures, we are able to characterize the overall distribution interval, a typical member and the most important member of the set, which we think best describes some of the most important features of the distribution of a fuzzy concept in an expert's mind.

In the new model of fuzzy reasoning, the concept *cover* of a set G is introduced: a group of fuzzy sets $\{A_1, A_2, \ldots, A_n\}$ is a cover of G if and only if $L(x) = \{\mu_{A_1}(x), \mu_{A_2}(x), \ldots, \mu_{A_n}(x)$ is a non-zero vector, where $\mu_{A_1}, \mu_{A_2}, \ldots, \mu_{A_n}$ is the membership functions of $A_1, A_2, ..., A_n$. This new inference model requires, for each variable, its range must be covered by its verbal descriptions. Otherwise, i.e., if an interval is not covered, any real point in that interval will have identical all-zero vector. Consequently, its inference result will be zero such that we are unable to predict the difference of reasoning results caused by different inputs. Obviously, this requirement is reasonable. Since a system provides a feedback mechanism, the performance of the inference rules as well as the fuzzy descriptions can be shown by graphics and therefore the characteristics of the definition can be visually verified. If some interval is not covered, a change can easily be made by a call to the Modifying Module.

The Knowledge Modification Subsystem

The Knowledge Modification Subsystem (KMS) is another important one in the ARSBT system. It is designed to have the capability of modifying the fuzzy definition of linguistic descriptions for each variable and the fuzzy inference rules. Its user interface is also menu-driven together with the graphics tool to provide easy and flexible interactions. The functions provided by KMS are:

- 1. modify the fuzzy set definition of linguistic description. The functions in this part include :

  a) show the fuzzy definitions of all the linguistic descriptions in graphics and ask the user whether the definitions are satisfactory.

  b) modify the fuzzy definition by either assembling a new membership function or by entering new coordinates for the function.

  c) show the new definition in graphics with analysis.

  d) show all the new fuzzy definitions in graphics.
- 2. modify the fuzzy inference rule base

  a) show the original rule and its certainty value in the form:

  IF x=description1 THEN

  In what degree y=description2 (old-certainty):

  a new value or a return is expected from the user.

  b) remember the new certainty value or keep the old one if the certainty value is satisfactory.
- 3. quit from the modification subsystem and return back to the system menu.

The modification menu provides the user with the above options. A run on some simulated data gives the user a sense of how well the system performs. An examination of the definition of linguistic descriptions and inference rules can assist the user in improving the coherence of the rules as well as the terms defined.

The Knowledge Reasoning Subsystem

The Reasoning Subsystem (KRS) is responsible for taking input data, converting it into necessary internal form, applying the fuzzy inference rules, calculating the corresponding output, illustrating the reasoning results in graphics and providing a brief explanation about the changes on the result curve. It can take input from the user entry or generate simulated input data by itself. The functions provided by KRS are:

- 1. reason according to the input data provided by user, or
- 2. reason according to the simulated input data over the entire defined interval,
- 3. demonstrate the reasoning results by color graphics,
- 4. briefly explain the trend of changes of the curve of reasoning results,
- 5. store the reasoning results in an external file for further study.

A dialogue starts with a question asking the user to choose one of the function 1 and 2 shown above. After preparing the input data the system starts reasoning against the data and shows the result curve in graphics. At the request of the user, the system provides a brief explanation on the result. While the user is checking the reasoning result, a sorted external data file is created simultaneously. The user may later study this file of both input and output data. If unsatisfied, the user may choose to examine or even to modify the related definitions and rules. Again the precision of the system being built is under full control of the user.

Performance

The performance of the system has been satisfactory in terms of reasoning precision, ease of use and intuition of the graphics representation. The reasoning precision is shown by the smooth final result curve. As compared in paper [2] the precision of the new inference model is superior to others. With semantic analysis and interactive modification, the remaining error can be reduced until a user is satisfied. The interface of the ARSBT system proves to be user-friendly. This is demonstrated by the fact that a user may change

mode from learning to reasoning or from reasoning to modifying at a touch of a key. All dialogues of the interface are in plain English and come with easy-to-follow instructions. The system questions, system instructions, system warnings and user input are all shown in different colors. Moreover the interface is designed to be robust to virtually any input. The check on the type and range of input data prevents the system from crashing by an accidental typing error. The graphics representation gives the user an intuitive illustration of all the definitions of linguistic descriptions, inference rules and reasoning results. Since the relation between the definition of meaning and the reasoning error and the relation between inference rules and the system performance can be visually verified, the control over the remaining error becomes possible.

3. Conclusion

The ARSBT system is an approximate reasoning system building tool based on a new inference model. It is composed of three subsystems: knowledge acquisition, knowledge modification and knowledge reasoning. Its menu-driven, restricted natural language and graphics interface has the distinct advantage of flexibility. The fact that typing and ranging checks are performed on all data entry greatly increases the reliability of the system. The ability of semantic analysis provides the user with effective measures in evaluating the machine representation of human knowledge. Its implementation has achieved the goal of a approximate reasoning with desirable accuracy, which has demonstrated the applicability of fuzzy reasoning.

References

[1] Z. Cao and A. Kandel, *Applicability of some fuzzy implication operators,* Fuzzy Sets and Systems, vol. 30 (1989), 1-36.

[2] Z. Cao, A. Kandel and L. Li, *A new model of fuzzy reasoning,* forthcomming.

[3] A. Kandel, *Fuzzy sets, fuzzy algebra and fuzzy statistics,* Proceeding of the IEEE, vol. 66, no. 12, December (1978), 1619-1639.

[4] A. Kandel, *Fuzzy mathematical techniques with applications,* Addison-Wesley Publishing Company, Reading, Massachusetts. (1986).

[5] E.H. Mamdani, *Applications of fuzzy algorithms for control of simple dynamic plant,* Proceedings of IEEE, vol. 121 (1974), 1585-1588.

[6] M. Mizumoto and H.J. Zimmermann, *Comparison of fuzzy reasoning methods,* Fuzzy Sets and Systems, vol. 8 (1982), 253-283.

[7] L.A. Zadeh, *From circuit theory to system theory.* Proc. Institute of Radio Engineers, vol. 50(1962), 856-865.

[8] L.A. Zadeh, *Fuzzy sets.* Information and Control, vol. 8 (1965), 338-353.

[9] L.A. Zadeh, *Fuzzy algorithm.* Information and Control, vol. 12 (1968), 94-102.

[10] L.A. Zadeh, *Outline of a new approach to the analysis of complex systems and decision processes,* IEEE Trans. Systems Man Cybernet., vol. 3 (1973), 28-44.

[11] L.A. Zadeh, *The concept of a linguistic variable and its application to approximate reasoning,* Information Science, vol. 8 (1975).

[12] L.A. Zadeh, *A theory of approximate reasoning,* in (J.E. Hayes et al eds.) Machine Intelligence(John Wiley and Sons, New York, 1979), 149-194.

A FIRST ORDER LOGIC OF CHRONOLOGICAL IGNORANCE*

Alexander W. Kirmse and Daniel G. Schwartz
Department of Computer Science and
SUS Center for Artificial Intelligence
The Florida State University
Tallahassee, Florida 32306

ABSTRACT

This paper presents an extension CI^* of Shoham's logic of chronological ignorance CI [SHOHAM 88] to the first-order case. The arguments leading to use of a temporal non-monotonic modal logic for the prediction task are outlined and the syntax and semantics of CI^* are specified. Causal theories are identified within CI^* and shown to have a unique chronologically most ignorant model.

INTRODUCTION

Time and change are crucial to the human perception of the real world. Classical logics do not associate formulae with intervals in time. Temporal logics have been proposed in [McDERMOTT 82], [ALLEN 84], [SHOHAM 88], and others. This paper presents Shoham's way of deriving a temporal logic to argue about change and extends his logic of chronological ignorance to the first-order case. We will adopt all of Shoham's notational conventions.

The Structure of Time

The logic of chronological ignorance (CI) as presented by Shoham and the extension presented below rely on a very pragmatic view of time. Time is assumed to be linear, discrete and unbounded in both the past and the future. Time *points* are assumed to be atomic to time, not *intervals*. It is modeled by the set of whole numbers. The $\leq$-operation with its common interpretation provides the required order on time. This concept of time is used in all digital clocks–however, with lower and upper bounds due to physical limitations. Since our daily life seems to function well with those, its adoption here seems justified.

* This work was supported by the Office of Naval Research, Grant No. N00014-87-G0219.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

A First-Order logic of Time Intervals

In an initial attempt at providing an adequate temporal logic, formulae in first-order predicate calculus (FOPC) can be associated with an interval in time denoted by its beginning and end point. Let $\mathbf{r}$ be an n-ary relation symbol, the $ntrm_j$ non-temporal terms, and i an interval in time. Then a standard predicate

$$\mathbf{r}\ (ntrm_1, ...,\ ntrm_n)$$

becomes the pair

$$(i,\ \mathbf{r}\ (ntrm_1, ...,\ ntrm_n)).$$

According to the discussion above, i is denoted by $(t_1,\ t_2)$, its beginning and end points, leaving us with

$$((t_1,\ t_2),\ \mathbf{r}\ (ntrm_1, ...,\ ntrm_n)).$$

In keeping with the notation used by McDermott and Allen, but using only one predicate-type (as opposed to Allen's HOLDS, OCCURS and OCCURRING), the standard well-formed formula (wff) will be

$$\mathbf{TRUE}\ (t_1, t_2, \mathbf{r}\ (ntrm_1, ...,\ ntrm_n)).$$

For example, the statement

$$\forall x\ (\mathbf{BRAND}\ (x,\ Lamborghini) \supset \mathbf{FAST}\ (x))$$

can be modified into

$$\forall x\ [\mathbf{TRUE}\ (1963, 1988, \mathbf{BRAND}\ (x, Lamborghini)) \supset \mathbf{TRUE}\ (1963, 1988, \mathbf{FAST}\ (x))]$$

keeping one from falsely associating the predicate **FAST** with Ferruccio Lamborghini's pre-1963 products, all of which were agricultural tractors.

THE QUALIFICATION PROBLEM

The logic being developed should be suitable to argue about change and to make predictions. It takes great effort to do this with the above logic. Take the example of starting a vehicle. From

TRUE $(1, 3,$ **ACTIVE** $(ignition))$ and
TRUE $(7, 8,$ **ACTIVE** $(starter))$

one would like to conclude

TRUE $(9, 10,$ **RUNNING** $(engine))$

As almost everyone has experienced, a dead battery, flooded carburetor, or other adverse mundane conditions can interfere with the above. Deciding which conditions to consider in making an inference precise, yet efficient, is called the *qualification problem*. Adding the negation of all possible adverse conditions to the left-hand side of an inference helps, but makes the inference computationally expensive

Another problem is whether
TRUE $(7, 8,$ **ACTIVE** $(ignition))$ is a valid derivation from the given axioms. More generally, the question is to what extent into the future predictions can be made. This *extended qualification problem* and the suggested cure through potential histories as suggested by Shoham will not be addressed here. Frame axioms such as

TRUE $(n, n,$ **ACTIVE** $(ignition)) \wedge$
$\neg$**TRUE** $(n, n,$ **REMOVED** $(ignition\text{-}key)) \wedge$
$\neg$**TRUE** $(n, n,$ **DEFECTIVE** $(fuse)) \wedge \ldots \supset$
TRUE $(n+1, n+1,$ **ACTIVE** $(ignition))$

will bridge the gap.

Non-Monotonicity

The former problem provides a scenario where a set of facts A entails a conclusion C, but the extended set $A \wedge B$ does not (if B is of the "dead battery"-type).

$$A \models C \text{ , but } A \wedge B \not\models C$$

This leads to the use of a non-monotonic logic, where a *preference relation* $\sqsubset$ on the set of possible models is provided. The familiar notion of satisfaction is replaced by that of *preferential satisfaction*, where a model M preferentially satisfies A if M satisfies A and there is no model M' preferable to M that also satisfies A. This notion is non-intuitive, but well-defined.

CHRONOLOGICAL IGNORANCE

The preference relation on the set of potential models can be custom-tailored to suit the purpose of a particular logic. For the prediction task, those models are preferable where adverse conditions occur as early as possible, so that at a later time the prediction can be made without problems. The preference relation $\sqsubset$ is defined as follows:

Definition 1. *Let $S = \{\varphi_1, ..., \varphi_n\}$ be a set of formulae and M_1 and M_2 two interpretations. M_2 is chronologically smaller in S than M_1 if there exists a time t_0 such that*

1. *for all $\varphi \in S$ and all $t_1, t_2 \leq t_0$, if $M_2 \models$* **TRUE** (t_1, t_2, φ) *then also $M_1 \models$* **TRUE** (t_1, t_2, φ), *and*
2. *there exists an $\varphi \in S$ and a $t \leq t_0$ such that $M_1 \models$* **TRUE** (t, t_0, φ) *but $M_2 \not\models$* **TRUE** (t, t_0, φ).

This could now be applied to *chronologically minimize* all the adverse conditions mentioned in the example above. The set S would contain all the dead batteries, missing coils, etc., and preferred models would be those minimal in S. In these models, all the "good" things would happen: **TRUE** $(1, 3,$ **ACTIVE** $(ignition))$ would propagate to **TRUE** $(8, 8,$ **ACTIVE** $(ignition))$ by use of the frame axiom. Together with
TRUE $(7, 8,$ **ACTIVE** $(starter))$ this would allow the prediction of **TRUE** $(9, 9,$ **RUNNING** $(engine))$.

The strategy outlined above requires manual selection of S. Since the resulting logic should be suitable for mechanization, and the selection of S requires intuition, the result is not satisfactory. So far, there is essentially a different logic for each scenario.

A nice solution is to simply have S include all formulae. The preferred models then have minimal occurrences of dead batteries, flooded carburetors and anonymous phone calls, where one needs not care that the latter has nothing to do with starting a car.

Modal Operators

A hindrance is that for any wff φ in a logic $\mathcal{L}$ with preference relation $\sqsubset$ either φ or $\neg\varphi$ holds true. It is not possible to minimize all the φs or $\neg\varphi$s which actually happen. A different view of what to minimize is necessary. One opts to minimize over the wffs φ which one *knows* to have occurred. This alteration provides the excluded middle: not knowing φ does not bring about $\neg\varphi$.

The needed tool is found in modal logics, which provide a Kripke structure instead of a single model as an interpretation. A Kripke structure provides a set $\mathcal{W}$ of possible *worlds*. Each of the elements in this set is a classical interpretation. The criteria for satisfaction of a formula φ with respect to a single world W within $\mathcal{W}$ are similar to those in standard FOPC. The $\Box$-operator allows statements interpreted in W to be made about all worlds *accessible* from W.

Let each world in the Kripke structure (KS) represent a possible course of the universe. For any wff φ, it is either true or false in every one of the worlds at each point in time. Assuming all worlds fall into the same equivalence class, *known facts* are asserted by $\Box\varphi$, stating that φ holds in every one of the worlds in $\mathcal{W}$. The diamond operator $\Diamond$ is defined by $\Diamond\varphi \equiv \neg\Box\neg\varphi$, expressing that no knowledge exists to the contrary of φ.

The predictions to be made will be of the following kind: if these facts are known at these times and it is not known that certain other facts are true, then a specific fact is known at a *later* time. In a chronologically minimal model, i.e. a preferred model with respect to $\sqsubset$ here, of

$\Box($$n$, n, **ACTIVE** $(ignition)) \wedge$
$\Box($$n$, n, **ACTIVE** $(starter)) \wedge$
$\Diamond($$n$, n, **DEFECTIVE** $(battery)) \wedge$
$\Diamond$ {negation of other adverse conditions} $\supset$
$\Box($$n+1$, $n+1$, **RUNNING** $(engine))$

all the "good" things happen and the adverse ones do not. Knowledge of an adverse condition φ', however, prevents the prediction from being made since $\Diamond\varphi'$ will no longer be valid.

THE LOGIC CI^*

With the motivation and tools from the previous sections the logic CI^* can now be specified.

Syntax. Given the following domains of temporal symbols:

TC: a set of time points,
TV: a set of time variables, and
TF: a set of temporal function symbols,

the set of temporal terms can be defined as follows:

1. All elements in TC are temporal terms.
2. All elements in TV are temporal terms.
3. Let $ttrm_1, ..., ttrm_n$ be temporal terms, and let f be an n-ary temporal function symbol, then $f(ttrm_1, ..., ttrm_n)$ is a temporal term.
4. There are no other temporal terms.

Given these domains for the non-temporal symbols:

C: a set of constants, disjoint from TC,
V: a set of variables, disjoint from TV,
F: a set of function symbols, disjoint from TF, and
R: a set of relation symbols,

the set of non-temporal terms is defined in complete analogy to the set of temporal terms, substituting C, V, and F for TC, TV, and TF, respectively.

With this, the set of wffs is then defined as:

1. Given temporal terms $ttrm_1$ and $ttrm_2$, then $ttrm_1 = ttrm_2$ and $ttrm_1 \preceq ttrm_2$ are wffs.
2. Given temporal terms $ttrm_1$ and $ttrm_2$, non-temporal terms $trm_1, ..., trm_n$, and an n-ary relation symbol r $\in$ R, then **TRUE** $(ttrm_1, ttrm_2, \mathbf{r}\ (trm_1, , trm_n))$ is a wff.
3. Given wffs φ and φ', then $\varphi \wedge \varphi'$ and $\neg\varphi$ are wffs.
4. Given wff φ and a variable $v \in$ TV $\cup$ V, then $\forall v\ \varphi$ is a wff.
5. Given wff φ, then $\Box\varphi$ is a wff.
6. There are no other wffs.

The above provides sufficiently many logical connectors. For notational convenience the usual definitions of $\vee$, $\supset$, $\exists$ and so on are assumed in terms of the above. $\Diamond\varphi$ is defined as $\neg\Box\ \neg\varphi$. As used in an example above – apologies to the reader for not introducing it first – $\Box$**TRUE** $(t_1, t_2, \mathbf{r}\ (...))$ can be abbreviated to $\Box(t_1,\ t_2,\ \mathbf{r}\ (...))$.

Semantics. In FOPC a single interpretation is sufficient to provide all of the symbols in the formulae with an assigned meaning. For the non-monotonic logic CI^* a Kripke structure (KS) with a preference criterion on the set of possible worlds $\mathcal{W}$ is provided. KS is a pair (TI, $\mathcal{W}$), where TI provides the interpretation for temporal terms and $\mathcal{W}$ provides the set of possible interpretations for the non-temporal terms. Giving only a single interpretation for temporal terms follows the observation that different events or facts do not change the course of time itself.

Formally, TI is a set $(\mathcal{Z}, \leq, TFN, M_t)$, where

$\mathcal{Z}$ is the set of integers,
$\leq$ is the familiar order on $\mathcal{Z}$,
TFN is a set of total functions in $\bigcup_k(\mathcal{Z}^k \rightarrow \mathcal{Z})$, and
$M_t = (M_{T_1}, M_{T_2})$ is the temporal meaning function where
- M_{T_1} : TC $\rightarrow \mathcal{Z}$ assigns temporal constant symbols a value in the universe of timepoints and
- M_{T_2} : TF $\rightarrow TFN$ assigns every n-ary temporal function symbol an n-ary function.

$\mathcal{W}$ is a set of interpretations W, where $W = (\Omega, FN, RL, M_I)$. Explicitly,

Ω is a universe of entities disjoint from $\mathcal{Z}$,
FN is a set of total functions in $\bigcup_k(\Omega^k \rightarrow \Omega)$,
RL is a set of relations over Ω, and
M_I is a triple $(M_{I_1}, M_{I_2}, M_{I_3})$ where
- M_{I_1} : $\mathcal{Z} \times \mathcal{Z} \times$ C $\rightarrow \Omega$ assigns values to constants,
- M_{I_2} : $\mathcal{Z} \times \mathcal{Z} \times$ F $\rightarrow FN$ associates function symbols with functions, and
- M_{I_3} : $\mathcal{Z} \times \mathcal{Z} \times$ R $\rightarrow RL$ associates relation symbols with relations.

Even though M_I is the meaning function for non-temporal terms, the above induces a temporal dependency. For example, in a fifteenth century time context the constant *usa* certainly requires a different assignment to an individual in Ω than in a twentieth century time context. The same goes for functions, such as *president* (*usa*), where the result of the mapping depends on the time frame.

The functions in the triple M_I are symmetric in the first two arguments, i.e. an interval in time is specified by its beginning and endpoint, no matter in which order they are listed. The order is induced by $\leq$ on $\mathcal{Z}$. According to the initial conventions about how to model time for the purposes here $\mathcal{Z}$ was chosen as the universe of timepoints. For a more general semantics, TW as a designator for the universe of timepoints can easily be substituted in the above. Also, we may want to think of the integers in $\mathcal{Z}$ as being somehow branded as different (for example by indexing them with T), so that Ω and $\mathcal{Z}$ can be disjoint, but we still have a way of using integers in our universe of discourse.

A *variable assignment* with respect to TI and a $W \in \mathcal{W}$ is a pair of functions $VA = (VA_T, VA_I)$, where

VA_T: $\mathtt{TV} \rightarrow TW$ assigns values to temporal variables and

VA_I: $\mathtt{V} \rightarrow \Omega$ assigns values to non-temporal variables.

For notational convenience in the listing of conditions for satisfiability we introduce a meaning function M that unifies the effect of M_T, a specific VA and a specific M_A. To be explicit,

- if $\mathtt{tv} \in \mathtt{TV}$ then $M(\mathtt{tv}) = VA_T(\mathtt{tv})$,
- if $\mathtt{tc} \in \mathtt{TC}$ then $M(\mathtt{tc}) = M_{T_1}(\mathtt{tc})$,
- if $\mathtt{tf} \in \mathtt{TF}$ and $ttrm = \mathtt{tf}(ttrm_1, ..., ttrm_n)$ is a temporal term, then $M(ttrm) = M_{T_2}(\mathtt{tf}(M(ttrm_1), ..., M(ttrm_n))$,
- if $v \in \mathtt{V}$, then for all $t_1, t_2 \in \mathcal{Z}$, $M(t_1, t_2, v) = VA_I(v)$,
- if $c \in \mathtt{C}$, then $M(t_1, t_2, c) = M_{I_1}(t_1, t_2, c)$, and
- if $f \in \mathtt{F}$ and $trm = f(trm_1, ..., trm_n)$ is a non-temporal term, then $M(trm) = M_{I_2}(t_1, t_2, f)(M(trm_1), ..., M(trm_n))$.

We will write $KS, W \models \varphi\ [VA]$ to denote that the interpretation structure KS and interpretation $W \in \mathcal{W}$ satisfy wff φ under variable assignment VA. The following are the conditions for satisfiability:

$KS, W \models ttrm_1 = ttrm_2[VA]$ iff $M(ttrm_1) = M(ttrm_2)$.

$KS, W \models ttrm_1 \leq ttrm_2[VA]$ iff $M(ttrm_1) \leq M(ttrm_2)$.

$KS, W \models \mathbf{TRUE}\ (ttrm_a, ttrm_b, \mathtt{r}\ (trm_1, ..., trm_n))\ [VA]$ iff $(M(M(ttrm_a), M(ttrm_b), trm_1), ..., M(M(ttrm_a), M(ttrm_b), trm_n)) \in M_{I_2}(M(trm_a), M(trm_b), \mathtt{r})$.

$KS, W \models \varphi_1 \wedge \varphi_2[VA]$ iff $KS, W \models \varphi_1[VA]$ and $KS, W \models \varphi_2[VA]$.

$KS, W \models \neg\varphi$ iff $KS, W \not\models \varphi[VA]$.

$KS, W \models \forall z\ \varphi$ iff $KS, W \models \varphi[VA']$ for all VA' that agree with VA everywhere except possibly on z.

$KS, W \models \Box\varphi[VA]$ iff $KS, W' \models \varphi$ for all $W' \in \mathcal{W}$.

Up to this point the description yields a rather ordinary monotonic modal logic. Only the special role of time distinguishes this logic so far and makes it suitable to represent temporal knowledge. The following definitions can remain as they are for classical logics: a Kripke structure KS and an interpretation in W are a model for a formula φ if KS and W satisfy φ for any variable assingment. A formula φ is valid if its negation has no model.

The motivation for requiring non-monotonicity in a logic for the prediction task was given earlier. This goal is achieved by installing a preference criterion on the Kripke structures. Some preliminary definitions are required:

Definition 2. *Base wffs are those without an occurrence of the modal operator.*

Definition 3. *The latest time point (l.t.p.) of a base formula is the latest time point with respect to the order on $\mathcal{Z}$ in it. Explicitly:*

1. *The l.t.p. of* **TRUE** $(t_1, t_2, \mathtt{r}\ (trm_1, ..., trm_n))$ *is the maximum of t_1 and t_2.*
2. *The l.t.p. of $\varphi_1 \wedge \varphi_2$ is the latest of the l.t.p.'s of φ_1 and φ_2.*
3. *The l.t.p. of $\neg\varphi$ is the l.t.p. of φ.*
4. *The l.t.p. of $\forall v\ \varphi$ is the earliest among the l.t.p.'s of of all φ' that result from substituting a constant in place of all occurrences of v, or $-\infty$, if no such earliest time point exists.*

The latter is non-intuitive, but needed since later formulae with an earliest l.t.p. will be of interest.

Definition 4. *A Kripke structure KS_2 is chronologically more ignorant than a Kripke structure KS_1 if there exists a time t_0 such that*

1. *for any base sentence φ with an l.t.p. $\leq t_0$, if $KS_2 \models \Box\varphi$, then also $KS_1 \models \Box\varphi$, and*
2. *there exists some base sentence φ whose l.t.p is t_0 such that $KS_1 \models \Box\varphi$, but $KS_2 \not\models \Box\varphi$.*

This will be denoted by $KS_1 \sqsubset_{ci} KS_2$.

Definition 5. *KS is said to be a chronologically most ignorant (c.m.i.) model of φ if $KS \models_{\sqsubset_{ci}} \varphi$, that is, if $KS \models \varphi$ and there exists no KS' such that $KS' \models \varphi$ and $KS \sqsubset_{ci} IS'$.*

*CI** is the logic obtained from the syntax and semantics given above with $\sqsubset_{ci}$ as a preference criterion on the Kripke structures.

CAUSAL THEORIES

Within *CI** sets of sentences can be identified that have a unique chronologically most ignorant model.

Definition 6. *Base sentences in CI*are those without an occurrence of the modal operator. They refer to the modelled world, rather than to knowledge of it.*

Definition 7. *Atomic base sentences in CI*are of the form* **TRUE** $(t_1, t_2, \mathtt{r}\ (trm_1, ..., trm_n))$ *or* **TRUE** $(t_1, t_2, \neg\mathtt{r}\ (trm_1, ..., trm_n))$, *where r is an n-ary relation symbol and the terms trm_i are ground terms. Though the predicate structure is retained, the statement is essentially of the same nature as a proposition in propositional calculus.*

Definition 8. *A theory in CI*is a collection of sentences in CI*.*

Given these preliminary definitions, causal theories in *CI**can be identified as the following:

Definition 9. *A causal theory Ξ in CI*is a theory in which all sentences are causal rules of the type*

$$\Phi \wedge \Theta \supset \Box\varphi.$$

The following conditions apply:

1. *φ is an atomic base sentence.*
2. *Φ is a conjunction of sentences $\Box\varphi_i$, and Θ is a conjunction of sentences $\diamond\varphi_i$, where φ_i is an atomic base sentence and the l.t.p. of φ_i precedes t_1, i.e. if the l.t.p. of φ_i is t_i, then $t < t_i$. Either Φ or Θ or both may be empty, in which case the universally true con-*

stant T *is substituted in their place. A sentence in which* Φ *is empty is termed a boundary condition.*

3. *For all boundary conditions* $\Theta \supset \Box(t_1,\ t_2,\ \mathrm{r}(trm_1, ...,\ trm_n))$ *there exists a time* t_0, *such that* t_0 *precedes* t_1 *and* t_2.
4. *For any two causal rules in* Ξ, *if one contains* $\Diamond(t_1,\ t_2,\ \mathrm{r}(trm_1, ...,\ trm_n))$ *on its l.h.s., then the other will not contain* $\Diamond(t_1,\ t_2,\ \neg\mathrm{r}(trm_1, ...,\ trm_n))$ *on its l.h.s., for any* t_1 *and* t_2.
5. *If* $\Phi_1 \wedge \Theta_1 \supset \Box(t_1,\ t_2,\ \mathrm{r}(trm_1, ...,\ trm_n))$ *and* $\Phi_2 \wedge \Theta_2 \supset \Box(t_1,\ t_2,\ \neg\mathrm{r}(trm_1, ...,\ trm_n))$ *are sentences in* Ξ, *then* $\Phi_1 \wedge \Theta_1 \wedge \Phi_2 \wedge \Theta_2$ *is inconsistent.*

The structure of the causal rules provides for the desired "knowing this, not knowing that, we will predict φ" - structure. The time constraint on sentences in Φ and Θ ensures that a prediction for time t does not rely on events at time t or thereafter. Boundary conditions are a way of including knowledge that comes into being without an explicit reason. For a proof by induction on time in the unique c.m.i. model theorem the basis t_0 is needed, and is mandated by condition 3. With respect to the time models in machines this boundedness in the past does not pose a restriction. A sentence of the type $\Diamond\varphi$ is practically a default truth assignment to φ. It does not make sense to have both φ and $\neg\varphi$ true by default, which is assured by condition 4. Condition 5 introduces another intuitive restriction by expressing that consistent causes cannot bring about an inconsistent result.

Definition 10. *A time-bounded Kripke structure* M/t *is a structure which can be viewed as an incomplete Kripke structure. It assigns a truth value only to those base sentences whose l.t.p.*$\leq t$. *Truth values of ordinary sentences with respect to* M/t *are determined using the usual compositional rules.* M/t *partially satisfies a theory* Ξ *if* M/t *satisfies all members of* Ξ *whose l.t.p.*$\leq t$.

We can now prove that a causal theory not only has a c.m.i. model, but that all its c.m.i. models are in fact all the same since in all of them the same base sentences are known.

Theorem 1. *(Unique c.m.i. model of causal theories in CI*) Let* Ξ *be a causal theory in CI*, then*

1. Ξ *has a c.m.i. model, and*
2. *if* M_1 *and* M_2 *are both c.m.i. models of* Ξ, *and* φ *any base sentence, then* $M_1 \models \Box\varphi$ *iff* $M_2 \models \Box\varphi$.

Proof: The proof is constructive. A model M for the causal theory Ξ is generated and shown to be chronologically more ignorant than any model for Ξ that differs from M in the truth value of some base sentence φ.

We begin with a time bounded interpretation M/t_0. Time t_0 is picked as outlined in condition 3 of definition 9. If φ is a base tautology with l.t.p. $\leq t_0$, then $M/t_0 \models \Box\varphi$. These are the only base wffs with l.t.p. $\leq t_0$ not falsified by M/t_0.

M/t is modified into $M/t+1$ in the following manner: let $CONSEQUENTS_{t+1} = \{\Box(t',\ t+1,\ \varphi) : \Phi \wedge \Theta \supset \Box(t',\ t+1,\ \varphi) \in \Xi \text{ and } M/t \models \Phi \wedge \Theta\}$. $M/t+1$ is obtained by adopting all of M/t and making all wffs in $CONSEQUENTS_{t+1}$ and their tautological consequences true. All other base wffs with l.t.p. $t+1$ are falsified. By condition 5 of definition 4, $CONSEQUENTS_{t+1}$ does not contain both a $\Box\varphi$ and a $\Box\neg\varphi$. The induction step thus induces no inconsistency. With the consistency of M/t_0 this means that no inconsistency can be induces at all.

M as constructed above is obviously a model for Ξ. Its uniqueness as a c.m.i. model remains to be shown. Assume there is an interpretation M', that differs from M in the truth value of a non-tautological base sentence φ. If φ's l.t.p. is $\leq t_0$, then M' is chronologically less ignorant than M. Otherwise, let t be the earliest point in time, where M and M' differ in the truth value of a base sentence (φ's l.t.p. is t). By construction of M, such a t exists and $t_0 < t$. Then two cases are to be considered.

$M \models \Box\varphi$, $M' \not\models \Box\varphi$: Ξ contains a causal rule $\Phi \wedge \Theta \supset \Box\varphi$. The l.t.p. of $\Phi \wedge \Theta$ is $< t$. Since t is the earliest time for M and M' to disagree, $M \models \Phi \wedge \Theta$ and $M' \models \Phi \wedge \Theta$. But with $M' \not\models \Box\varphi$, M' cannot be a model for Ξ.

$M \not\models \Box\varphi$, $M' \models \Box\varphi$: as seen in the former case, for a base sentence φ with l.t.p. $> t_0$, if $M \models \Box\varphi$, then also $M' \models \Box\varphi$. But since $M \not\models \Box\varphi$ here, we can conclude that M' is chronologically less ignorant than M. □

CONCLUSION

Given the pragmatic assumptions made about the structure of time, and the constructive nature of the proof, a mechanization of the generation of a c.m.i. model seems plausible, given the underlying causal theory is finite. Unfortunately the transformation of general inference rules can easily generate infinitely many causal rules. Cutoff times limiting the range of the predictions may help. And, as Shoham points out in [SHOHAM 88], the generation of the model can be accelerated by only considering those time points that are potentially l.t.p. of known atomic base sentences.

Bibliography

Allen, J. F. 1984. "Towards a General Theory of Action and Time." *Artificial Intelligence*, 23(2) (July) 123–154.

Galton, A. 1987. "Temporal Logic and Computer Science: An Overview." in *Temporal Logics And Their Applications*, A. Galton, ed., Academic Press, San Diego, CA, 1–52.

Hughes, G. E., and M. J. Cresswell. 1969. *Introduction to Modal Logic*. Methuen, London, Great Britain.

Kripke, S. 1963. "Semantical Considerations on Modal Logic." *Acta Philosophica Fennica*, 16: 83–94.

McDermott, D. V. 1982. "A Temporal Logic for Reasoning About Processes and Plans". *Cognitive Science*. 6 (June) 101–155.

Shoham, Y. 1988. *Reasoning About Change*. MIT Press, Cambridge, Mass., and London, Great Britain.

CONTRADICTORY BELIEF AND LOGIC
Doxastic Logic for the Real World

Roderic A. Girle
Automated Reasoning Project
Australian National University
Canberra, A.C.T. 2601
AUSTRALIA.

ABSTRACT

The doxastic interpretation of modal logic is the interpretation of modal logic for *belief*. Doxastic systems based on classically consistent logics run contrary to the reality of inconsistent and unstructured belief systems. There are more doxastically plausible logics, logics which are inconsistency tolerant: the *paraconsistent, non-normal, non-cartesian* doxastic logics. The aims of this paper are to discuss such logics, and to investigate basing automated theorem provers for such logics on semantic tableau.

INTRODUCTION

The doxastic interpretation of modal logic is the interpretation of modal logic for *belief*. Many such interpretations were vigorously investigated by philosophers and logicians during the 1960s and into the 1970s, but thereafter interest in this field waned somewhat for two main reasons. In the first place, the logically consistent and highly structured nature of most doxastic systems runs contrary to the psychological realities of real world belief systems which are often inconsistent and unstructured. Secondly, the researchers, like researchers in so many other areas of logic, lacked a suitable applications medium in which to test their ideas, test the efficacy or otherwise of the plethora of logical systems they spawned, and in which to bring about some practical resolution of the conflicting claims concerning these systems.

If these problems can be dealt with, then doxastic logic gives us a logic which is far better than the principle alternative, probabilistic logic, which has been suggested for coping with belief. It is better because it is simpler, and does not fall into some of the paradoxical and counter-intuitive traps of probability logics. For example, agnostic lack of belief does not have to classified as 50-50 uncertainty. General hypotheses do not have to be given a probability of zero, which is what they would have to be given if we were to take their generality seriously, and probabilities do not have to be juggled into relative order. Doxastic logic gives us a *qualitative* rather than a *quantitative* model for dealing with belief.

It has become obvious to many researchers that one of the main problem areas for doxastic logic is logical consistency. Most doxastic logics are interpretations of *classical* modal logics. Classical modal logics are built upon classical propositional and predicate logic. Classical logic is utterly intolerant of inconsistency, and is reduced to uncontrollable inferential idiocy in the presence of contradiction. It is vital, therefore, that inconsistency tolerant logics be investigated as a base on which doxastic logic can be built. Such non-classical logics are to be found in the family of *paraconsistent* logics. These logics have been thoroughly investigated, and now is the time for us to look at their application as a base for doxastic logic.

There are also a series of problems at the modal level of doxastic logic which run in parallel with problems in *epistemic* logic, the logic of *knowledge*. In that area of logic, many researchers agree that *normal cartesian* modal logics (**T**, **S4**, and **S5**) are too strong, and that plausible models for knowledge are more likely to be found in the weaker *non-normal non-cartesian* modal logics such as **S2** and **E2**. (Girle 1970, Scotch and Jennings 1981) Similarly, the doxastically interpreted strong modal systems (**DT** and **D4**) are implausible, while the weaker *non-normal non-cartesian* doxastic logics, such as **D2** and **C2**, are more likely to provide plausible models for belief. Given the potential importance of the doxastic interpretation of paraconsistent modal logics, the time is now right to embark on a computational exploration of this part of what can well be described as a doxastic labyrinth. (Girle and McRobbie 1988)

The development of a range of powerful Artificial Intelligence techniques in the areas of automated theorem proving and expert systems has provided us with a set of applications in which doxastic logics can be applied and tested. Judgements could then be made as to the ease with which they might be applied, and as to the degree of their usefulness.

The aim of this paper is threefold. First to draw attention to the paraconsistent logics and to point out how their inconsistency tolerance provides us with a better base than

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

classical logic on which to build doxastic logics. With their use there is no need to invent elaborate, even baroque devices, to avoid the problems which can arise when bodies of data, infected with contradiction, are acted upon by classical inference engines. This leads us on immediately to our second object.

The second aim of this paper is to draw attention to the importance of the non-normal logics from the point of view of their doxastic interpretation and to some of the results of the considerable amount of research into them. In doing so we suggest that the relative neglect that these logics have suffered is due in part to what is seen as their formidable complexity, particularly their proof theory. However we believe that this modal complexity is more apparent than real and can to a large degree be overcome using alternative formulations in combination with the tools and technologies of computer science and AI, specifically those of automated theorem proving.

Thus the third aim of this paper is to investigate automated theorem proving for non-normal paraconsistent logics. We will concern ourselves with a semantic base on which to build theorem provers in this area. In particular, Hintikka style model-set/model-system semantics are developed for these logics. It will be shown that such semantics lend themselves to prover implementation in a fairly obvious and transparent way.

PARACONSISTENT LOGICS

The problem which classical logic has with inconsistency can be exemplified by the classical theorems of the form:

$$(A \ \& \ \sim A) \rightarrow D$$

known as *ex falso quodlibet* (Whatever you wish follows from contradiction). So, when a contradiction is deduced from a set of beliefs, it also follows by *ex falso*, that any proposition, including every possible contradiction, follows validly from that set of beliefs. There is no restriction of the consequences of inconsistency to related or relevant beliefs. So, if we apply a classical inference engine to drawing conclusions from an inconsistent set of beliefs, a malignant inconsistency spreads across absolutely everything.

But this does not happen in real life belief systems. Reasonable people isolate inconsistency, suspend judgement, or follow other inconsistency handling procedures. Classical inference engines cannot deal so easily with the inconsistency of real world belief systems. So, when doxastic logics are built on a classical base they are equally unable to cope sensibly with inconsistency. One standard classical response has been to develop restricted doxastic logics by interpreting doxastic logics as logics for *rational belief*. But this interpretation begs a wide range of questions concerning rational and reasonable belief in favour of classical logical consistency. Not only is there no conceptual impossibility about declaring that someone has inconsistent beliefs, but it is by no means clear that everyone with logically inconsistent belief systems is irrational. There are at least three important concerns.

First, it is not clear that a necessary condition for a rational belief system is *classical* logical consistency. Second, it is not even clear that the attributions of rationality should depend on the logical structure of a system of beliefs. It can be argued that rationality should depend on the way in which an *agent* deals with contradictions when they are detected in a system of beliefs.

Thirdly, there is a need to develop some taxonomy for inconsistency in belief systems. We might consider a taxonomy related to the *depth* of its occurrence. Some inconsistencies occur on the *surface* of belief systems. The most obvious are the *prima face* contradictions of the form:

$$(p \ \& \ \sim p)$$

Because of the classical threat of global inconsistency there is a fairly standard, but not uncontroversial, expectation that a rational agent will resolve such contradictions. But in some contexts it might be better to leave such inconsistencies unresolved for the time being. In this case we need a logic that will at least quarantine the inconsistency. Paraconsistent logics will do just this, since they lack *ex falso quodlibet* and its related theorems. Such logics will help us to deal with contradiction in a more rational fashion than indicated by classical logic. On the other hand, not all inconsistency is on the surface. Some inconsistencies are very deeply buried in belief systems. They often become apparent only after the malignancy has been widely spread, even to totally unrelated areas. If a logic lacks *ex falso quodlibet* it will give us an inference system which is inoculated against the spread of more deeply buried contradiction.

In this paper we focus on the logics around **RM**. There are Kripke style semantics for these systems. Important work began with Dunn's construction of binary semantics for **RM** (Dunn 1976). This work has recently been extended by Surendonk, Slaney and Girle (1989).

DOXASTIC INTERPRETATIONS OF MODAL LOGICS

The general principle under which doxastic interpretations are given to modal logics is that the $\Box$ operator is indexed to some believer, a, and is transformed to B_a. A formula $B_a p$ is read as "*a believes that p*". A general doxastic logic for more than one believer is simply a multiply-modal logic (Rennie 1970), consisting of a logic for each believer (index), with, if so desired, some interconnecting theorems (axioms).

The logics we are to consider lack at least three of the features of the doxastically interpreted strong *normal* modal logics. First, all normal modal logics contain the highly contentious *Rule of Necessitation* - from the theoremhood

of A infer the theoremhood of $\Box A$. When this is given a doxastic interpretation it means that if A is a theorem, then everybody believes that A. This principle, the epistemic analogue of which is called the principle of logical omniscience, is clearly contentious, if not absurd, under doxastic interpretation. Some weakening of this principle is clearly called for if we are to have a more plausible doxastic logic.

The non-normal modal logics do not contain the full-blooded Rule of Necessitation. They usually contain rules which can be described as 'weaker' than Necessitation. Some such rules allow that if A is a theorem of propositional logic then $B_a A$ is a theorem. Other such rules allow inferences with modal content. An example of such a rule is *BKR*, which under epistemic interpretation would be - from the theoremhood of $B_a(A \rightarrow D)$ to infer the theoremhood of $B_a(B_a A \rightarrow B_a D)$ These weaker forms are slightly more acceptable under doxastic interpretation than the full-blooded Necessitation Rule.

Second, some of the stronger normal modal logics contain the controversial theorem $\Box A \rightarrow \Box\Box A$. Under an epistemic interpretation the last is known by logicians as the KK-thesis and by researchers in computer science and AI as the positive introspection principle. The analogue of the KK-thesis is implausible under either straightforward or modified interpretation. It can be set aside for the moment because it is a thesis of only the stronger normal doxastic modal logic **D4**. It is not a theorem of the weaker of the two normal doxastic logics we have mentioned, **DT**.

Third, the stronger modal logics have theorems of the form $B_a A$. Even when the full Necessitation Rule is not present, there does seem to be something contentious about any doxastic logic in which there are any theorems of the form $B_a A$. The categorical assertion that everyone, in some sense, must believe, indeed must know certain things, was given clearest expression in the work of Rene Descartes. Descartes' views have not gone unchallenged in the history of epistemology.

We can mark out the expression of the Cartesian view in doxastic logic by dividing the modal logics, particularly when they are considered under either doxastic or epistemic interpretation, into two classes. These are the *Cartesian* and the *non-Cartesian* (Girle 1970). The non-Cartesian logics are those in which there are no theorems of the form $B_a A$, or of the form $K_a A$ under epistemic interpretation. The semantics of the non-Cartesian logics are more complex than those of the Cartesian systems. The Cartesian non-normal logics are typified by the systems **S0.5**, **S1**, **S2** and **S3**. The non-Cartesian non-normal logics are typified by the systems **C0.5**, **C1**, **C2** and **C3**, and the related **D** and **E** systems.

A DOXASTIC SYSTEM

In this paper we make use of the non-classical propositional logic **CMS2** and the modal logic **C2**. **CMS2** is a system which develops from **RM**. The binary semantics for **RM** make use of three truth–values: $\{t, f\}$, $\{t\}$, and $\{f\}$. For **CMS2**, we add the additional value $\{\}$, and otherwise leave the semantics the same. **C2** is a non-Cartesian non-normal logic which is included in (is a sub-system of) both **D2** and **S2**. It has the simplest proof system and semantics in that group of logics, and is the simplest of the non-normal logics as a whole. Axiomatic versions of these systems may be retrieved from Lemon (1966). Neither **D2** nor **C2** have any theorems of the form $B_a A$.

We now set out the systems in terms of Hintikka's model–set model–system semantics (Hintikka 1962), but with $\Box$ replaced by B_x, B_y, ... , where x, y, ... are for any indices for believers, a, b, ...

We also define $C_x\alpha =_{df} \sim B_x \sim \alpha$

We use a propositional language defined in the usual way, with the connectives $\sim$ $\&$ $\vee$ and $\rightarrow$, and as many propositional letters as needed. The set of formulas is denoted by WFR (Well-formed Formulas of **RM** and nearby logics). These formulas are extended by the addition of four **flags**: *t, nt, f, nf*. The intuitive semantic content of these is *at least true, not true, at least false* and *not false* respectively. In the semantics for **CSM2**, a formula may be *both true and false, just true, just false* or *null value*. We define the set FWFR (flagged WFR) by:

If A is a WFR and * is a flag, then $*A$ is an FWFR.

A **CSM2**-model-system is a set of *model-sets* of FWFR, the model-sets being the field of two dyadic relations, R and S. If iRj, i is said *to have relevant access to j*. If iSj, then i is said *to have doxasitic access to j*. The membership of the model-sets in an **CSM2**-model-system is stipulated by the conditions below (where μ_i, μ_j, μ_n ... are for any model-sets in the model-system, and the letters *A, D, E,* ... are formula schema, and *!* stands for any *flag*). We begin with the conditions for the *paraconsistent* system:

(C.t0) $\{t\,A, nt\,A\}$ *is not a sub-set of any model-set.*
(C.f0) $\{f\,A, nf\,A\}$ *is not a sub-set of any model-set.*
(C.t~) If $\{t\ {\sim}A\} \subseteq \mu_i$, *then* $\{f\,A\} \subseteq \mu_i$.
(C.nt~) If $\{nt\ {\sim}A\} \subseteq \mu_i$, *then* $\{nf\,A\} \subseteq \mu_i$.
(C.f~) If $\{f\ {\sim}A\} \subseteq \mu_i$, *then* $\{t\,A\} \subseteq \mu_i$.
(C.nf~) If $\{nf\ {\sim}A\} \subseteq \mu_i$, *then* $\{nt\,A\} \subseteq \mu_i$.
(C.t&) If $\{t\,(A \,\&\, D)\} \subseteq \mu_i$, *then* $\{t\,A, t\,D\} \subseteq \mu_i$.
(C.nt&) If $\{nt\,(A \,\&\, D)\} \subseteq \mu_i$, *then either* $\{nt\,A\} \subseteq \mu_i$

(C.f&) If $\{f\,(A \,\&\, D)\} \subseteq \mu_i$, *then either* $\{f\,A\} \subseteq \mu_i$ *or* $\{f\,D\} \subseteq \mu_i$.
(C.nf&) If $\{nf\,(A \,\&\, D)\} \subseteq \mu_i$, *then* $\{nf\,A, nf\,D\} \subseteq \mu_i$.
(C.t∨) If $\{t\,(A \vee D)\} \subseteq \mu_i$, *then either* $\{t\,A\} \subseteq \mu_i$ *or* $\{t\,D\} \subseteq \mu_i$.
(C.nt∨) If $\{nt\,(A \vee D)\} \subseteq \mu_i$, *then* $\{nt\,A, nt\,D\} \subseteq \mu_i$.
(C.f∨) If $\{f\,(A \vee D)\} \subseteq \mu_i$, *then* $\{f\,A, f\,D\} \subseteq \mu_i$.
(C.nf∨) If $\{nf\,(A \vee D)\} \subseteq \mu_i$, *then either* $\{nf\,A\} \subseteq \mu_i$ *or* $\{nf\,D\} \subseteq \mu_i$.
(C.t→) If $\{t\,(A \rightarrow D)\} \subseteq \mu_i$, *then either* $\{nt\,A, f\,A\} \subseteq \mu_i$ *or* $\{nt\,A, nf\,D\} \subseteq \mu_i$ *or* $\{t\,D, f\,A\} \subseteq \mu_i$ *or* $\{t\,D, nf\,D\} \subseteq \mu_i$.
(C.t→R) If $\{t\,(A \rightarrow D)\} \subseteq \mu_i$ *and* $\mu_i R \mu_j$, *then either* $\{nt\,A, f\,A\} \subseteq \mu_j$ *or* $\{nt\,A, nf\,D\} \subseteq \mu_j$ *or* $\{t\,D, f\,A\} \subseteq \mu_j$ *or* $\{t\,D, nf\,D\} \subseteq \mu_j$.
(C.nt→) If $\{nt\,(A \rightarrow D)\} \subseteq \mu_i$, *then there is some model-set in the same model-system, say* μ_j, *such that* $\mu_i R \mu_j$ *and either* $\{t\,A, nt\,D\} \subseteq \mu_j$ *or* $\{nf\,A, f\,D\} \subseteq \mu_j$.
(C.f→) If $\{f\,(A \rightarrow D)\} \subseteq \mu_i$, *then* $\{t\,A, f\,D\} \subseteq \mu_i$.
(C.nf→) If $\{nf\,(A \rightarrow D)\} \subseteq \mu_i$, *then either* $\{nt\,A\} \subseteq \mu_i$ *or* $\{nf\,D\} \subseteq \mu_i$.
(C.Heriditary t) If $\{t\,A\} \subseteq \mu_i$ *and* $\mu_i R \mu_j$, *then* $\{t\,A\} \subseteq \mu_j$.
(C.Heriditary f) If $\{f\,A\} \subseteq \mu_i$ *and* $\mu_i R \mu_j$, *then* $\{f\,A\} \subseteq \mu_j$.

It is worth noting that the semantics for **RM** are gained by adding:

(C.Non-Null) $\{nt\,A, nf\,A\}$ *is not a sub-set of any model-set.*

The doxastic conditions are:

(C.~B) If $\{!\ {\sim}B_x A\} \subseteq \mu_i$, *then* $\{!\ C_x {\sim}A\} \subseteq \mu_i$.
(C.~C) If $\{!\ {\sim}C_x A\} \subseteq \mu_i$, *then* $\{!\ B_x {\sim}A\} \subseteq \mu_i$.
(CfB) If $\{f\,B_x A\} \subseteq \mu_i$, *then* $\{t\,C_x {\sim}A\} \subseteq \mu_i$.
(CfC) If $\{f\,C_x A\} \subseteq \mu_i$, *then* $\{t\,B_x {\sim}A\} \subseteq \mu_i$.
(C.ntB) If $\{nt\,B_x A\} \subseteq \mu_i$, *then* $\{nf\,C_x {\sim}A\} \subseteq \mu_i$.
(C.ntC) If $\{nt\,C_x A\} \subseteq \mu_i$, *then* $\{nf\,B_x {\sim}A\} \subseteq \mu_i$.
(C.nfB) If $\{nf\,B_x A\} \subseteq \mu_i$, *then* $\{nt\,C_x {\sim}A\} \subseteq \mu_i$.
(C.nfC) If $\{nf\,C_x A\} \subseteq \mu_i$, *then* $\{nt\,B_x {\sim}A\} \subseteq \mu_i$.
(C.C) If $\{t\,C_x A\} \subseteq \mu_i$, *then there is at least one model-set, say* μ_j, *in the model-system, such that* $\{t\,A\} \subseteq \mu_j$ *and* $\mu_i S \mu_j$.
(C.B+) If $\{t\,B_x A\} \subseteq \mu_i$ *and* $\mu_i S \mu_j$, *then* $\{t\,A\} \subseteq \mu_j$.
(C.CN) If $\{t\,C_x A\} \subseteq \mu_i$ *and* μ_i *is not accessed from any model-set, then there is at least one model-set, say* μ_j, *in the model-system, such that* $\{t\,A\} \subseteq \mu_j$ *and* $\mu_i S \mu_j$.
(C.CB) If $\{t\,C_x A\} \subseteq \mu_i$ *and, for any D,* $\{t\,B_x D\} \subseteq \mu_i$, *then there is at least one model-set, say* μ_j, *in the model-system, such that* $\{t\,A\} \subseteq \mu_j$ *and* $\mu_i S \mu_j$.

The definition of Validity is:

CMS2-*valid(A) iff there is no* μ_i *in any* **CMS2**-*model-system such that* $\{nt\,A\} \subseteq \mu_i$.

A validity test takes the form of a *reductio ad absurdum* proof. Such a proof can be set out in a refutation tree. It begins with the assertion that a not-true formula is in a model-set. Use is then made of the tree schema for **CMS2**-truth-trees. FWFR's are entered into the trees in the usual way, so that, for example, $f\,B\ (j)$ means that $\{f\,B\} \subseteq \mu_j$.

THE SEMANTICS IN LOGIC PROGRAMMING

The model-set/model-system semantics can be viewed as a thinly disguised prescription for list manipulation or, to put it in a more declarative vein, to declare certain things about lists. A model-set can be seen as a list of formulas, and a model-system seen as a list of lists. Formulas are decomposed in terms of the semantics. New lists can be generated. A closed list is a list which either contains nothing or contains formulas so as to breach the rules. The question of the S-Validity (where **S** is for some system) of a formula becomes the question, can the *not–true* formula be placed in a list of lists and the list changed by certain rules into a list of closed lists? If so, it is S-Valid, if not, not. Code for this kind of logic programming can be found in Fitting (1988).

Finally, some brief words about combinatorial explosion. There are two major strategies. The first is to take the prover which is a "brute force" utilisation of the model-set semantics, and to make it efficient by reducing the production of new lists. There are standard "short cuts" for trees which can be used. This is just a matter of fine tuning.

The second is to apply a technique which will make it clear that lists will fail to close without having to decompose the formulas. One such technique uses the fact that even when logics do not have a single finite characteristic model, they do have finite models. Such models validate all the theorems of a given logic, though some non-theorems as well. They invalidate the rest.

The general principle in this technique consists in using finite models of a given logic to restrict the search to a sub-space of the search space. This technique is knowledge-based since the search for a proof is fundamentally directed by the models which may be thought of as encoding sophisticated logical information about the structure of the logic. It is of course part of the wisdom of AI that if searches in symbolic computation are not to fall prey to combinatorial explosion, they must incorporate domain-

specific knowledge in such a way so as to give direction to the search. This technique plays a crucial role in the automated theorem proving system KRIPKE described in Thistlewaite, McRobbie and Meyer (1987). Such finite models can be found computationally. Fundamental research in this area is surveyed in Slaney (1980) and the applications of some of it is described in Slaney (1985).

These general techniques can be applied to modal truth-trees. Every node in a truth-tree can be seen as a set of sub-sets (or list of reduced lists) of the model-sets generated to that point in the construction of the tree. The sub-sets consist of the unresolved formulas. Since the tree is a refutation tree, and despite one's desire to see it close, the real aim is to arrive at an open path, that is, a path in which there is no closure. Each reduced list can be tested against a matrix to see whether appropriate conditions are met. If not, then there is not going to be closure of that path of the tree. The formula being tested will be rejected.

IMPLEMENTATION

Provers which have been produced by the methods set out above have been implemented and run in a parallel version on a Sequent parallel processor using Aurora Or-Parallel Prolog. (see Lusk *et. al.* 1988) It is clear that parallel processing is *prima face* appropriate for dealing with the model-set semantics. But there is one further point to be made.

The generation of a prover for **RM** made it clear that it would be possible to investigate SM and **CSM2** without a great deal of work "by hand". New results for SM, the system which uses the three values *just true*, *just false*, and *null*,
have flowed because of the ease with which investigations have been facilitated. Other closely related systems are being investigated. The investigations have been prompted by the **RM** prover. The provers for related systems can be generated with such ease that further research in the area is being greatly facilitated.

ACKNOWLEDGEMENTS

I wish to acknowledge the work of and correspondence with Melvin Fitting, whose elegant PROLOG theorem provers for **S2** and **S3** started me off in this work.

REFERENCES

Dunn, J.M. 1976. "A Kripke-Style Semantics for R-Mingle Using a Binary Accessibility Relation", *Studia Logica*, 35, 163-172.

Fitting, M. 1988. "First–Order Modal Tableau", *Journal of Automated Reasoning*, 4(2), 191-213

Girle, R.A. 1970. *Explanatory Models for Knowledge and Belief*, Unpublished M.A.Thesis, University of Queensland.

Girle, R.A. 1974. "Epistemic Logic, Language, and Concepts", *Logique et Analyse*, 64, 359-373

Girle, R.A. 1975. "S1 ≠ S0.9", *Notre Dame Journal of Formal Logic*, 16, 339-344

Girle, R.A. and McRobbie, M.A. 1988. "Exploring the Epistemic Labyrinth", in *Proceedings of the Australian Joint Artificial Intelligence Conference* (Adelaide, SA, Nov. 15-18) 104-124

Hintikka, J.J. 1962. *Knowledge and Belief: An Introduction to the Logic of the two Notions*, Cornell University Press, Ithaca.

Hughes, G.E. and Cresswell, M.J. 1972. *An Introduction to Modal Logics*, Methuen, London, Corrected Repr.

Kripke, S.A. 1965. "Semantical Analysis of Modal Logic II: Non-Normal Modal Propositional Calculi", 206-220 in *The Theory of Models*, ed. J.W. Addison, L. Henkin and A. Tarski, North-Holland, Amsterdam.

Lemmon, E.J. 1966. "Algebraic Semantics for Modal Logics I, II", *Journal of Symbolic Logic*, 31, 46-65, 191- 218

Lusk, E., Warren, D.H.D., Haridi, S., Butler, R., Calderwood, A., Disz, T., Olson, R., Overbeek, R., Stevens, R., Szeredi, S., Brand, P., Carlsson, M., Ciepielewski, A. and Hausman, B. 1988. "The Aurora Or-Parallel Prolog System", *Proceedings of the International Conference on Fifth Generation Computer Systems*, forthcoming.

Rennie, M.K. 1970. "Models for Multiply-Modal Systems", *Zeitschrift fur mathematische Logik and Grundlagen der Mathematik*, 16

Schotch, P.K. and Jennings, R.E. 1981. "Epistemic Logic, Skepticism, and Non-Normal Modal Logic", *Philosophical Studies*, 40, 47-67.

Slaney, J.K. 1980. *Computers and Relevant Logics: A Project in Computing Matrix Model Structures for Propositional Logics*, Ph.D. Thesis, Australian National University.

Slaney, J.K. 1985. "3088 Varieties: A Solution to the Ackermann Constant Problem", *Journal of Symbolic Logic*, 50, 487-501.

Surendonk, T., Slaney, J.K. and Girle, R.A. 1989. "The Discreet Charm of F***", Research Report, Automated Reasoning Project, Australian National University.

Thistlewaite, P.B., McRobbie, M.A. and Meyer, R.K. 1987. *Automated Theorem Proving in Non-Classical Logic*, Research Notes in Theoretical Computer Science, Pitman, London, and Wiley, New York.

A COMPARATIVE ANALYSIS: CLASSICAL, FUZZY, AND QUANTUM LOGIC

Eddie Oshins
Department of Physics
Stanford University
Palo Alto, CA 94305

Kenneth M. Ford
Division of Computer Science
The University of West Florida

Rita V. Rodriguez Frank D. Anger
Computer Science Department
Florida Institute of Technology

ABSTRACT

This paper presents an overview of a simple fuzzy logical system as a non-complemented, distributive, and incomplete lattice. Fuzzy logic is contrasted with both classical and quantum logics in order to highlight the fundamental philosophical differences implied by the different models of the world. Logic and set theory are treated in parallel in all the models, and the tools of both lattice theory and Hilbert space are used to make the presentation precise. The elucidation of these models is significant for machine representation of the external world and also human psychology.

1. THE LATTICES OF CLASSICAL LOGIC

In order to reason about our experience, we use many mechanisms, but perhaps the best known and most studied is that of classical propositional logic and Cantor's set theory. These two methods are essentially the same if we allow the identification of a proposition with the set of all "possible worlds" in which the proposition is true. Specifically, classical logic posits a truth-value function which assigns a value of "True" or "False" to each proposition, where different truth-value functions correspond to different possible worlds (or models) for the propositions of interest. Conjunctions of propositions then correspond to intersections of the corresponding sets, disjunctions to unions, and negations to complements. Each truth-value function corresponds, in turn, to a set-theoretic membership function (or "characteristic function") which assigns 1 to members of the corresponding set and 0 to non-members.

This classical approach to reasoning is therefore a dichotomous one, referred to these days as a "two-valued logic." In addition, the approach yields a "crisp set theory," because it answers a clear "yes" or "no" to the question of membership in a set of any given object. The collection of all subsets of some universal set (or the collection of all propositions about some universe of discourse) has operations of intersection and union, making it a "lattice": a set with two operations $\wedge$ (intersection or conjunction) and $\vee$ (union or disjunction) satisfying (for all x, y, and z in the set):

(i) $x \wedge x = x, \quad x \vee x = x$ (idempotence)
(ii) $x \wedge (y \wedge z) = (x \wedge y) \wedge z$ (associativity)
(iii) $x \wedge y = y \wedge x, \quad x \vee y = y \vee x$ (commutativity).

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

The empty set (the False or Absurd proposition), 0, and the universal set (the True or Trivial proposition), 1, indicate that these elements are lattice identities: for all x,

(iv) $0 \wedge x = 0, \quad 0 \vee x = x, \quad 1 \wedge x = x, \quad 1 \vee x = 1.$

Adding a unique compatible complementation operator (write $x\sim$ for the complement or negation of x), the lattice becomes an "orthocomplemented lattice," or "ortholattice," satisfying for all x and y

(v) $(x\sim)\sim = x$ (involution)
(vi) $x \wedge x\sim = 0, \quad x \vee x\sim = 1$ (complement)
(vii) $(x \wedge y)\sim = x\sim \vee y\sim,$
$(x \vee y)\sim = x\sim \wedge y\sim.$ (DeMorgan)

Finally, since the lattice satisfies the distributive property,

(viii) $(x \wedge y) \vee z = (x \vee z) \wedge (y \vee z),$
$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z),$ (distributivity)

it forms an algebraic structure known as a "Boolean algebra."

In classical logic, implication, "$P \Rightarrow Q$", is equivalant to $Q \vee P\sim$. To assert the "truth" of $P \Rightarrow Q$ is to say that $Q \vee P\sim = 1$. It follows from distributivity that

$$P = P \wedge 1 = P \wedge (Q \vee P\sim) = (P \wedge Q) \vee (P \wedge P\sim)$$
$$= (P \wedge Q) \vee 0 = P \wedge Q.$$

Therefore, $P \Rightarrow Q$ if and only if $P = P \wedge Q$. For sets, this same relation corresponds to "contained in":

$$x \subseteq y \text{ if and only if } x = x \wedge y.$$

Implication or containment imposes an order relation (reflexive, transitive, and anti-symmetric) on the lattice of propositions or subsets, and thus we will simply write "$x \leq y$" for $x = x \wedge y$ in a general lattice. A set with such an order relation is called a "partially ordered set," or simply a "poset." Using this order relation, an additional property of the complement in a Boolean algebra is

(ix) if $x \leq y$ then $y\sim \leq x\sim$ (involutive anti-automorphism).

All Boolean algebras have two additional properties: "modularity" and "orthomodularity" (or "weak modularity"):

(x) if $x \leq z$ then $x \vee (y \wedge z) = (x \vee y) \wedge z$
for all x, y, z (modularity)
(xi) if $x \leq y$ then $y = x \vee (y \wedge x\sim)$ (weak modularity)

Some intuition can be gained for these properties through the "Hasse Diagrams" of Figures 1 and 2 (Oshins, Adelson, & McGoveran 1984): the first shows the Boolean algebra associated with a Venn diagram of three intersecting sets, while Figure 2 is a modular, non-distributive lattice representing subspaces of the plane. Furthermore, it is a simple exercise to show that distributivity implies modularity (but not conversely), and modularity and orthocomplemented imply orthomodular (but not conversely). Another helpful definition in this regard is that of "compatibility": x and y are compatible, written "$x \leftrightarrow y$", if and only if either of

$x = (x \wedge y) \vee (x \wedge y^{\sim})$, or
$y = (x \wedge y) \vee (x^{\sim} \wedge y)$.

Using this definition, orthomodularity can be rephrased as saying

(xii) if $x \leq y$ then $x \leftrightarrow y$ (orthomodularity)

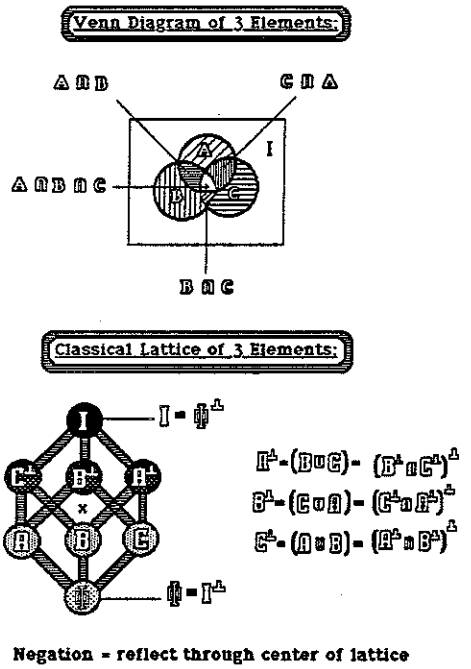


Figure 1: A Lattice of Classical Set Theory

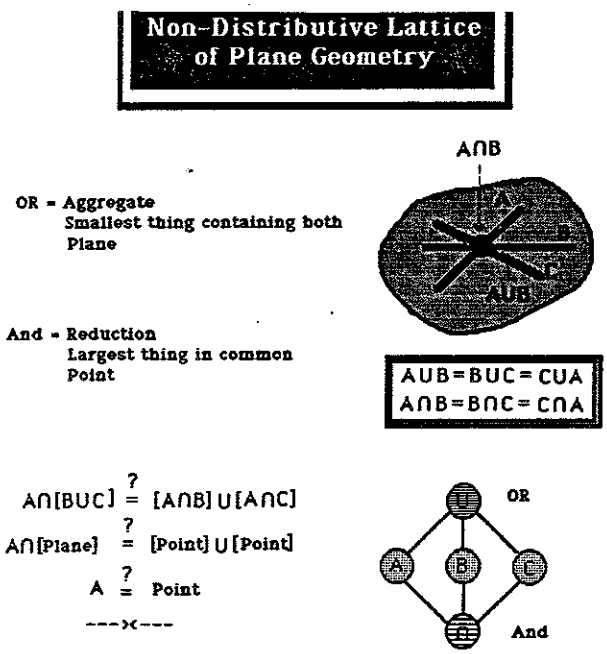


Figure 2: A Lattice of Subspaces of the Plane

2. Fuzzy Logic

An alternative approach to both logic and set theory arises from the fuzzy approach of Zadeh (1965), which starts with a truth-value function whose range is the whole interval from 0 to 1. This means that rather than being true or false, each proposition has a value which is supposed to provide a measure of its "likelihood." Rather than having worlds in which the proposition is true, we have worlds in which the proposition is more likely or less likely. For a given proposition, P, a fuzzy membership function, μ_p, is defined that maps each possible world into the number in the interval from 0 to 1 which is the likelihood of the proposition holding in this world. Thus the membership function is

$$\mu_p : \{\text{possible worlds}\} \text{ ----> } (0, 1).$$

(We consider here "strict" fuzzy logic which never assigns the values 0 or 1; hence the range is the "open" interval (0, 1) excluding the endpoints, which provide a non-fuzzy limit of classical Boolean algebra or crisp set theory.)

In the fuzzy logic approach, the conjunction and disjunction of propositions are interpreted as in classical logic except that the truth values are defined, as expressed by the membership functions, as

$$\mu_{P \wedge Q}(W) = \min\{\mu_p(W),\ \mu_Q(W)\},$$
$$\mu_{P \vee Q}(W) = \max\{\mu_p(W),\ \mu_Q(W)\}.$$

In other words, the likelihood of P and Q is the minimum of the likelihood of either one, while the likelihood of P or Q is the maximum likelihood of the two. It follows that $P \Rightarrow Q$ implies that $\mu_P \leq \mu_Q$ always.

In parallel with the correspondence between propositions and sets in a classical logic, "fuzzy sets" are defined via the fuzzy membership function, with fuzzy intersection and fuzzy union given by

$$(\mu_1 \wedge \mu_2)(W) = \min\{\mu_1(W),\ \mu_2(W)\},$$
$$(\mu_1 \vee \mu_2)(W) = \max\{\mu_1(W),\ \mu_2(W)\}.$$

There is now once again a one-to-one correspondence between the propositions of fuzzy logic and fuzzy sets (although in the strict fuzzy logic the propositions True and False are discarded), and in either case a lattice is obtained satisfying the lattice axioms (i) through (iii) above. In the strict fuzzy theory, however, the lattice has no identities 0 and 1. It is therefore impossible for the lattice to have ordinary orthocomplements or to qualify as a Boolean algebra. Nevertheless, in fuzzy theory there is a "fuzzy complement," $P^{\wedge}$, defined for each proposition P as the proposition satisfying

$$\mu_{P^{\wedge}}(W) = 1 - \mu_p(W).$$

Correspondingly, each fuzzy set has a complement given as

$$\mu^{\wedge}(W) = 1 - \mu(W).$$

Notice that this complement satisfies involution (v) and DeMorgan's Laws (vii), but complementation (vi) becomes the weak statement

$$(vi')\quad 0 < \mu \wedge \mu^{\wedge} = \min\{\mu,\ 1 - \mu\},$$
$$\mu \vee \mu^{\wedge} = \max\{\mu,\ 1 - \mu\} < 1.$$

(If True and False were included, with $\mu_{True} = 1$ and $\mu_{False} = 0$, the identities rule (iv) would also hold, but the lattice would still not be complemented and hence not orthocomplemented.) Finally, the distributive rule (viii) still holds for fuzzy sets, and as a result, modularity holds. Of course, without a true complement, orthomodularity is out of the question.

3. Quantum Logic

The development of modern quantum physics has revealed that there are natural phenomena which cannot be described and reasoned about in the framework of classical logic. The behavior of elementary particles exhibits a property that can often be phrased in the form "P or Q is true even though P is not true and Q is not true." For example, "light is a wave phenomenon or a particle phenomenon" even though neither one describes light adequately. As Bohr wrote in 1929 (1961, p.96): Quantum experience

> "forces us to adopt a new mode of description designated as complementary in the sense that any given application of classical concepts precludes the simultaneous use of other classical concepts which in a different connection are equally necessary for the elucidation of the phenomena."

As such, complementary constructs provide competing alternative frames of reference: WAVE or PARTICLE = 1 (necessary) and WAVE and PARTICLE = 0 (mutually exclusive).

More specifically, the well known experiments (see, for example, Feynman) which show that when light is passed through two slits in an opaque screen, an interference pattern is formed on a photosensitive plate on the other side of the screen, even when only a single photon (irreducible packet of energy) is projected at a time. The photon must pass through one slit or pass through the other slit, but the observed behavior is not that of a photon which has simply passed through a single slit. This can be experimentally verified by covering one of the slits, at which point the interference behavior disappears.

In quantum formalisms, one considers a type of encoding of patterns and of regularities in physical experience which is at fundamental variance with classical parallel process modeling. The quantum process is described as a "non-selecting measurement" (Schwinger 1970). It occurs when two empirical processes are possible but there is no empirical procedure that is capable of distinguishing which one has actually occurred. It is then empirically meaningless to speak about there being individual events (in the classical sense of having a fully defined and fixed sample space). Quantum theory provides for this type of fundamental "ambiguity" by means of symmetries on irreducible equivalence class structures: this cannot be formally accomplished within classical models.

What, then, must give way in order to provide a logic in which to discuss these quantum phenomena? In the experiment with the photons and slits mentioned above (see Figure 3), if a photon is known to pass through slit A (by covering B, for example), it arrives at the plate according to a probability distribution centered at A'; if, on the other hand, it is known to pass through slit B, it must arrive according to a distribution centered at the point B' on the plate.

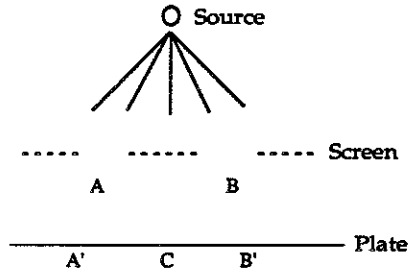


Figure 3. An Experiment Splitting a Light Beam

Letting Q be the statement "the photon passed through slit A"; R be the statement "the photon passed through slit B"; and P be the statement "the photon arrived near point C on the plate," we conclude that $P \wedge Q$ is false in almost all cases and $P \wedge R$ is also false in almost all cases. Consequently, $(P \wedge Q) \vee (P \wedge R)$ is false in almost all cases. Nonetheless, when a photon is projected, it is most frequently observed to arrive near point C. Since the photon is indivisible, it must have come through one of the slits: therefore, $P \wedge (Q \vee R)$ is true in a large per cent of the cases. This contradicts the distributive law (viii)!

The foregoing example brings in the concept of probabilistic behavior of elementary particles and expresses results in terms of distributions. It must be emphasized immediately that, unlike the fuzzy logic approach, nothing uncertain is implied about the behavior of individual particles: it is possible to measure with great precision the point of arrival of a single photon on the plate of Figure 3. Therefore, an individual photon either does or does not arrive at a point near C, and, accordingly, $P \wedge (Q \vee R)$ is either completely true or completely false for the given photon. The set of photons satisfying $P \wedge (Q \vee R)$ is therefore a "crisp" set. The allegation that $P \wedge (Q \vee R)$ is not equal to $(P \wedge Q) \vee (P \wedge R)$ is to be interpreted empirically: in experiments in which it is possible to measure $P \wedge Q$ or $P \wedge R$ (by covering one of the slits) the results obtained differ greatly from those in which $P \wedge (Q \vee R)$ is measured (by leaving both slits open). The probability distributions for photons passing through individual slits do not add up to the distribution for photons allowed to pass through either slit.

One widely used construction of a quantum logic (Greechie & Gudder 1973) is based on a collection of propositions (or "questions" or "experiments") about a physical system which are related by $x \leq y$ if and only if for every state of the system in which x is ALWAYS found to hold, y is also ALWAYS found to hold. (One must actually be a little more careful by defining the elements of the logic to be equivalence classes of propositions so as to ensure that $x \leq y$ and $y \leq x$ implies $x = y$. We will assume this has been done and ignore the formal details.) A partial order on the propositions is obtained, hence a conjunction, $x \wedge y$, can be defined as the experiment "test the truth of one of x or y at random and let that be the truth of $x \wedge y$." To see that $x \wedge y$ is the greatest lower bound of x and y, if $z \leq x$ and $z \leq y$, then in any situation that z is necessarily true, both x and y must be true, consequently $x \wedge y$ must be found to be true also. Conversely, for $x \wedge y$ to always hold in some state of the system it must be that x always holds and y always holds; thus

$$x \wedge y \leq x \text{ and } x \wedge y \leq y.$$

The disjunction in such a quantum logic is more difficult. It is not merely the experiment which says, "test the truth of x and the truth of y, and if either holds then declare $x \vee y$ true." Take for example the physical system consisting of a single toothbrush and a suitcase divided into two separate compartments (left and right). Let x be the experiment "open the suitcase and find the toothbrush in the left compartment," while y is "open the suitcase and find the toothbrush in the right compartment." The spurious $x \vee y$ just suggested would be "open the suitcase and find the toothbrush in either the left or right compartment." There are, however, other experiments such as z = "open the suitcase three times and find the toothbrush in the same side every time." This experiment will always yield "true" if x always yields "true," and likewise if y always does; however, if someone keeps switching the toothbrush from one side to the other, $x \vee y$ will always yield true but z will not. Therefore this $x \vee y$ is not the least upper bound of x and y.

The proper definition of $x \vee y$ is the greatest lower bound of all propositions z which are $\geq x$ and $\geq y$. (There may be an infinite number of such propositions, but the same definition given for $x \wedge y$ extends to any countable number of propositions.) Finally, a complement can also be defined: $x\sim$ says to perform the same experiment as for x but answer "true" if and only if x yields "false." The resulting structure is an orthomodular lattice of propositions (properties (i) through (vii) and (xi).)

A collection of functions, called "states," from the lattice to the interval [0, 1] are now added to represent the probability of obtaining the value "true" for each proposition. Each state, s, satisfies the properties

$s(0) = 0$,
$s(1) = 1$, and
$s(x \vee y) = s(x) + s(y)$ for all x, y with $x \leq y\sim$.

Moreover, the collection of states must be "full," meaning that they can be used to determine the order on the lattice:

for all x, y: $x \leq y$ if and only if $s(x) \leq s(y)$ for all states s.

This corresponds to the idea that if x is not $\leq$ y then there should exist some state of the system referred to by these propositions in which it is more likely that x is true than that y is true.

The states, which are functions from the lattice to [0, 1] can be thought of as representing the possible worlds, as do the truth functions of classical and fuzzy. They can therefore be used to define something akin to a fuzzy membership function: for any given x in the lattice, define

$$\mu'_x(s) = s(x) \text{ for all states s.}$$

The significant difference between the quantum μ'_x and the fuzzy μ_p is in the conflicting requirements whenever $x \leq y\sim$:

$$\mu_{P \vee Q}(W) = \max\{\mu_P(W), \mu_Q(W)\}, \text{ and}$$
$$\mu'_{x \vee y}(s) = s(x \vee y) = s(x) + s(y) = \mu'_x(s) + \mu'_y(s).$$

Although fuzzy and quantum logics appear to share a probabilistic/likelihood approach to the problem of describing the world, the similarity is superficial. The fuzzy logic approach treats the ideas of truth and membership themselves as not binary, whereas in quantum logic they remain as in classical logic. The probabilities, or states, of quantum logic refer to the probabilistic outcomes of performing repeated experiments which, on any given execution, return a clear-cut "yes" or "no." Where a fuzzy proposition such as "Joe is tall" has a value of, say, .8 in a world in which Joe is six feet tall, a quantum proposition such as "The toothbrush is found in the left compartment of the suitcase" has a value of 1 in a world in which, upon opening the suitcase, the toothbrush is found in the lefthand compartment. In a world in which the owner of the suitcase tends to keep his toothbrush in the left side, the corresponding quantum logic state may assign s(toothbrush in left side) a value of .8, but this is not to imply that when you open the suitcase you will not know with certainty that the toothbrush is or is not in the lefthand compartment.

What distinguishes quantum logic more dramatically from classical and fuzzy logics is the underlying non-distributivity of the quantum lattice of propositions. An orthomodular lattice of propositions divides itself into subsets of compatible (see "$\leftrightarrow$" above) propositions: any distributive lattice has all its elements compatible. Incompatible propositions are those which represent measurements or experiments which cannot be performed simultaneously. For example, the Heisenberg Uncertainty Principle says that measurement of the momentum of an elementary particle and measurement of the position of the same particle are incompatible in this sense: empirically, there is no state of simultaneous position and momentum. This and the photon experiment presented earlier cannot be described in terms of fuzzy logic but do fit the quantum logic scheme.

One of the difficulties of working with quantum logics is that the structure of orthomodular lattices is complex and not well understood. Despite a remarkable representation theorem of Piron (1976), in contrast to the Boolean algebras of classical logic, which are very regular and always representable as sets of subsets, orthomodular lattices can boast no such simple characterization (Anger 1986; Kalmbach 1983). On the contrary, researchers have been able to show extreme examples such as orthomodular lattices on which it is impossible to define even one state (Greechie 1971).

The next section presents the Hilbert space projection lattice, which has served in place of the "sets of subsets" representation of classical logic; nevertheless, Greechie (1969) has shown that there are orthomodular posets that cannot be represented even in this manner. Despite these difficulties, the need for non-classical models is evident in describing quantum phenomena (Jauch 1968; Schwinger 1970; McGoveran 1984), and there is also considerable evidence that quantum logic provides a better model for many human thought processes than do classical or fuzzy models (Oshins 1984, 1989).

4. Logic and Hilbert Space

Hilbert space has long been used in quantum mechanics as the natural environment for expressing quantum phenomena (Heisenberg 1958; Von Weizsacker 1955, 1958, 1970, 1971). Von Weizsacker, von Neumann, Birkhoff, and Mackey laid much of the mathematical foundations, while later Jauch, Piron, and Finkelstein have helped develop the theory to its present state. The lattice of closed subspaces in Hilbert space provides a meta-logic for our logic, since it contains lattices of subspaces which are orthomodular and non-distributive. Closed subspaces correspond to "projection operators" on the Hilbert space which represent orthogonal projections of the space on the subspace. Since such linear operators can be represented by (possibly infinite) matrices, the quantum logic can be represented as a class of matrices of complex numbers with the properties:

(M1)	$M^2 = M$	(idempotent)
(M2)	M commutes with its adjoint $MM^* = M^*M$	(conjugate transpose): (normal)

Hilbert spaces are complete inner-product (vector) spaces and hence have a concept of orthogonality: two vectors are orthogonal if their inner product is zero. Similarly, two projection operators are "orthogonal" if their associated subspaces (images) are orthogonal. The orthogonal complement, $M^\perp$, of a projection matrix M is another projection matrix such that

$$M^\perp M = MM^\perp = 0 \qquad \text{(the zero matrix).}$$

Finally, the intersection of closed subspaces V and W is a closed subspace $V \wedge W$, while $V \vee W$ is defined as the smallest closed subspace containing both V and W (the "span" of V and W). It is a fundamental theorem of the field that the closed subspaces of a Hilbert space form an orthomodular but non-distributive lattice, called the "projection lattice" of the space (Von Neumann 1955; Beran 1984).

In order to represent fuzzy logic in terms of projection operators on Hilbert space, we consider a particular lattice of propositions, L, and fix a proposition, P of L. Since μ maps P possible worlds into the interval (0, 1), it induces a total order on the set of worlds:

$$W1 \leq W2 \text{ if and only if } \mu_P(W1) \leq \mu_P(W2).$$

If M is a matrix whose rows are put into one-to-one, order-preserving correspondence, f, with a sequence of points from 0 to 1, we can think of placing along the diagonal the values M[k,k] = 1 if f(k) is in the range of μ_P, and M[k,k] = 0 P otherwise. This makes M into a projection operator representing the given proposition, P, and satisfying the conditions given above but in which all pairs of projections are compatible. In particular, for any such matrices, M and N,

$$MN = NM \text{ (commutative)}, \quad M^* = M \text{ (self adjoint)},$$

and the "spectrum," or set of eigenvalues, is just {0, 1}; however, the orthogonal complement, $M^{\perp}$, does not correspond to the fuzzy complement, $\mu^{\wedge}$. In fact,

$$M^{\perp}[k, k] = 1 - M[k, k] \text{ for all k, or}$$
$$M^{\perp}[k, k] = 1 \text{ if and only if } M[k, k] = 0,$$

compared with

$$M^{\wedge}[k, k] = 1 \text{ if and only if}$$
$$f(k) \text{ is in the range of } \mu_{P^{\wedge}} = 1 - \mu_P \text{ if and only if}$$
$$1 - f(k) \text{ is in the range of } \mu_P .$$

By adjoining all the orthogonal complements, a projection lattice is obtained for fuzzy logic. This lattice is, however, distributive and therefore a Boolean algebra, just as in the case of classical logic.

5. Conclusion

By comparing the lattice properties of the logics known as Classical, Quantum, and Fuzzy, it is possible to make a comparison of the power or weakness of the three logics. Classical logic produces a Boolean algebra, Quantum logic produces an orthomodular lattice, and Fuzzy logic produces a distributive but not complemented lattice. By representing membership functions by projection operators on Hilbert space, it is possible to add orthocomplements to the fuzzy lattice in a natural way and thereby obtain a Boolean algebra. In terms of the lattices, it is seen that the fuzzy logic lattice is somewhat more general than the classical by not requiring complements, but it is less general than quantum logic by requiring distributivity. Fuzzy allows discussion of sets which do not have complements but prevents the discussion of incompatible pairs of sets. The fuzzy approach talks of the fuzzy set of people who Bill likes with no concern for dividing the world into liked and unliked people; the quantum approach talks of the set of circumstances in which electrons have momentum and the set of circumstances in which they have position with no concern for the fact that, although electrons always have position or momentum, there are many occasions in which electrons do not have position and do not have momentum. All the photons that reached the plane behind the screen went through slit A or through slit B, but many of them did not go through slit A and did not go through slit B.

A final illustration: a suitcase is constructed with two compartments in such a way that only one compartment can be opened at a time. A particular shirt may be in the left side, in the right side, or still in the drawer back home. Classical logic would divide the set S of possible worlds in which the shirt is in the suitcase into two mutually exclusive and all inclusive subsets: A = worlds in which the shirt is in the left side, and B = worlds in which the shirt is in the right side. Thus,

$$S = A \vee B \quad \text{and} \quad 0 = A \wedge B.$$

Fuzzy logic would ascribe to each of the possible worlds a likelihood that the shirt is in the left side and a likelihood that it is in the right side and thereby create a likelihood equal to the minimum of these two that the shirt is in both the left and the right sides. Furthermore, the likelihood of the shirt being in the suitcase will be assigned the maximum likelihood of its being in the left or in the right sides. Lastly, quantum logic will only consider worlds in which the shirt is FOUND to be in the right side, FOUND to be in the left side or FOUND to be at home in the drawer. If P is the proposition "found on left," Q = "found on right," and R = "found in drawer," then any state s will satisfy

$$s(P) + s(Q) = s(P \vee Q) \quad (\text{because } Q \leq P^{\sim})$$
$$= s(\sim R) \quad (\text{by what we know about shirts}).$$

The set of worlds for which the shirt is always found in the right side is the set of states for which s(Q) = 1. There may also be states which say that s(Q) = .7, s(P) = .2, and s(R) = .1, but here $s(P \wedge Q) = s(0) = 0$ and $s(P \vee Q) = .9 = 1 - s(R)$. There is no possibility that the shirt is in both sides or nowhere at all.

References

Anger, F., J. Sarmiento, and R. Rodriguez. 1986. "Representative graphs of r-regular partial planes and representation of orthomodular posets." *Journal of Discrete Applied Mathematics*, 15, 1-10.

Beran, L. 1984. *Orthomodular Lattices, Algebraic Approach.* D. Reidel Pub. Co., Dordrecht.

Bohr, N. 1961. *Atomic Theory and the Description of Nature.* Cambridge Univ. Press, Cambridge, England.

Feynman, R.P. 1951. "The concept of probability in quantum mechanics." *Second Berkeley Symposium on Mathematical Statistics and Probability*, Univ. CA. Press, Berkeley, CA.

Greechie, R.J. 1969. "An orthomodular poset with a full set of states not embeddable in Hilbert space." *Caribbean Journal of Mathematics*, 1, 15-26.

Greechie, R.J. 1971. "Orthomodular lattices admitting no states." *Journal of Combinatorial Theory*, 10, 119-132.

Greechie, R.J. and S.P. Gudder. 1973. "Quantum Logics." In *Contemporary Research in the Foundations and Philosophy of Quantum Theory.* Reidel, Boston, MA.

Heisenberg, W. 1958. "Language and Reality in Modern Physics." *Physics and Philosophy.* Harper & Row, NY.

Jauch, J.M. 1968. *Foundations of Quantum Mechanics.* Addison-Wesley, Reading, MA.

Jauch, J.M. 1972. "On Bras and Kets." *Aspects of Quantum Theory.* University Press, Cambridge, MA.

Kalmbach, G. 1983. *Orthomodular Lattices.* Academic Press, NY.

McGoveran, D. 1984. *Discrete Approaches to Natural Philosophy.* Alternative Natural Philosophy Association. Boulder Creek, CA.

Oshins, E. 1984. "A Quantum Approach to Psychology: Spinors, Rotations, and Non-Selecting Ambiguity- Part I: Quantum Logic Representations in Psychology." In *Discrete Approaches to Natural Philosophy.* ed. D. McGoveran.

Oshins, E. 1989. Quantum psychology and future directions. Paper presented to panal on "The future of scienific psychology." AAAS Annual Meeting, San Fransisco, CA.

Oshins, E., D. Adelson, and D. McGoveran. 1984. Clarifying fuzzy logic: A spectral decomposition and iconic realization. In *Discrete Approaches to Natural Philosophy*, ed. D. McGoveran. Alternative Natural Philosophy Association. Boulder Creek, CA.

Oshins, E. and D. McGoveran. 1980. "...thoughts about logic about thoughts...: The Question of Schizophrenia?." *Systems Science and Science.* Proceedings of the 24th Meeting of the Society for General Systems Research, ed. B.H. Banathy, (Jan).

Piron, C. 1976. *Foundations of Quantum Mechanics,* W. A. Benjamin, Inc. Reading, MA.

Schwinger, J. 1970. *Quantum Kinematics and Dynamics.* W. A. Benjamin, Inc., NY.

Von Neumann, J. 1955. *Mathematical Foundations of Quantum Mechanics.* Princeton Univ. Press, Princeton, NJ.

Von Weizsacker, C.F. 1955. "Komplementariat und Logic." *Die Naturwissenschaften,* 42, 521-529, 545-555.

Von Weizsacker, C.F. 1958. "Die Quantentheorie der Einfachen Alternative." *Zeitschrift fur Naturforschung,* 13a, 245-253.

Von Weizsacker, C.F. 1970. "Philosophical Foundations of Quantum Theory." MAT-17-1970, The Institute of Mathematical Sciences, Madras, India.

Von Weizsacker, C.F. 1971. *The Unity of Nature.* Farrar, Straus, and Giroux, Inc., NY.

Zadeh, L. 1965. "Fuzzy Sets." *Information Control,* 8, 338-353.

MANIPULATION PLANNING - AN EXAMPLE USING KNOWLEDGE-BASED MODELING AND AUTOMATIC PROGRAM GENERATION

Mark M. Thomas
Automation Software Innovations
P. O. Box 609
Edgewater, FL 32032

Jeffrey M. Becker
Martin Marietta Astronautics
P. O. Box 179
Denver, CO 80201

ABSTRACT

This paper discusses manipulation-planning issues and research, suggests an implementation strategy, and reviews some preliminary development results. Results include software models for a planar peg-insertion task, and a program-generation implementation, in which a domain-independent task planner synthesizes sequences of low-level routines to perform computation goals.

INTRODUCTION

Man will soon be able to send robotic vehicles to other planets, equipped with manipulators that are stronger, more precise, more dextrous and possess greater finesse than humans. Probably more significant, these devices will also be better at manipulation planning - evaluating their environment and planning the use of their improved manipulation capabilities to achieve high-level mission goals. Advanced robotic manipulators will perform a diversity of terrestrial and space applications. The primary motivations for their use are to remove operators from hazardous environments, improve product quality and reliability, reduce production costs, and provide prosthetic components that enhance the quality of life for handicapped individuals.

The greatest challenge and opportunity for improving our ability to use robotic systems lies in the development of tools and techniques for intelligent robotic control. Intelligent robotic control refers to the computer control of robotic systems in response to high-level (often "fuzzy" or qualitative) user commands by incorporating intelligent functions, such as memory utilization, adaptation or learning, and multi-level decision making; it involves the application of techniques from the fields of Artificial Intelligence, Operations Research and Control (Saridis 1983). Some of the characteristics that make intelligent control necessary and distinguish it from conventional control are planning complexity (usually due to the number of control variables and the nonlinearity of terms involving them), multi-objective performance and constraint criteria, imprecise nature of goals and constraints, incorporation of feedback from a variety of sensors, and need for temporal task decomposition.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

MANIPULATION PLANNING

A manipulation planner is one component in a planning/control hierarchy for robotic manipulators. It receives goals from a higher-level (Task Level) planner, each of which specifies an individual manipulation activity. The manipulation planner maps these goals, under constraints imposed by the environment and the robot hardware and software, into a sequence of commands, which the robot controller uses to drive the manipulator and tooling. These commands define the robot trajectories, control modes and activities. Figure 1 shows some characteristics and examples of manipulation planner input and output.

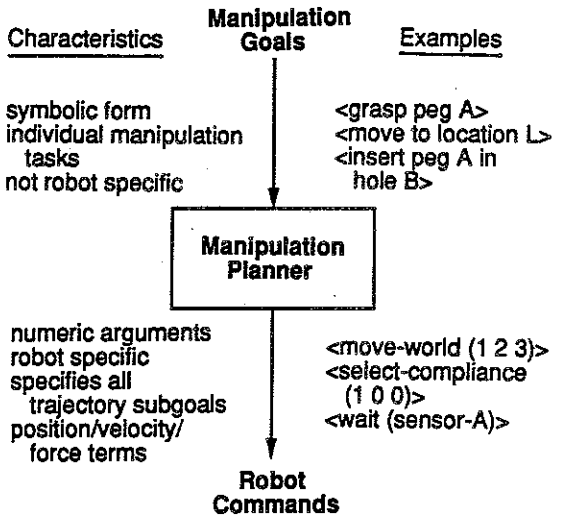


Figure 1. Manipulation planner interfaces.

There are three fundamental tasks that a manipulation planner must perform:

1. Generate collision-free motion trajectories, which satisfy additional motion requirements, such as joint position and rate limits, and optimize performance criteria, such as travel time.

2. Determine grasp and approach configurations for part acquisition, which satisfy constraints imposed by part and gripper geometry, part mounting, etc., and optimize criteria such as grasp strength and stability.

3. Define force/impedance commands for tasks that require interaction between the manipulator and environment.

The manipulation planner generally uses models of the robot and environment geometry, and must compensate for the fact that there is some uncertainty in these models. The environment model is often defined a priori, but can be measured during operation by higher-level sensing functions, such as processing of video camera images.

Algorithms for planning collision-free motions that have been proposed and simulated or prototyped, generally fall into two categories: 1) graph-search algorithms and 2) incremental-motion/local-optimization approaches. In the graph-search algorithms, the interactions between the robot or manipulator and the environment are processed to generate a graph of feasible trajectory segments. This graph is then searched for a connected path from the start node to the goal node, using standard techniques such as breadth-first or A*. A seminal work by Lozano-Perez and Wesley on planning paths among polyhedral objects (Lozano-Perez and Wesley 1979) typifies the approach; it describes several fertile techniques, including configuration space, visibility graphs and A* search. A number of other works use similar graph-search algorithms (Brooks 1984; Lozano-Perez 1986; Hoermann 1987). One difficult issue in the general planning case is the handling of spatial rotations (Canny 1985; Donald 1987). Fundamental topics associated with these approaches are their computational complexity and completeness (Yap 1987).

The incremental-motion techniques rely on local optimization or low-level control functions to instantaneously satisfy the collision-avoidance constraints during operation. An implementation by Khatib, for example, defines potential fields in the environment that lead to forces of attraction from the goal and repulsion from the obstacles in the environment (Khatib 1985). These force contributions are summed with other control terms to provide joint drive commands that compel the manipulator to slide around obstacles and toward the goal. Other works in this vein use optimization to define incremental position updates or path segments (Grechanovsky and Pinsker 1983; Suh and Shin 1988). One implementation uses this goal-attraction/ obstacle-repulsion approach to generate nodes for a search graph during the search (Graham et al. 1987).

Algorithms based on the graph-search approach incorporate global information in the planning, are generally complete at a resolution (i.e., if a solution exists, it will be found), and are very efficient when the graph can be constructed a priori. On the other hand, they can be somewhat difficult to implement, represent only a small subset of the feasible trajectories, and can require parameter discretization, which leads to a very large planning graph. The properties of the incremental-motion techniques are nearly opposite - they do not require discretization, can optimize a variety of criteria simultaneously, handle moving obstacles directly, are fairly simple to implement, but are not complete, use only local information for planning and generally require a higher-level planner to avoid local optima. Algorithms of both types have been prototyped and demonstrated in laboratory environments.

The development of multi-fingered robotic hands has spurred strong interest in modeling and planning of part acquisition strategies. Much of this research relies heavily on fundamental kinematic and quasi-static force transformations that represent the interactions between the gripping fingers and the objects being manipulated (Montana 1988; Nguyen 1988). Several works have investigated the synthesis of grasps that are optimal with respect to criteria such as grasp stiffness, stability and resistance to slip (Cutkosky 1985; Volz et al. 1987; Li and Sastry 1988). Several types of finger contact are considered. These efforts deal mainly with characterizing the grasp itself and constraints on subsequent motion, but less with the grasping operation. One novel algorithm considers the pushing and sliding effects during gripping of small polygonal objects lying on a table; gripping strategies are synthesized that lead to stable grasps in the presence of some uncertainty in the object's original position (Brost 1988).

The desire to apply autonomous robots to a wider array of manipulation tasks and environments, and the realization that very accurate positioning and modeling are often not feasible, has led to significant recent interest in fine-motion planning and control. Fine-motion control refers to the simultaneous control of positions and forces of interaction with the environment, or control in which uncertainties in the robot and environment models and state dominate the operation. Simple hardware devices that control interaction forces have proven highly successful for limited classes of well-structured fine-motion tasks, such as manufacturing assembly (Nevins and Whitney 1978). Algorithms for software control of interaction forces have been demonstrated in laboratories (Raibert and Craig 1981; Hogan 1985), but suffer from difficulty of programming and a lack of thorough analysis. The issues associated with autonomous programming of these strategies are also being addressed (Lee and Huang 1985). One planning technique uses the concepts of friction cone, configuration space and damping control to reduce fine-motion planning tasks to equivalent geometric problems (Lozano-Perez et al. 1984; Erdmann 1986). Technologies for both grasp and fine-motion planning are in their infancy - some basic techniques for simplified cases are being prototyped in laboratories, and analytical foundations for further work are being developed.

A MODEL-BASED PLANNING PARADIGM

The basic manipulation-planning problem is to generate a sequence of reference trajectories that lead to a desired goal and satisfy a set of operation constraints. The trajectories, goal and constraints are specified in terms of generalized positions, rates, forces and relations among these terms. The complexity of this planning problem derives from the number of independent variables and the nonlinearity of the equations that characterize system operation in terms of these variables. It is difficult to directly represent and parametrize the trajectories that satisfy the motion constraints and accomplish the goal. On the other hand, manipulator operation can usually be well characterized for planning and control purposes by a few tractable models of mechanical system operation and a database of reasonable size, which contains the parameters for the particular system. Automatic methods of formulating equations of motion for analysis and simulation uses of these models are becoming well developed. These considerations suggest the use of a hypothesize-and-test planning approach, in which prospective trajectories are generated and evaluated by simulation (Myers and Agin 1983).

However, using simple models directly and evaluating a variety of control strategy options can lead to a burdensome amount of calculation. One solution is to encode promising strategies for different planning cases in rules, and employ production-system or pattern-matching algorithms for selecting applicable strategies. It will ultimately be

up to the computer to learn the rules from evaluation of system operation and simulations (e.g., Dufay and Latombe 1984). A suggested approach for simplifying the programming of these analysis and planning tools is knowledge-based modeling. In it, the programmer defines useful object types and behaviors, and rules for their application and interaction. The user prescribes a system model by selecting applicable objects, defining some of their properties, and specifying what modeling results are of interest. The computer selects the applicable models and processing steps, and performs the computations, state updates, and input and output processing. This removes the burden of selecting the program control structure from the programmer and user. In an advanced knowledge-based simulation, the computer would also generate scenarios, suggest and evaluate solution strategies, learn from previous runs, and display and explain its results and reasoning (Reddy 1987).

PLANAR PEG INSERTION

These modeling and planning techniques are being investigated for the example of planar peg insertions. This simplified task demonstrates the fundamental features of fine-manipulation, including multi-dimensional motion planning, interaction with the environment, transitions between various modes of constraint and interaction, and importance of uncertainty. Tools for modeling and simulating peg insertions have been developed, including position and velocity calculation procedures, automatic and manual controllers (the potential for learning rules from simulated control by an experienced operator is alluring), graphic displays of operation, a closest-approach procedure, and models of interaction forces and friction. The implementation uses a generalized coordinate formulation. The basic modeling objects are motions and forces, and low-level transformation routines are defined for these objects.

DOMAIN-INDEPENDENT TASK PLANNER

A domain-independent Task Planner synthesizes computation strategies for analysis or simulation tasks, using the low-level transformations. This section describes the Task Planner fundamentals and the next section discusses its use for an automatic program generation implementation. The Task Planner is a hierarchical, nonlinear, backward-chaining planner. It belongs to the same family as ABSTRIPS (Sacerdoti 1973), Nonlin (Tate 1977), NOAH (Sacerdoti 1977), SIPE (Wilkins 1984) and TWEAK (Chapman 1984). It has been used for control of an integrated robotic system that performed inspection, construction and maintenance tasks (Becker and Garrett 1987). The Task Planner is hierarchical in the sense of ABSTRIPS - goals are assigned a level of difficulty and only the highest level unsatisfied goals are worked on. Nonlinear plans are achieved by 1) delaying ordering decisions between selected operators until assertion or resource conflicts make ordering necessary, and 2) allowing serendipitous goal reduction.

The Task Planner is built on a unique temporal logic system (TLS). TLS is an embedded logic programming language written in Common Lisp. Like Prolog, TLS allows programs to be written in terms of facts and rules. It uses the same backward-chaining proof mechanism as Prolog. However, instead of all assertions being "at the same time," assertions are indexed to time points. The temporal database (called a *time map*) is a directed, acyclic graph of time points, which makes it especially well-suited for nonlinear planning systems. TLS automatically manages the propagation of assertions forward in time and clips propagation when a conflicting assertion is reached. TLS also automatically checks that the duration constraints are consistent using a dynamic programming algorithm.

Besides regular facts and rules as provided in Prolog, called *adders* in TLS, TLS provides *user* and *deleter* assertions. Adders propagate to successors of the time point at which they are asserted; neither users or deleters are propagated. A deleter is a retraction installed in the time map. Adder propagation may be clipped by a matching deleter or conflicting adder. Users are like floating queries installed in the time map. A user is *satisfied* if it unifies with a propagated adder; satisfied users may be protected. When conflicting facts are asserted at parallel (or the same) time points, a *fact conflict* exists. Adders involved in a fact conflict are depropagated to reflect their uncertain status. A time interval is associated with two time points (start and end). Its duration is specified numerically as a minimum and maximum, and can be declared as either an estimate or constraint. The time map contains slots for recording unsatisfied users, user protection violations, fact conflicts and duration conflicts.

Domain knowledge used by the Task Planner consists of operators, goals, goal axioms, and object descriptions. Operators describe the capabilities available for modifying the state of the problem space - the plan consists of a sequence of operators to apply to change from an initial state to a goal state. Each operator has prerequisites and preconditions that must be true for the operator to apply. Preconditions become new subgoals but prerequisites must be satisfied at the time the operator is selected. Temporal constraints allow the knowledge engineer to specify orderings getween the times at which the preconditions are accomplished, but are not used in this application. Constraints limit the allowable bindings of the operator variables. Object descriptions are encoded as facts and rules that can be used when determining if these constraints are true.

Goals define the vocabulary of the Task Planner. Preconditions, prerequisites and adders are all goals. The knowledge engineer associates a level with each goal, used for directing search during planning. The adders field of an operator specifies the new conditions that hold after application of the operator (i.e., the goals it satisfies). Deleters are derived from the adders using goal axioms. Goal axioms describe relationships between goals, i.e., which other goals are added or denied when a goal is asserted. Operators have a number of other fields that are not used in this application, including: resources (the facilities utilized by the operator, used to determine whether operators can occur in parallel); and message pattern, command stream and reply pattern (these define the command vocabulary used during execution of the operator).

The Task Planner algorithm is as follows. The Task Planner first finds the set of highest level unsatisfied goals and the set of operators that can plan for any of these goals. Possible instantiations of operators are found by finding proofs of the operator instantiation constraints in the planning domain description using the TLS backward-chaining proof mechanism. Operators are rated according to domain-independent heuristics. The "best" operator is installed in the time map by creating a time interval that represents the start time, end time and duration of the operator. Additional intervals are asserted so that the operator is constrained to occur within the plan and before all operators whose subgoals it plans for. Goals that the new operator plans for are protected so that the

time map will indicate when these goals are clobbered. Goals that are satisfied but not explicitly planned for are not protected; if one becomes unsatisfied, it is said to be reactivated.

Preconditions of the new operator are installed as user assertions in the time map at the start time point of the operator. Postconditions are installed as adder and deleter assertions in the time map. If the fact-conflict slot of the time map is not empty after the operator is installed, the planner uses heuristics to choose a preferred odering and installs additional time intervals in the time map. Backtracking choices are recorded for alternative operators and orderings. Backtracking occurs when a planned-for goal is clobbered or when no satisfactory operator or ordering can be found. The planning algorithm continues until all goal and user assertions are satisfied and no fact or time conflicts exist.

AUTOMATIC PROGRAM GENERATION

Automatic programming is the automation of some part of the programming process; it differs from compilation of a programming language in that the transformations it uses are not applied in a rigid, predetermined manner, but rather on the basis of analysis and exploration of their potential application (Barr and Feigenbaum 1981). The operators in this Task Planner implementation are analysis routines, used for computing state transitions, force and kinematic transformations, displaying graphics, etc. The objects in the implementation are data structures, as created by Common Lisp *defstruct* forms. Their individual fields are either values (any Common Lisp object) or other structure objects. The goals are values in the leaf nodes of the objects (i.e., computation of these values). Compilable executive routines are created because analysis programs are usually called repeatedly. Also, the computations themselves can be time consuming and should not be performed during search for a computation strategy, in general.

As an illustrative example, consider finding the position and orientation of a reference frame in a moving body of a manipulation system. The *frame* structure contains two *basic-frame* objects, one representing the local motion of the frame relative to some part and the second defining its absolute motion (i.e., relative to a world reference). Each basic-frame object contains a displacement matrix (disp-mat) and a *motion* object, which is the basic structure for a vector of generalized motion coordinates. All object types also contain name and identifier fields.

Object types are defined using a macro *defstruct-apg*, which provides an enhanced version of Common Lisp defstruct. The syntax is like that of defstruct, with the addition of two keyword options to the individual slot descriptions: 1) type - indicates the object type of the value in this slot, and 2) upward - a boolean. The upward field indicates if the slot contains a higher-level object and is used for determining printing and default initialization behaviors. Execution of the defstruct-apg form defines a new data structure type, a print function for structures of that type (and myriad functions created by defstruct), and the logic system assertions for the new type. An object class is defined for the new object type. For each leaf slot of an object (i.e., slot for which no type-field is specified), a Task Planner goal is defined. Accessor functors are defined for the remaining slots (functors are the functions of the Task Planner domain). Figure 2 illustrates some of the Task Planner object description rules created for a frame object. The defstruct-apg form also generates a creation function,

```
(DEFSTRUCT-APG FRAME
  "Represents motion of a reference frame."
  (PART        NIL :TYPE 'PART     :UPWARD T)
  (LOCAL-FRAME NIL :TYPE 'BASIC-FRAME)
  (ABS-FRAME   NIL :TYPE 'BASIC-FRAME))

(MAKE-FRAME :NAME 'EXAMPLE-FRAME
            :PART (MAKE-PART))
```

a) some defining forms

```
(ENUMERATION MOTION NIL)
(GOAL (MOTION-DISP MOTION BOOLEAN) :LEVEL 3)
(ENUMERATION BASIC-FRAME NIL)
(GOAL (BF-DISP-MAT BASIC-FRAME BOOLEAN) :LEVEL 3)
(FUNCTOR (BF-MOTION BASIC-FRAME MOTION))
(ENUMERATION FRAME NIL)
(FUNCTOR (FRAME-PART FRAME PART))
(FUNCTOR (FRAME-LOCAL-FRAME FRAME BASIC-FRAME))
(FUNCTOR (FRAME-ABS-FRAME FRAME BASIC-FRAME))
```

b) some facts created during defstruct

```
(TYPE MOTION-0 MOTION)
(TYPE BASIC-FRAME-0 BASIC-FRAME)
(BF-MOTION BASIC-FRAME-0 MOTION-0)
(TYPE MOTION-1 MOTION)
(TYPE BASIC-FRAME-1 BASIC-FRAME)
(BF-MOTION BASIC-FRAME-1 MOTION-1)
(TYPE EXAMPLE-FRAME FRAME)
(FRAME-LOCAL-FRAME EXAMPLE-FRAME BASIC-FRAME-0)
(FRAME-ABS-FRAME EXAMPLE-FRAME BASIC-FRAME-1)
```

c) some facts created during instantiation

```
(BF-DISP-MAT BASIC-FRAME-0 NIL)
(MOTION-DISP MOTION-0 T)
(BF-DISP-MAT BASIC-FRAME-1 NIL)
(BF-DISP-MAT PART-0 T)
(MOTION-DISP MOTION-1 NIL)
```

d) initial state assertions

Figure 2. Reference-frame positioning example.

which makes new instances of the object type and asserts the corresponding facts (see Fig. 2).

The absolute position and orientation of a reference frame are specified by the disp-mat field of the basic-frame structure in the abs-frame slot. Two low-level routines are available for computing this displacement matrix: 1) *frame-abs-disp-mat*, which uses the local disp-mat of the frame and the disp-mat of the part containing the frame; and 2) *basic-frame-disp-from-pos*, which computes the displacement matrix for any basic-frame from its three-component position vector (a direct transformation among position specifications). Figure 3 shows the operator definitions for these routines. An example frame instance was created and it was asserted that the part disp-mat and local frame's position vector were known. The program generated a two-step routine for computing the absolute disp-mat: 1) call *basic-frame-disp-from-pos* to compute the local frame's disp-mat and 2) call *frame-abs-disp-mat*. The Task Planner can also generalize the result, creating a macro operator with no bound variables that can be applied whenever a similar planning problem is encountered.

SUMMARY

Manipulation planning is a complex task, exhibiting most of the features of intelligent control. Much research

is being performed and several innovative algorithms have been developed for the three main tasks: collision avoidance, part acquisition and fine-motion planning. Knowledge-based modeling and simulation are tools that can simplify manipulation planning and analysis tasks. In the work described here, a domain-independent task planner was used to create executive analysis modules from low-level kinematic and dynamic transformation routines. This application does not use several of the advanced features of the task planner, such as time intervals, uncertainty propagation and operator reordering. It could benefit from additional features such as optimal search and storing fact justifications. Future efforts will involve extending the implemented analysis capabilities, automating the generation of operators from analysis routines, separating computations for values that vary over time from those do not, and generating efficient computation strategies.

```
(STATIC-OPERATOR (FRAME-ABS-DISP-MAT $F $P $BFA $BFL)
  :CONSTRAINTS ((TYPE $F FRAME) (TYPE $P PART)
     (TYPE $BFA BASIC-FRAME) (TYPE $BFL BASIC-FRAME)
     (FRAME-PART $F $P) (FRAME-ABS-FRAME $F $BFA)
     (FRAME-LOCAL-FRAME $F $BFL))
  :ADDERS ((BF-DISP-MAT $BFA))
  :PRECONDITIONS (LOCAL-DISP-MAT (BF-DISP-MAT $BFL)
     PART-DISP-MAT (BF-DISP-MAT $P)))

(STATIC-OPERATOR (BASIC-FRAME-DISP-FROM-POS $BF $M)
  :CONSTRAINTS ((TYPE $BF BASIC-FRAME) (TYPE $M MOTION)
     (BF-MOTION $BF $M))
  :ADDERS ((BF-DISP-MAT $BF))
  :PRECONDITIONS ()                    ;no operators
  :PREREQUISITES ((MOTION-DISP $M)))   ; plan for this
```

Figure 3. Operators for computing disp-mat of a basic-frame.

REFERENCES

Barr, Avron and Edward A. Feigenbaum (editors). 1981. *The Handbook of Artificial Intelligence*, William Kaufmann, Inc., Los Alto, CA, Vol. II, 297ff.

Becker, J. and F. Garrett. 1987. "An Architecture for Intelligent Task Automation." In *Proceedings AAAI-87*. 672-676.

Brooks, Rodney A. 1984. "Planning Collision Free Motions for Pick and Place Operations." In *Robotics Research*. MIT Press, Cambridge, MA, 5-37.

Brost, Randy C. 1988. "Automatic Grasp Planning in the Presence of Uncertainty." *International Journal of Robotics Research*, 7(1): 3-17.

Canny, John. 1985. "A Voronoi Method for the Piano-Movers Problem." In *Proceedings IEEE International Conference on Robotics and Automation*, St. Louis, MO, 530-535.

Chapman, D. 1984. "Planning for Conjunctive Goals." MIT Technical Report 802, Cambridge, MA.

Cutkosky, Mark R. 1985. *Robotic Grasping and Fine Manipulation*. Kluwer Academic Publishers, Boston, MA.

Donald, Bruce R. 1987. "A Search Algorithm for Motion Planning with Six Degrees of Freedom." *Artificial Intelligence*, 31: 295-353.

Dufay, Bruno and Jean-Claude Latombe. 1984. "An Approach to Automatic Robot Programming Based on Inductive Learning." In *Robotics Research*. MIT Press, Cambridge, MA, 97-115.

Erdmann, Michael. 1986. "Using Backprojections for Fine Motion Planning with Uncertainty." *International Journal of Robotics Research*, 5(1): 19-45.

Graham, J., M. Thomas, R. Glade, T. Holt and G. Bruno. 1987. "Intelligent Task Automation Path Planner." Intelligent Task Automation Phase II Seventh Quarterly Report, Martin Marietta Aerospace, Denver, CO.

Grechanovsky, Eugene and I. Sh. Pinsker. 1983. "An Algorithm for Moving a Computer-Controlled Manipulator While Avoiding Obstacles." In Proceedings Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, FRG, 807-813.

Hoermann, K. 1987. "A Cartesian Approach to Findpath for Industrial Robots." In *Languages for Sensor-Based Control in Robotics*, Springer-Verlag, Berlin, 425-450.

Hogan, Neville. 1985. "Impedance Control: An Approach to Manipulation; Part 1 - Theory, Part 2 - Implementation, Part 3 - Applications." *ASME J. of Dynamic Systems, Measurement and Control*, Vol. 107: 1-24.

Khatib, 0. 1985. "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots." In *Proceedings IEEE International Conference on Robotics and Automation*, St. Louis, MO, 500-505.

Lee, C. S. G. and D. Huang. 1985. "A Geometric Approach to Deriving Position/Force Trajectory in Fine Motion." In *Proceedings IEEE International Conference on Robotics and Automation*, St. Louis, MO. 691-697.

Li, Zexiang and S. Shankar Sastry. 1988. "Task-Oriented Optimal Grasping by Multifingered Robot Hands." *IEEE Journal of Robotics and Automation*, Vol. RA-4: 32-43.

Lozano-Perez, Tomas and Michael A. Wesley. 1979. "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles." *Communications of the ACM*, Vol. 22, 560-570.

Lozano-Perez, Tomas, Matthew T. Mason and Russel H. Taylor. 1984. "Automatic Synthesis of Fine-Motion Strategies for Robots." In *Robotics Research*. MIT Press, Cambridge, MA, 65-96.

Lozano-Perez, Tomas. 1986. "A Simple Motion Planning Algorithm for General Robot Manipulators." In *Proceedings AAAI-86: Science - Perception and Robotics*, 626-631.

Montana, David J. 1988. "The Kinematics of Contact and Grasp." *International Journal of Robotics Research*, 7(3): 17-32.

Myers, J. K. and G. J. Agin. 1983. "A Supervisory Collision-Avoidance System for Robot Controllers." In *Robotics Research and Advanced Applications*, ASME Publications, New York, 225-232.

Nevins, J. L. and D. E. Whitney. 1978. "Computer-Controlled Assembly." *Scientific American*, Vol. 238 (Feb): 62-74.

Nguyen, Van-Duc. 1988. "Constructing Force Closure Grasps." *International Journal of Robotics Research*, 7(3): 3-16.

Raibert, M. H. and J. J. Craig. 1981. "Hybrid Position/Force Control of Manipulators." *ASME Transactions J. of Dyn. Systems, Meas. and Control*, Vol. 102: 275-282.

Reddy, Ramana. 1987. "Epistemology of Knowledge Based Simulation." *Simulation*, 48(4): 162-166.

Sacerdoti, E. 1973. "Planning in a Hierarchy of Abstraction Spaces." In *Proceedings IJCAI-73*: 412-422.

Sacerdoti, E. 1977. *A Structure for Plans and Behavior*. North-Holland, New York.

Suh, Suk-Hwan and Kang G. Shin. 1988. "A Variational Dynamic Programming Approach to Robot-Path Planning With a Distance-Safety Criterion." *IEEE Journal of Robotics and Automation*, Vol. RA-4: 334-349.

Tate, A. 1977. "Generating Project Networks." In *Proceedings IJCAI-77*: 888-893.

Volz, Richard A., Jan Wolter, James Barber, Rajiv Desai and A. C. Woo. 1987. "Automatic Determination of Gripping Positions." In *Languages for Sensor-Based Control in Robotics*, Springer-Verlag, Berlin, 481-505.

Wilkins, D. 1984. "Domain-Independent Planning: Representation and Plan Generation." *Artificial Intelligence*, 22: 269-301.

Yap, Chee-Keng. 1987. "Algorithmic Motion Planning." In *Advances in Robotics, Vol. 1: Algorithmic and Geometric Aspects of Robotics*, Lawrence Erlbaum Associates, Hillsdale, NJ, 95-143.

Exploration Complexity of 2D Environments With Polygonal Obstacles

John D. McKendrick
and
Dr. L. Cheng
Department of Electrical and Computer Engineering
Florida Atlantic University
Boca Raton, Florida 33431

ABSTRACT

The exploration of unknown environments by autonomous robots presents many challenges. The problem of representing what the robot knows about the environment, such as what partially complete knowledge can be used without error, is essential to successful exploration. Basic issues of how to characterize object configurations in unknown, randomly structured two dimensional environments are examined. The focus is to seek a new approach to solving the problem of robot exploration and acquisition of knowledge about the environment through the use of non-distance criteria. Consideration is given to the development of a entropy-based path complexity metric for describing the general path complexity of the environment once it has been explored.

INTRODUCTION

Most approaches for any type of spatial motion analysis are made based on the measurement of the distances between prospective positions. Once a network of distances between potential points (ie., graph, map, etc.) has been developed, attempts can be made to minimize the need to move long distances between prospective position.

This research has focused on the problem of the acquisition of spatial knowledge about a potential operating environment. This problem involves exploration of a given environment to discover the spatial relationships between obstacles located in the environment. The term exploration can connote synonymous attributes that include: mapping, spatial knowledge acquisition, terrain model acquisition, knowledge acquisition, learning, data base development, etc. Previous research has focused on an attempt to minimize the distance required to move to complete the necessary exploration (knowledge acquisition) task. This research has focused on an alternate approach to the exploration process. This new approach seeks to utilize spatial relationships between obstacles and the environment. This approach is, however, quite old and might be considered a topological approach.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

NON-DISTANCE POSITION REFERENCING

Background

There has long been an interest in the development of a method for characterizing position without using distance. This approach can in part be traced to an early on expressed hope of Leibniz that a method could be found to describe position that did not employ a distance metric (Wilson,1984):

> *"I am not content with algebra, in that it yields neither the shortest proofs nor the most beautiful constructions of geometry. Consequently, in view of this I consider that we need yet another kind of analysis, geometric or linear, which deals with position as algebra deals with magnitude..."*

Euler in addressing the now well known problem of the Konigsberg Bridges took up the problem of the geometry of position:

> *"In addition to that branch of geometry which is concerned with magnitudes and which has always received the greatest attention, there is another branch, previously almost unknown, which Leibniz first mentioned, calling it the geometry of position [geometria situs]. This branch is concerned only with the determination of position and its properties: it does not involve measurements, or calculations made with them..."*

Euler did not propose what techniques nor what problems were relevant to the geometry of position other than to say that he considered the Konisberg Bridges as a problem needing a new approach. As the problem was posed, its solution involved only an relative location specification rather than an exact (x, y) position.

Another approach to a geometry of position was that of projective geometry and in particular to the development of projective geometry by Carnot in his Geometrie de Position published in 1803. Another development toward this approach was that of A.F. Mobius who introduced homogeneous coordinates in his book Der Barycentrische Calcul published in 1827.

A Practical Example

To demonstrate how relative position alone can be used to characterize position, consider the regions external to a triangular polygon as shown in Figure 1. The convention of proceeding counter-clockwise around the polygon is adopted. After choosing a vertex to start from, the regions will be either to the right-of the extended line that is directed from the starting

vertex to the next counter-clockwise vertex or to the left-of that line. Sign values (+/-) can be used to denote the left-of (+) or right-of (-) relationship. Additionally, by adopting a naming relationship in which the first item in a list relates to edge A, the second item to edge B and the third item to edge C, the list representation (- + +) would denote the region to the right of edge A, to the left of edge B and to the left-of edge C. A negetive sign implies that the corresponding edge is visible from any point in that region.

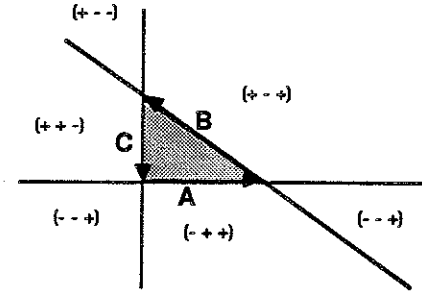


Figure 1. Non-Distance Definition of Position Relative to Triangular Polygon

This topological descriptor could be applied to characterizing position relative to two polygons as shown in Figure 2.

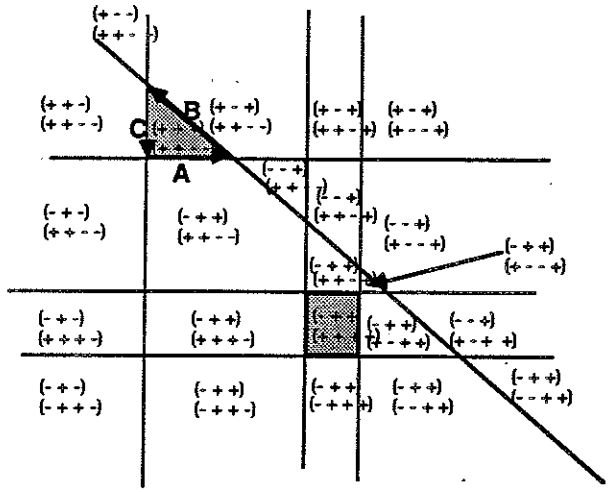


Figure 2. Topological Relationships Between Two Convex Polygons

The above topological metric to characterize position contrasts with the typical distance metric that is typically used in spatial problem solving. The topological metric addresses a logical or symbolic relationship while distance addresses a numerical relationship. Subsequently, it will be shown that a topological based heuristic can be used in place of a traditional numerical heuristic to complete the process of exploring a two dimensional environment cluttered with convex and concave polygonal obstacles.

AUTONOMOUS EXPLORATION AND NAVIGATION

Before considering the problem of topological based exploration, it is necessary to outline the needs of a mobile system that can operate autonomously in an unknown environment and to briefly describe how that system might employ traditional distance measuring techniques to explore and to navigate. Here the term *exploration* is used to denote that process in which spatial information is acquired for future (albeit, immediate) use and *navigation* is that process that can guide a mobile system safely past known obstacles. Autonomous operation includes the sub-problem of spatial exploration by a mobile learning system that must be able to simultaneously control its movements while continuing to acquire spatial information about the unknown or partially known environment.

Process of Exploring An Unknown Environment

In operating in a two-dimensional environment, the mobile system must have at least the ability to identify and record each polygonal vertex encountered, its location, the location of obstacle-free paths, and the identity of other vertices that are visible from each vertex. Even if the vehicle could only access a coordinate reference position when it was at a polygon vertex, it could be able to carry out exploration. A mobile system having the abilities as sketched above would, indeed, have a capability for exploring its environment.

Since one polygon edge could obscure other polygon edges/vertices, the only way to determine what vertices are actually visible is to be at (visit) each vertex and use a sensor to determine in what direction some line-of-sight is obscured by an obstacle. To reduce the total distance travelled in exploring the environment, it is usually desirable to minimize the number of times that the vehicle has to revisit a given vertex. Exploration of each vertex will require that a minimum total of N scanning operations (one from each vertex) be conducted. No obstacle can change its position and once the environment has been scanned from a vertex the visibility of all of the other vertices visible from it will have been determined. If a vertex is revisited, a second exploratory scanning operation will not be required. Because each vertex must be visited at least once, there will be a minimum of N-1 path element traversals required to visit each vertex (Rao et al,, 1988).

Having no a priori knowledge about the environment, the exploration technique must include a criteria for termination of the exploration process. Since every vertex will be visible from at least one other vertex which can be visited, a automaton having a list showing no more unvisited vertices can conclude that the exploration process has terminated.

Navigating In Known Environments

The result of an exploration process would be an ability to recall and identify what vertices had been visited, the measured distance of paths traversed, and the distance between visible vertices. This result would be the attainment of a spatial description of the environment (eg., a map, knowledge base, etc.). If the positions of vertices were related to a common reference point, the information could be used to plan globally optimized (minimal distance) movement. If a universal point of reference were not available, the information could only be used to plan locally optimized movement.

Once a description of the environment is available, two added efficiencies can be brought to bear on the problem of navigating amidst obstacles:

1. no further sensor operations are required for purely exploration purposes
2. path planning can be carried out as a purely computational problem that utilizes a well developed path planning technique known as "find-path."

The exploration process is thus a precursor to the "find-path" problem. The exploration problem has no well developed theoretical underpinning as does motion planning when a priori information is available. In considering recurring exploration and navigation activities, it is useful to consider the following activities:

Scanning operation- once at a vertex, a scanning sensor is used to determine all of the other vertices visible from that vertex.

Path element traversal- the vehicle has two "move" options:

1) along a line-of-visibility from one vertex to another and

2) along the edge of a polygon

Path decision criteria- the approach for deciding which path element to follow next (eg., closest, left-most, edge, line-of-visibility, etc.)

ENVIRONMENT CHARACTERISTICS

In focusing on the exploration process, a reduction of the dimensionality of the environment to order two is appropriate. Obstacles consist of a finite number of random size/shape polygons random distributed throughout the environment. Due to the simplification that only polygons with planar faces are considered (no curved faces), a complete description of the environment would consist of a vertex identifier and its reference coordinates. There would be at least three vertices for each polygon, and there would be a total of N vertices in the environment.

Represention of Environment Knowledge

In order to record the spatial information about obstacles the environment, Visibility Graphs have been widely used to represent spatial relationships between entities that are depicted as nodes. Formally, a Visibility Graph (VG) of an environment which has a set of obstacles that have a vertex set (V) and an edge set (E) is defined as:

$$VG = (V, E)$$

where

(i) V is the set of all nodes of the obstacles

(ii) a line joining the vertices v_i and v_j forms an edge (v_i, v_j) in E if and only if it is an edge of an obstacle or it is not intersected by another obstacle.

In this spatial representation, edges E represent obstacle-free lines-of-visibility between vertices. Vertices not connected by an edge are obscure from each other. Relating distance to the edge would signify the distance between the two vertices. The environment is assumed to be of a bounded finite size in that the obstacles are all contained within the finite area of the environment.

AUTONOMOUS SYSTEM SIMULATION

Following the development of a theoretical framework for solving the problem of navigating amidst known obstacles (Lozano-Perez and Wesley, 1979 and Reif 1979) other approaches for planning collision-free paths for vehicles have been developed. Different approaches concerning how to best approach the problem of navigating amidst obstacles whose positions are unknown or partially explored have been presented (Chatila,1982; Chattergey,1985; Crowley,1985; Giralt et al, 1984; Iyengar et al, 1986 and others). Although many of those approaches are shown to be effective for their specific environments, a rigorous mathematical proof of their correctness or convergence has only been presented in a few cases (Oommen et al,1985; Rao et al, 1988).

Combined Exploration and Navigation

One technique that would allow combining the objectives of both goal seeking and exploration is to allow limited exploration of the environment while the automaton is attempting to attain a specified goal. A technique for doing this has been addressed, algorithmically, (Iyengar et al,, 1985).

Another technique toward the implementation of combined exploration and goal seeking was through the use of a rule-based system (McKendrick, 1989). There the environment was defined using Lisp data structures and motion through the environment was controlled by the OPS5 rules. Once a start position and desired goal positions had been specified, the exploration and path finding simulation began. The results of a scanning operation were kept in a knowledge base (OPS5 Working Memory) while decisions of what next course of action to take were exercised when knowledge elements satisfied rule specifications. Rather than having the primary objective of exploring the environment, the primary goal was to achieve the designated goal and only carry out limited exploration when close to an unexplored vertex. Partial knowledge about the environment was accumulated in Working Memory as new traversals through the environment were made. Figure 3 shows that vertex visibility knowledge increased, Percent of Environment Known, as each of four traversals toward a goal was completed. By the time the forth goal has been reached, more than 60% of the vertex locations have been learned (visited).

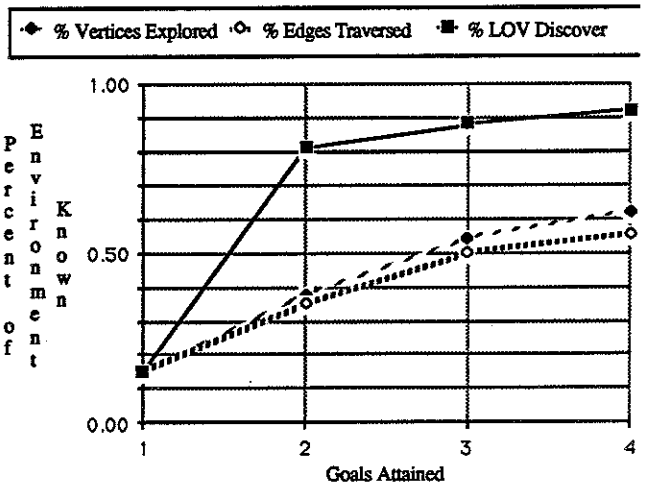


Figure 3. Accumulation of Knowledge of the Environment As Path Traversal Progresses

Incomplete or partial knowledge is valuable in that with no other information, it can serve as a limited basis for carrying out obstacle avoiding path element traversals. It is useful information up to the exploration point attained.

Pure Environment Exploration

Another way to look at exploration is to consider it as the primary task for the vehicle to accomplish. As

was pointed out above, assuming that complete exploration of an environment has taken place allows subsequent decisions of how to move most efficiently to be matters of computation alone.

Within the context of exploration, the question arises as to at what point the environment would be completely known. Most research has focused on achieving the goal and little research has considered the problem of complete exploration of the environment. In studying exploration, it would be beneficial to have a measure of exploration efficiency to compare different approaches. Initially, one could consider using mertics of the number of vertices visited/revisited, the least path length, the easiest to implement, least number of total path elements, least total path length, etc.

Exploration Algorithm Based on Least Distance

An exploration algorithm based choosing the path element having the least-distance to any currently visible vertices has been proposed (Rao et al, 1988). The currently visible vertices were placed in an "adjacency list." An adjacency list of other vertices v of VG was obtained by placing the vehicle at v and allowing it to scan the whole environment detecting what other vertices were visible from that position. They then placed the adjacency list of v in a Partial Visibility Graph PVG. PVG was augmented after each visit to a new vertex. They proved that the VG is corrected and they insured that all of the vertices of the obstacle were visited. The PVG was shown to converge to the VG which insured the completely mapped environment.

An example of the result of applying this algorithm is shown in Figure 4. Vertex 1 is the starting point. The 2 underlined vertices are the only ones which are revisited. A total of 18 path element traversals are performed.

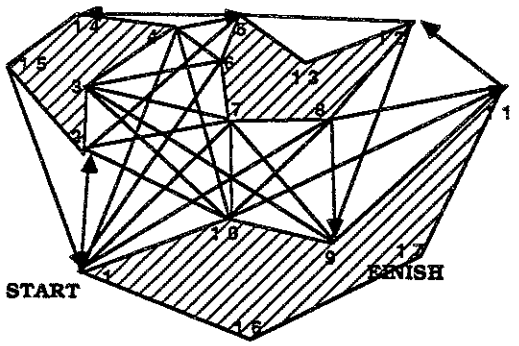


Figure 4. Exploration Resulting From Applying Shortest Distance Metric

Non-distance Path Selection Metric

Alternate criteria can be considered in choosing which path element to follow next. We conjecture that selection of the left-most (or right-most) unvisited vertex will also result in the fewest number of vertices being revisited.

Consider how this algorithm might be applied to explore an unknown environment shown in Figure 5. The automaton starts at vertex 1. From vertex 1, it performs a scanning operation. The result of the operation is the identification of unknown vertices and the relative positions of each. This information is placed into the automatons knowledge base (e.g., Barycentric Visibility Matrix (McKendrick, 1989)). This Matrix could contain the present vertex identification (vertex 1) and relative positions of the visible vertices. Since the actual names of the visible vertices are not yet known, they are given a temporary name. The relative position is associated with its temporary name.

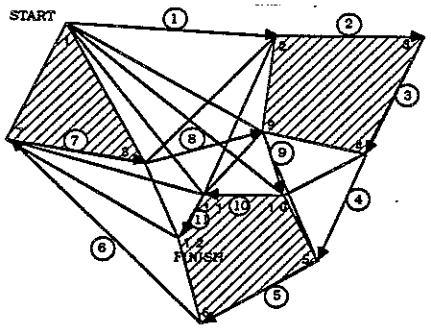


Figure 5. Result of Applying "Left-Most" Not-Visited Vertex Criteria

Following the scanning operation, the left-most vertex that has not yet been visited (vertex 2) is selected and the vehicle proceeds to that vertex. Once at that new vertex, that vertex is assigned a permanent name (ie, 2), it is marked as visited and another scanning operation is performed. The circled numbers denote path element traversals while the arrows indicate the direction of traversal. The procedure is repeated until there are no more vertices that have not been visited. Due to the fact that the BV Matrix contains all of the visibility information, there may be a vertex that cannot be reached unless the vehicle revisits one of the already visited vertices. In that case, the vehicle must revisit some vertex(s) on its way to the unvisited vertex. A scanning operation is not necessary at the revisited vertex .

When this path move criteria is applied to the environment which was explored by the shortest-next-move-distance, the 3 underlined vertices shown in Figure 6 were visited twice. There were 19 path element traversals. It is suggested that this criteria has value for some environments. It is interesting to note that the technique is exhaustive in that it is able to reach every vertex.

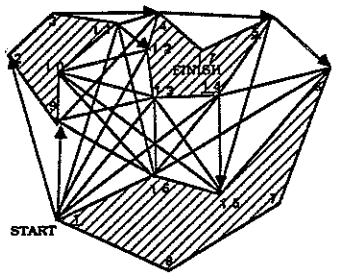


Figure 6. Application of Nearest Vertex Strategy to Explore Environment

PATH DECISION COMPLEXITY

There appears to be some similarity between the attempt to measure the decision complexity of exploring an unknown environment and the attempt to measure the "structural complexity" (psychological complexity) of a computer program. Within certain limitations, the Visibility Graph appears to be related to the Program Flow Graph.

The most popular program structural complexity metric, the McCabe Cyclomatic Number, was derived

from the notion of relating program complexity to the complexity of the flow through a directed graph that is used to represent program control flow. That approach looked at the structural complexity of the variety of basic paths that could be taken to reach a desired end-point or goal. McCabe extended the idea to the problem of computing the number of basic paths that when taken in combination would generate every possible path. That approach assumed that each node could be reached by the entry node and that from each node, the exit node could be reached.

The cyclomatic number V(G) of a graph G, with n vertices, e edges, and p connected components can be computed using the relationship:

V(G): e - n + 2p

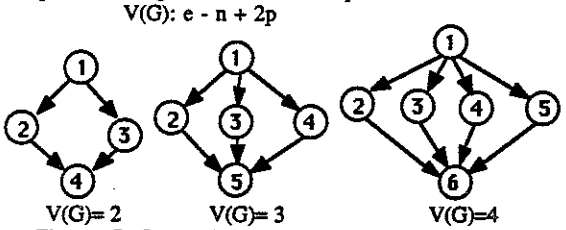


Figure 7. Increasing the Number of Paths Toward Goal

This approach has provided a relatively simple and straight-forward method for measuring path decision complexity.

Entropy Based Complexity Measure

An attempt has been made to apply an entropy measure to the problem the path decision uncertainty that may be present when one is about to select from a variety of potential paths through a directed graph (Davis and LeBlanc, 1988). Uncertainty arises from the options presented by predicates that allow one to choose alternative paths to modules in a computer program.

In the theory as set forth by C. Shannon (Shannon, 1949), the entropy of an experiment E can be calculated as a summation of the probabilities of achieving the various "classes" of the states of the experiment. Here, P is a function that assigns real probabilities to a finite collection of events: A_1, A_2, ...A_n. The entropy is given by

$$H = -\sum P(A_i) \log P(A_i)$$

Davis and LeBlanc refer to program "chunks," that are comparable to graph "nodes." Their development of the use of the entropy measure focused on the context of graph nodes. The nodes of a graph representing the program control (decision) structure were classified based on their in-degree and out-degree. An "ordering" of entropy was considered such that in "first order" entropy, nodes having the same in-degree and out-degree were considered to be equivalent. "Second order" entropy considered nodes to be equivalent only if they had the same number and type of neighbors at one arc distance. For example, the first order entropy for the graph in Figure 8 is H=2.13.

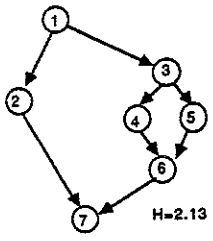


Figure 8. Graph With Davis/LeBlanc Entropy

Some simulations have been undertaken to develop entropy values of several different graph structures that represent simplistic explored environments. At this point, a consistent formulation has not yet been achieved. The merit in having an entropy measure to describe an already explored environment would be in its ability to provide a probabilistic exploration bound to contrast with the topological exploration heuristic discussed previously.

Reference List:

Brooks, R.A.1983. "Solving the Find-Path Problem by a Good Representation of Free-Space." *IEEE Transactions of Systems, Man and Cybernetics.* SMC-13 (Mar/Apr): 190-197.

Chatila, R.1982. "Path Planning and Environment Learning in a Mobile Robot System." In *Proceedings of the European Conference on Artificial Intelligence.* 59-60.

Chatila, R. and J.P. Laumond.1985. "Position Referencing and Consistent World Modeling for Mobile Robots." In *Proceedings of the 2nd IEEE Conference on Robotics and Automation* (Mar 10-12). 134-139.

Crowley, J.L. 1984. "Navigation for an Intelligent Mobile Robot." In *Proceedings of the 1st IEEE Conference on the Applications of Artificial Intelligence* (Denver,CO.). IEEE, Piscataway, N.J. 79-84.

Davis, J. S. and LeBlanc, R. J. 1988. "A Study of the Applicability of the Complexity Measure." *IEEE Transactions on Software Engineering* 14, no. 9 (Sep): 1366-1372.

Iyengar, S.S.; C.C. Jorgensen; S.V.N Rao; and C.R. Weisben. 1986. "Robot Navigation Algorithm Using Learned Spatial Graphs." *Robotica* 4 (Jan.): 93-103.

Lozano-Perez, T. 1983. "Spatial Planning: A Configuration Space Approach." *IEEE Trans-actions on Computers* 32, no.2 (Feb.): 108-121.

Lozano-Perez, T. and M.A. Wesley .1979. "Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles." *Communications of the ACM* 22, no. 10 (Oct.): 560-572.

McCabe, T. J. 1976. "A Complexity Measure," IEEE Transactions on Software Engineering, SE-2, (Dec):308-320.

McKendrick, J.D. 1989. "Simulation of Autonomous Knowledge-Based Navigation in Unknown Two-Dimensional Environment with Polygonal Obstacles," MS Thesis, Florida Atlantic University: Boca Raton, FL.

Oommen, B.J.; S.S. Iyengar; N,S.V. Rao; and R.L. Kashyap. 1987. "Robot Navigation in Unknown Terrains Using Learned Visibility Graphs. Part 1: The Disjoint Convex Obstacle Case. *IEEE Journal of Robotics and Automation* RA-3, no. 6 (Dec.): 672-681.

Rao, N,S.V., S.S. Iyengar, B.J. Oommen, B.J. and R.L. Kashyap. 1988. "On Terrain Model Acquisition by a Point Robot Amidst Polyhedral Obstacles." *IEEE Journal of Robotics and Automation* RA-4, no. 4 (Aug.): 450-455.

Shannon, C.E. and W. Weaver. 1949. The Mathematical Theory of Communication. Univ. of Illinois Press: Urbana, IL.

Wilson, R. J. 1984. "Analysis Situs," *Graph Theory With Applications to Algorithms and Computer Science.* John Wiley and Sons, 789-800.

Key Words: Exploration, Autonomous System Navigation, Path Desision Making

THE USE OF BRAIN ACTIVITY FOR TRIGGERING INFORMATIONAL DISPLAYS

Bruce R. Dunn
Laboratory for Studies in Neurocognition
Department of Psychology
The University of West Florida
11000 University Parkway
Pensacola, FL 32514

Kenneth M. Ford
Department of Computer Science
The University of West Florida
11000 University Parkway
Pensacola, FL 32514

INTRODUCTION

The purpose of this paper is to report on a continuing research program aimed at the development and unification of the necessary foundations for a successful approach to the design and construction of a brain-triggered computer system. This work may serve as the basis for advances in man-machine interaction (e.g., cognitively adaptive interfaces), as well as computer-aided learning systems in which the system adapts its manner of pedagogy to the cognitive style of the student.

Research has shown strong links between specific aspects of the brain's electrical activity (EEG) and certain cognitive states (Doyle, Ornstein, & Galin, 1974; Dunn, 1985a). An important implication of this research is that electrophysiologic activity can be used as a basis to monitor a person's attention or processing level via a computer. That is, by detecting changes in a person's electrophysiologic activity an associated real-time computer system can assess momentary shifts in attentiveness. Furthermore, a shifting or waning level of attentiveness can serve as a "trigger" for an associated expert system to make controlled changes in the way information is displayed to subjects during various performance (e.g., flight simulation) or acquisition tasks (e.g., text comprehension) to increase the subject's performance. Several pioneering studies suggest that this can be accomplished (Bauer & Nirnberger, 1980, 1981; Cacioppo & Sandman, 1978; Guttmann & Bauer, 1984; Sandman, McCaane, Kaiser, & Diamond, 1977).

Bauer and Nirnberger (1980), for example, programmed a computer to calculate the means for 2-second intervals of ongoing EEG while subjects were engaged in a paired-associate learning task. When the mean difference between any two of these 2-second EEG epochs was equal to or greater than -25μv or +25μv, the computer presented a word-pair to the subject on a CRT. The data showed that learning of the word-pairs was facilitated during negative shifts (-25μv) rather than positive shifts (+25μv). Hinger, 1976 (reported in Guttmann & Bauer, 1984) demonstrated that the ability to learn nonsense syllables is reduced when subjects are producing high-levels of alpha (8-13 Hz) activity. Thus it appears that at least two EEG measures (i.e., alpha activity and EEG potential shifts) have been identified that could be used to monitor a person's state of attention to a given task.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

Although very important because of its implication for the military (e.g., on-line computer monitoring and control of fighter pilot's performance) and the industrial and educational communities (e.g., on-line computer control of a person's acquisition of complex material) this type of research suffers from two major limitations -- either of which could prove to be a critical flaw. First, all previous contingent display research has ignored individual differences in learning and brain organization. For example, Sandra McGlone (1980) and Jere Levy (1980) argue that male and female brains may have structural differences and that these differences may be related to differential cognitive performance. Furthermore, Dunn and his colleagues (Dunn, 1985a, 1985b; Dunn, Gould & Singer, 1981; Reddix & Dunn, 1984; Rust, 1982) have identified two learning styles, each of which exhibit differential alpha activity patterns and each of which employ learning strategies that are different from the other. This latter research is important because it has shown that alpha measurements appear to measure diametrically opposite attentional and cognitive processes when these two cognitive styles are compared. The implication of this research is that alpha is *not* a universal predictor of performance as suggested by the results of Hinger (1976) reported earlier, which showed that the learning of nonsense syllables is reduced when subjects are producing high-levels of alpha. Our research (described below) shows that although the majority of the population does learn better when they are producing less alpha, a large minority actually shows better learning when they are producing *greater* than their normal alpha activity. Thus, for brain-triggered informational displays to function effectively, individual differences need to be taken into account.

The second reason that current brain-triggered systems will have limited utility is because very few brain sites are typically used to trigger the computer driven display changes. Consequently, there is a high-probability that the brain site which is the most correlated with a given cognitive process will *not* be identified and used.

PROBLEM OF LIMITED RECORDING SITES

The obvious solution to the problem of limited recording sites is to increase the number of EEG channels collected. However, the interpretation of *patterns* of EEG activity can be greatly enhanced by application of a computer-based technology -- *topographic mapping*.

Topographic mapping of the brain's electrical activity is a relatively new methodology which has been developed to reduce, summarize, and display various features of electrical brain activity from a variety of scalp locations (typically ten or more). The product of this data manipulation is a topographic map of the electrical activity across the entire cerebral cortex of the brain (Duffy, Burchfiel, & Lombroso, 1979; Torello & Duffy, 1985). Since EEG can be collected during the execution of a given performance or cognitive task, the topographic image and related EEG data can be used to identify which of the brain sites are the most correlated with the performance of that task for a *particular* individual. Thus, topographic imaging allows a domain expert to see *individual patterns* of EEG activity which could typically elude him or her when using more traditional measures.

Figures 1 and 2 will illustrate our point. The three topographic maps in Figure 1 show the distribution of 10 Hz activity (the middle of the alpha range as determined by a FFT (fast Fourier transform) of the EEG records) across the cerebral cortex of the brain when all three individuals were reading the same text segment. Fifteen electrode sites were used.

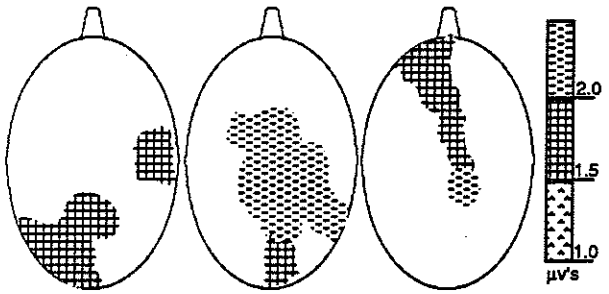


Figure 1: Topographic maps for three individuals reading the same text segments.

The topographic map was generated by calculating the values for the pixels between electrodes by using a linear interpolation of the amplitude values of the three electrodes nearest that given pixel (the three nearest neighbors to that pixel). The amplitude values at 10 Hz are shown in microvolts to the right of the figures. Thus, it can be seen that EEG activity varies *across* individuals when they perform the same task.

Figure 2 shows the 10 Hz activity of the same individual, while reading different text segments. As can be seen below, the EEG patterns generated *within* the same individual vary as a function of the stimulus they are processing, albeit not as much as across individuals (Figure 1).

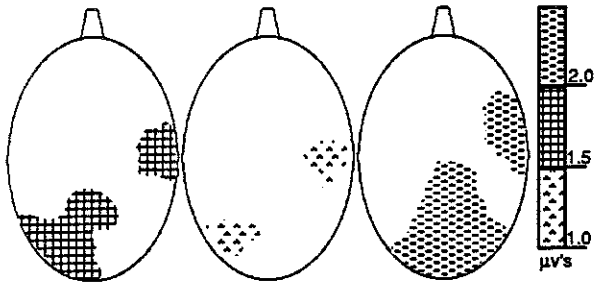


Figure 2: Topographic maps for the same individual reading three different text segments.

These rather obvious differences across individuals (and within the same individual) strongly support our notion that if accurate brain-triggered informational displays are to become a reality -- it is critical to determine a particular individual's pattern of EEG activity for a particular task. The graphical representation of brain activity provided by topographical imaging enables a domain expert to more readily (in comparison with a profusion of numeric data) interpret EEG data. As discussed below, an expert system could readily interpret EEG patterns for the determination of which electrodes to monitor (for a given person), the individual's major learning style, attention to a specific task, etc.

INDIVIDUAL DIFFERENCE PROBLEM

The study described below *underscores* the necessity of determining individual differences in brain activity as the basis for brain triggered changes in informational displays. More importantly, it will allow us to describe a current direction of our research program. Unfortunately, the study is inherently weak because only two brain sites were used to determine these differences. As described above, we believe more accuracy can be gained by using multiple recording sites, coupled with an expert system based on *topographic brain imaging*.

Much of our research is based on *bimodal theory*. According to the theory, the human brain has at least two qualitatively different modes of thought: the *analytic mode* (i.e., a logical, linear, and sequential processing system) and the *holistic mode* (i.e., a simultaneous, parallel, or gestalt processing system). Several researchers (Deikman, 1976; Dunn, 1985; Hymes, et al., 1977) point out that although every person utilizes these two modes, some people typically use one more than the other. Hence, an extreme analytical person would tend to process complex information (like text) in a logical (or hierarchical) sequential manner; whereas, a holistic would tend to employ more paralogical or gestalt processing. Research has indicated that these two modes can be determined by EEG patterns with analytics producing less alpha activity than holistics (e.g., Deikman, 1971; Dunn, et al., 1981; Reddix & Dunn, 1984).

Figure 3 shows the differences in cortical distributions of 10 Hz activity (in the middle of the 8-12 Hz alpha range) between two analytics (left) and two holistics (right) during an eyes-open baseline condition.

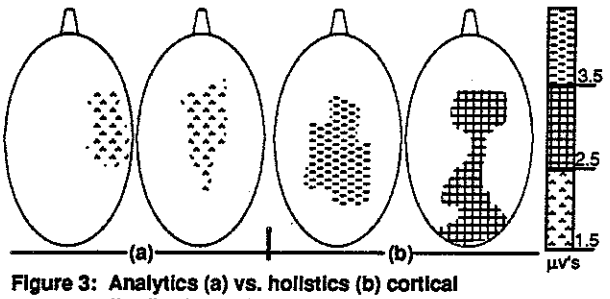


Figure 3: Analytics (a) vs. holistics (b) cortical distributions of 10 Hz. activity.

The study described below was conducted prior to the development of our current brain imaging system. However, it is important (in this context) because it revealed that analytics and holistics produce differential alpha activity when reading *specific* text items; and further, that their recall of that item was a function of their alpha level at the time of reading the information.

DUNN, RANEY, AND RUST STUDY

EEG and recall data were collected from twenty, right-handed 9th grade boys using expository and descriptive texts. Two parietal/temporal sites were used for EEG recording. The left hemisphere electrode was placed over Wernicke's area, which is supposedly involved in comprehension (Geschwind, 1979). An identical placement was made on the right hemisphere. A two-minute baseline was recorded at the end of a ten-minute initial rest period. After the baseline was taken, subjects were given two texts to read and were required to write down all they could remember immediately after the presentation of each passage. EEG recordings were taken during both the reading and recall periods. For the sake of brevity, only the data collected for the most logically organized passage is reported here.

All texts were analyzed and all recall protocols were scored using Bonnie Meyer's (1975) content-structure analysis system. Although quite complicated, this text analysis system allows the researcher to determine the important semantic and logical aspects of a text by organizing text items in a hierarchy from highly important to less logically and semantically important text items. Analysis of the passage using Meyer's system yielded a hierarchy containing five levels of importance. This permitted us to present segments of the passage on separate slides with some of the information being critically important (level-1 or "gist" information) and other information having relatively less importance for comprehension of the passage (level-5 contains the least important material). Consequently, EEG could be recorded while people were encoding or processing text of known semantic importance. FFTs were then conducted on all the EEG data.

Upon completion of the study, the twenty subjects were dichotomized into analytic and holistic groups, based on their summed bilateral EEG baseline alpha activity. As per previous research (e.g., Dunn et al., 1981) subjects' total alpha scores were rank ordered, and those who fell above the median alpha production for the group were identified as holistics and those who fell below the median were termed analytics.

Conditional probabilities were computed between the subjects' alpha power scores, generated at encoding, and their proportional recall scores as a function of their cognitive style. Figure 4 illustrates the procedure we used.

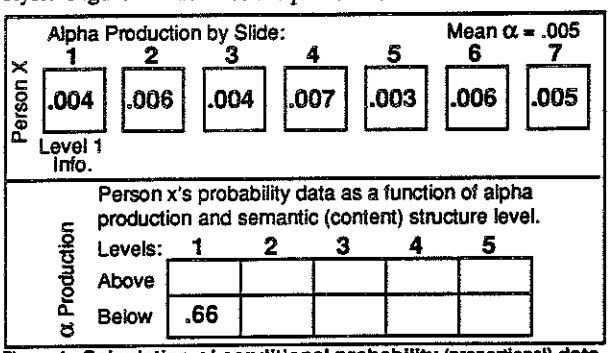


Figure 4: Calculation of conditional probability (proportional) data.

Specifically, the total alpha produced by a given subject while reading the information contained on a particular slide was computed. Then the alpha produced during the reading of the slides was summed and averaged. For any given slide, it was determined whether the alpha produced during the reading of that slide was above or below that particular student's mean alpha (averaged across all reading slides). In this example, the subject's alpha production for slide 1 (.004) was below his mean alpha (.005) for the reading series. Therefore the proportion of information he recalled from slide 1 (.66) was recorded in the data matrix (bottom half of Figure 4). His other data were analyzed for the remaining slides.

These conditional proportional recall data produced by all subjects were analyzed using a 2 (type of processor: analytic vs. holistic) by 2 (type of alpha score: above or below mean alpha score) by 5 (levels of importance: Levels 1-5) analysis of variance.

ANALYSES AND RESULTS

The important 3-way, type of processor by type of alpha score by levels of importance interaction reached statistical significance ($p < .04$). This interaction is shown in Figure 5 as two, 2-way interaction components for ease of explanation.

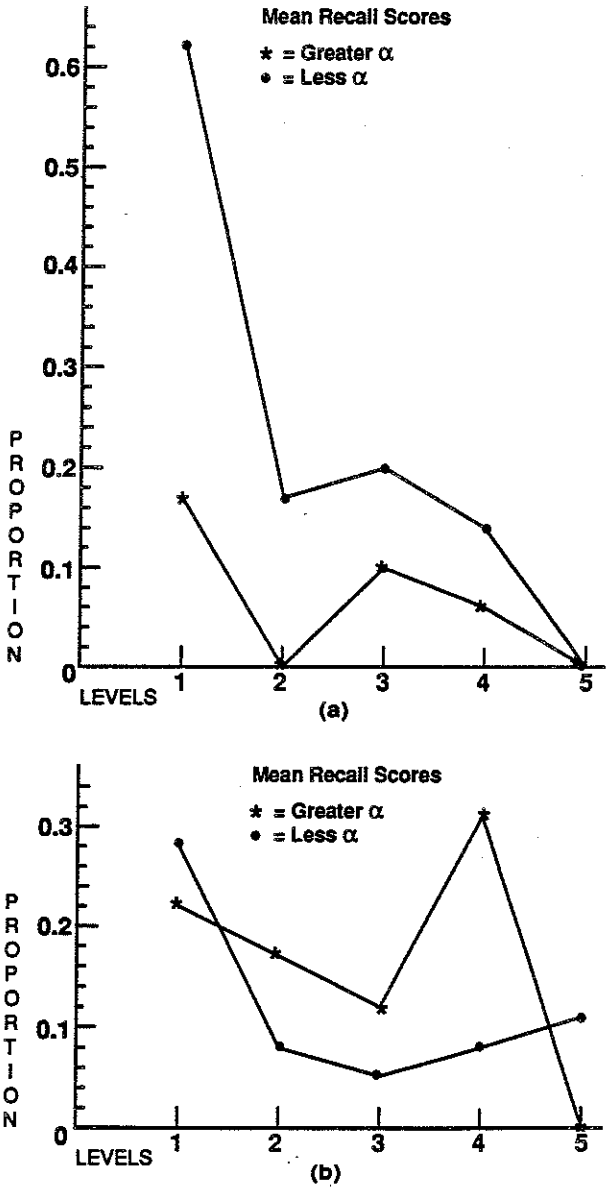


Figure 5: Analytic (a) vs. holistic (b) recall as function of α-level.

From Figure 5 it can be seen that analytics' (i.e., natural low-alpha producers) greatest recall occurred when they were producing less than their mean alpha while they were reading the material. When they (analytics) were generating more than their mean alpha, their ability to later recall what they were reading was diminished. In contrast, it can be seen that when holistics (i.e., natural high-alpha producers) are asked to read complex text, there is a greater probability that they will recall that text later if they were generating *greater* than their mean alpha during the act of reading, particularly at some subordinate levels (e.g., level 4). However, if they were producing *less* than their mean alpha at the moment of reading or acquisition they tended not to recall that subordinate information later.

FUTURE DIRECTIONS

In our research program we are using topographic brain imaging in order to determine which brain sites are the most active for a *given* individual during the execution of a specific cognitive or performance task. Once identified, those brain sites (unique to that individual) will then be used as the basis for the construction of a real-time expert system that triggers appropriate changes in the informational displays presented to the subject. In order to demonstrate how a brain-triggering system could be used, we will use a task similar to that reported above (i.e., reading segments of expository text).

For an individual subject, EEG would be recorded both during a baseline and a reading pretest. During the reading pretest, text segments (of various levels of importance), would be displayed on the computer screen. Based on the pretest baseline, the computer would then calculate the person's level of alpha (or other frequencies) from computer selected cortical sites and determine whether that person was an analytic or a holistic. The reading pretest data would then be analyzed to identify the most appropriate brain sites for that particular individual.

For example, if the EEG from the previously chosen monitoring sites indicates that the person is probably not attending to the display or may not be grasping the information currently being displayed, the computer would send "remedial" information to the display screen – perhaps in a different format. In this way, an adaptive feedback loop may be established between man and machine -- the computer display is controlled (to some degree) by the human's cognitive activity and yet the informational display (both content and format) is itself a major influence on that activity.

In summary, we believe that when an individual is required to perform a complex task (e.g., fly a fighter plane) or learn highly-complex information (e.g., school-related learning), greater performance will occur when the human and machine are working in a more synergistic fashion than is now the norm. This research is a step in that direction. Specifically, we are in the process of exploring the feasibility of computer-driven "brain-triggered" informational displays with both educational and military implications.

REFERENCES

Bauer, H. and G. Nirnberger. 1981. Concept identification as a function of preceding negative or positive spontaneous shifts in slow brain potentials. *Psychophysiology*, 18, 466-469.

Bauer, H. and G. Nirnberger. 1980. Paired associate learning with feedback of DC potential shifts of the cerebral cortex. *Archives of Psychologie*, 132, 237-239.

Cacioppo, J.T. and C.A. Sandman. 1978. Physiological differentiation of sensory and cognitive tasks as a function of warning, processing demands, and reported unpleasantness. *Biological Psychology*, 6, 181-192.

Deikman, A.J. 1976. Bimodal consciousness and the mystic experience. In *Symposium on Consciousness*, P.R. Lee, R.E. Ornstein, D. Galin, A.J. Deikman, and C.T. Tart, eds. The Viking Press, New York.

Deikman, A.J. 1971. Bimodal consciousness. *Archives of General Psychiatry*, 25, 481-489.

Doyle, J.C., R. Ornstein, and D. Galin. 1974. Lateral specialization of cognitive mode II: EEG frequency analysis. *Psychophysiology*, 11, 567-578.

Duffy, F.H., J.L. Burchfiel, and C.T. Lombroso. 1979. Brain electrical activity mapping BEAM: A method for extending the clinical utility of EEG and evoked potential data. *Annals of Neurology*, 5, 309-321.

Dunn, B.R. 1985a. Bimodal processing and memory from text. In *Psychophysiological Aspects of Reading and Learning*, V. Rentel, S. Corson, and B.R. Dunn, eds. Gordon & Breach Science Publishers, New York.

Dunn, B.R. 1985b. Brain organization and cognitive style: Facts, Fancies, and Possibilities. Paper presented at the 11th Western Symposium on Learning, Western Washington University, Bellingham, WA.

Dunn, B.R., J.E. Gould, and M. Singer. 1981. "Cognitive style differences in expository prose recall." Tech. Report 210. The Center for the Study of Reading, Univ. of Ill.

Geschwind, N. 1979. Specialization of the human brain. *Scientific American*, 3, 158-168.

Guttmann, G. and H. Bauer. 1984. The brain-trigger design. In *Brain and Information: Event-Related Potentials*, R. Karrer, J. Cohen, and P. Tueting, eds. New York Academy of Sciences, New York.

Hinger, A. 1976. Elektroenzephalogramm und Lernleistung. Unpublished Doctoral Diss., Wein. Cited in Guttmann, G. and H. Bauer. 1984. The brain-trigger design. In *Brain and Information: Event-Related Potentials*, R. Karrer, J. Cohen, and P. Tueting, eds. New York Academy of Sciences, New York.

Hymes, D., B.R. Dunn, J.E. Gould, and W. Harris. 1977. Effects of mode of conscious processing on recall and clustering. Paper presented at the annual meeting of the Southeastern Psychological Association, Hollywood, FL.

Levy, J. 1980. Cerebral asymmetry and the psychology of man. In *The Brain and Psychology*, M. Wittrock, ed. Academic Press, New York.

McGlone, S. 1980. Sex differences in human brain asymmetry: A critical study. *Behavioral and Brain Sciences*, 3, 215-263.

Meyer, B.J.F. 1975. *The organization of prose and its effects on memory*. North-Holland Pub. Co., Amsterdam.

Reddix, M.D. and B.R Dunn. 1984. "EEG alpha production correlates of cognitive style differences and recall of metaphor from poetry." Tech. Report 324. Urbana, Illinois: The Center for the Study of Reading, Univ. of Ill. Abstracted in *Resources in Education*.

Rust, D.T. 1982. EEG alpha responses related to semantic recall in two groups of children. Unpublished Doctoral Dissertation, Walden University.

Sandman, C.A., T.R. McCanne, D.N. Kaiser, and B. Diamond. 1977. Heart rate and cardiac phase influences on visual perception. *Journal of Comparative and Physiological Psychology*, 91, 189-193.

Torello, M.W. and F.H. Duffy. 1985. Using brain electrical activity mapping to diagnose learning disabilities. In *Learning and the Brain: Vol. 24 Theory into Practice*, M.L. Languis, ed. Columbus, OH: The Ohio State Univ., 95-99.

A STEP TOWARD AN INTELLIGENT SOFTWARE TRANSPORTATION SYSTEM

Fred Y. Wu
Department of Electrical and Computer Engineering
University of Miami
P.O. Box 248294, Coral Gables, Florida 33124

ABSTRACT

Knowledge of program conversion is very hard to be represented by traditional representations, because syntactic and non-syntactic knowledge involved can not be completely formulated in rules or is even not vocal expressible. A multi-based representation and a concept learning algorithm are proposed to accept composite expressions in describing program conversion techniques, and to construct a self-organized concept hierarchy. The multi-based representation combining advantages of logical, semantic network, procedural, and frame-based representations is designed to absorb a wide variety of epistemological primitives. The concept learning algorithm designed to conduct a multi-medium search in relating concepts also provides a solution to the general problem of frame-based representations in differentiating assertional and structural properties.

INTRODUCTION

Program conversion is a fundamental task of reusing software. Related research can be classified by the degree of automation into two categories: the automatic and the interactive approach. Research on automatic convertors has focused on strategies of automated transformations, emphasizing the needs to eliminate clerical mistakes by reducing human interventions. Successful results have been announced for language-to-language translations under specific restrictions (Boyle and Muralidharan 1984). Generally speaking, there are three levels of conversion involved in an automatic translation process, the syntactic level, the semantic level, and the system level. At the syntactic level, differences of syntax can be usually taken care of by a line-by-line translation in the automatic conversion process, but the same technique is usually not applicable to problems at the semantic level; an overall understanding of semantic characteristics of both languages is also required to handle problems due to structural differences between languages. A missing feature in the target language, as a direct result of semantic differences, could be compensated by a subroutine simulating the particular function using the target language. This subroutine is saved in a library and is automatically invoked to insert compatible codes to the target program whenever the missing function is identified in a conversion process. The automatic conversion approach seems to work fine for most of problems at the syntactic and the semantic level, however, conversion at the system level is almost impossible to be fully automated. For example, if a source program contains optimized codes produced by using machine or system dependent tricks, manual work is

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

required to rewrite the importable part, for the required conversion usually can not be described algorithmically. Therefore, a fully automatic conversion is actually impossible without restricting the translation at a very limited level.

Another approach centered around the development of partial automatic convertors, depicted as interactive program development tools. By using such a software transportation system, users are required to make decisions in each crucial point in a conversion process to select the next translation procedure, but are thus allowed to direct the conversion process by adding environmental specifications or considerations for specific goals. More general conversion problems can be solved at the expense of less security in using the system, because the increased flexibility is due to the fact that the system itself has no strategies programed, and the whole conversion process relies on human decisions at each stage, suffering from the problem of inevitable human mistakes.

Besides the research classified by automation degree, some software transportation research advocates the development of very high level specification languages (Wile 1983; Leong 1981) to represent translation schemes. This approach did improve the interactive converter methodology, but the semantic gap between the special language and the language familiar to human experts raised another problem.

We think it may not be an appropriate idea to prohibit human interventions since human decisions are necessary for a general conversion task, but we should not completely rely on manual work in making all the decisions either, for human mistakes may frequently take place in such a complex process. We believe it is a better approach to guide human manipulations, instead of reducing or preventing them, by adopting AI techniques in the program conversion process.

By developing an intelligent convertor, sufficient information and suggestions are generated at each decision point to help the human expert, in the meanwhile, the whole conversion process is monitored by a mechanic reasoning module which prevents any possible inconsistent decisions. The highest control authority is still reserved for the user, but one can conduct the whole process without worrying about human mistakes. Semantic problems of using special languages can be solved by developing a user-friendly interface supported by a special internal knowledge representation. Instead of using a specification language, users can easily transfer program conversion knowledge by using description tools provided by the intelligent interface.

METHODOLOGY

Knowledge of program conversion is very hard to be represented by traditional representations, because syntactic and non-syntactic knowledge involved can not be completely formulated in rules or is even not vocal expressible. A multi-based representation and a concept learning algorithm are proposed to accept necessary composite expressions in describing program conversion techniques, and to construct a self-organized concept hierarchy.

Knowledge Representation

To completely describe a given concept, four knowledge attributes, each standing for a particular point of view, have been identified as follows:

- Symbolic attribute: This attribute provides information of entities mentioned in a concept, emphasizing atomic elements involved.
- Logical attribute: This attribute describes the logical relationship between entities, emphasizing the implied cause-effect factors.

- Hierarchical attribute: This attribute describes the structural relationship among entities, providing clues for property inheritance.

- Procedural attribute: This attribute describes a routine work by defined steps, emphasizing the sequence of actions to be taken.

A multi-based representation is proposed to embrace all the four knowledge attributes described above. The new representation is motivated by the frame-based representation, but with slots classified and specified for each of the knowledge attributes. For example, a concept could be defined as:

CONCEPT 1:

1. AB is a subset of U.

2. F is a subset of AB.

3. If G then X

4. If F then Do Routine 1

The knowledge attributes involved in this concept are:

- Symbolic attribute: A, B, F, G, X, U

- Logical attribute: 3, 4

- Hierarchical attribute: 1, 2

- Procedural attribute: 4

Frames are no longer connected by the IS-A assertions commonly used in general frame-based representations, but are automatically organized and conceptually related to each other through symbolic and hierarchical attributes. Human experts may update the description of a particular concept and indirectly change its relationship with other concepts. Side effects could be immediately reflected to the concept hierarchy through property inheritance. Instead of directly defining the relationship between concepts by IS-A assertions, users are allowed to define the semantic relationship only between symbolic attributes through the specified hierarchical attribute, and then the relationship between concepts is automatically adjusted by the updated concept hierarchy. This approach solves the general interpreting problems of frame-based representations in differentiating assertional and structural properties (Brachman, Fikes, and Levesque, 1983).

Knowledge Elicitation

After a new concept has been inserted according to the pedagogical habit of the human expert, to automate the process of differentiating knowledge attributes is very difficult; therefore, it is a better approach to clarify different knowledge attributes (Wu 1988a, 1988b) at the time the concept being acquired. A knowledge acquisition module can be developed based on the internal multi-based representation to generate questions in different points of view according to the defined knowledge attributes, and also allow the human expert to describe a particular skill in a natural and comprehensive way without learning another specification language.

Knowledge Synthesis

Acquired knowledge is stored as a set of concepts represented in frames and structurally organized through the information provided by the symbolic and hierarchical attributes of each frame. An inference strategy is required to efficiently synthesize relevant concepts to solve a given problem or to identify new or inconsistent knowledge for updating the knowledge base. The guidelines for developing such an inference engine based on the proposed knowledge representation are listed as follows:

- Step 1: Use the extracted symbolic attribute of input as an identifier to locate all relevant concepts organized in the concept hierarchy according to the following criteria.

  1. Select concepts with a matched symbolic attribute.
  2. Select concepts with a symbolic attribute subsuming the identifier.
  3. If 1 and 2 fail, select concepts with a symbolic attribute which shares a common subset with the identifier.

- Step 2: Select useful logic rules and procedures from the selected concepts in step 1 according to the following criteria.

  1. Select rules and procedures with a matched symbolic attribute.
  2. Select rules and procedures with a symbolic attribute subsuming the identifier.
  3. If 1 and 2 fail, select rules and procedures with a symbolic attribute which shares a common subset with the identifier.
  4. If 3 fails, select all rules and procedures.

- Step 3: Activate rules or procedures according to the symbolic identifiers.

EXAMPLES

The concept hierarchy is defined and shown in Figure 1.

1. Concept 1: AB $\subseteq$ U, F $\subseteq$ AB, G $\Rightarrow$ X, F $\Rightarrow$ routine 1.
2. Concept 2: H $\subseteq$ CD, CD $\subseteq$ U, U $\Rightarrow$ routine 2.
3. Concept 3: CDE $\Rightarrow$ Y.

Example 1: In solving a certain problem, the entity AB is inferred.

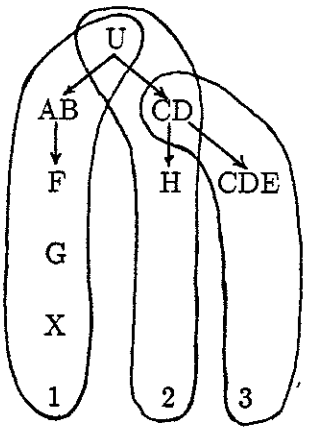


Figure 1: Concept hierarchy

Concept 1 will be selected because its symbolic attribute is matched with AB, and concept 2 will also be selected through hierarchy climbing. (AB $\subseteq$ U, U is in concept 2.) No solution can be found by using concept 1, which only tells that AB is subsumed by U, but the solution is implied in concept 2, which asserts that U triggers routine 2; therefore AB will trigger routine 2 through property inheritance. (AB $\subseteq$ U, U $\Rightarrow$ routine 2, therefore AB $\Rightarrow$ routine 2.)

Example 2: In solving a certain problem, the entity A is inferred.

The failure of symbolic matching indicates that this problem can not be solved directly from learned concepts, and additional information is required. AB is first found as the subset of the identifier A, and then U is found as the superset of AB. The inference strategy suggests to select all concepts covering U because U and A are both the superset of AB, and hypotheses could be generated through investigating U based on analogical reasoning techniques. Questions or suggestions will thus be generated according to the possible solutions deducted from those chosen concepts.

CONCLUSION

In summary, the proposed methodology can be described in two parts. Externally, the user interface of the intelligent convertor contains a knowledge acquisition module for

extracting human expertise in a multi-media fashion instead of requiring users to learn another specification language. In addition to the general forms such as Horn clauses or context-free grammars, descriptions by examples or limited English sentences are also acceptable to the system. With such abundant description tools, human experts can define translation schemes in a comprehensive manner, and to understand naturally how to input further information under the guidance of the system.

It may need both syntactic and non-syntactic knowledge to describe a particular conversion step. Syntactic knowledge could be encoded as procedures or logic-based Horn clauses, while non-syntactic knowledge could be described by examples or graphs. Internally, a multi-based representation is used to catch such a composite description by grouping different descriptions of a concept in slots of a frame by which the overall understanding of a specific conversion step is represented. Syntactic knowledge such as syntax differences between languages can be saved in slots of symbolic and logical attributes; non-syntactic knowledge such as the relation between global and local variables can be saved in slots of the hierarchical attribute; necessary manual coding processes can be saved in the slots of the procedural attribute. A knowledge acquisition module keeps updating those frames by monitoring the human expert's each decision. Human experts are responsible to give explanations for detected inconsistent decisions, and new knowledge obtained will be put into appropriate slots of a frame. The intelligent convertor actually works together with the human expert, assisting him throughout the conversion process by generating suggestions or activating automatic translation procedures, in the meantime, learning new strategies to improve its knowledge base.

REFERENCES

Boyle, J.M. and M.N. Muralidharan. 1984. "Program Reusability through Program Transformation." *IEEE Transactions on Software Engineering* SE-10, no. 5 (Sept.): 574-588.

Brachman, R.; R. Fike; and H. Levesque. 1983. "KRYPTON: A Functional Approach to Knowledge Representation." *Computer* 16, no. 10 (Oct.): 67-74.

Leong, S. 1987. "The Design and Implementation of a Converter Writing System," In *Proceedings of the IEEE Miami Technicon '87* (Miami, Fl, Oct. 28-30). IEEE, New York, NY, 313-316.

Wile, D.S. 1983. "Program Developments: Formal Explanations of Implementations." *Communications of the ACM* 26, (Nov.): 902-911.

Wu, F.Y. 1988a. "A Virtual Operator Technique for Enhancement of Computer-to-Computer Interactivity." Ph.D. dissertation, Iowa State University, 1988.

Wu, F.Y. 1988b. "Analogical Knowledge Transformation for Generating Computer Commands." In *Proceedings of the 1988 International Symposium on Mini and Microcomputers* (Miami Beach, Fl, Dec. 14-16). Acta press, Anaheim, CA, 112-115.

Stack-Based Processor Architecture for Real-Time Implementation of Declarative Programming Languages.

Andrew Kornecki, Embry Riddle Aeronautical University
Daytona Beach, Florida

Chris W. Malinowski, Harris Semiconductor
Melbourne, Florida

1. ABSTRACT

As evidenced by a number of papers published within the last five years in research and industrial publications, the main bottleneck of AI languages is associated with a poor match between the traditional Von Neumann architectures of CISC machines (onto which predominantly the AI languages have been ported), and the semantics of the languages themselves. A number of researchers have pointed out that multiple-stack based architectures, which specifically enhance the execution of threaded languages (such as Forth) hold the greatest promise for significant performance increase of declarative languages. Such improvement, in turn, is the key to successful implementation of real-time expert systems. The first experimental confirmation of this notion was reported by Paloski and Odette [Pa86], [Od88] where a real-time Prolog implementation was ported to a two-stack-based processor modelled after Charles Moore's Forth Virtual Machine (FVM). Subsequently, this architecture became the basis for the development of the RTX-2000 16-bit processor by Harris Semiconductor. The processor directly executes the Forth language, thus providing a very efficient platform for direct execution of threaded implementations of other declarative languages, such as Lisp and Prolog.

2. INTRODUCTION

The execution flow in a von Neumann machine is typically characterized by the sequence: fetch, decode, modify, and execute. Since a program has to rely on the available instruction set, the programmer's ability to acquire and modify the means of instructing a computer is limited. Development of algorithmic/procedural high level languages has facilitated the human-machine translation process, in an effort to alleviate this semantic gap. The underlying idea behind another, declarative approach to programming is to treat it as a part of problem-solving activity rather than a cumbersome task of mechanical translation conforming to a precise set of syntactical and semantic rules. Using the declarative approach to programming is an intelectually rewarding activity of expressing the knowledge about a problem while using an acceptable level of formalism. This approach may thus lead to much greater insight to the problem at hand.

Software technology in general, and procedural programming languages in particular, have historically followed the evolution of computer hardware. Processors are typically expected to execute their programs in the way hardware allows them to -- rather than in the way the computational problem could be most efficiently formulated and solved.

The development of procedural languages was a natural solution to the growing complexity of hardware programming, since it allowed to insulate the computational task itself from the hardware used to solve it. Naturally then, conventional high level languages provided an easy, but not the most efficient, way of presenting the problem to a computer. As such, *the first high level languages were conceived as bridges over the widening gap between the way programmers express the solutions to their problems, and the way computers actually go about executing them.*

The severity of such a semantic gap became particularly evident in recent years, when modern computing technology clearly began to evolve in two distinctly different directions: *data processing technology and control technology.* The former domain strives for a maximum sustained or average system throughput, at the expense of system program size, power consumption, component count, memory speed and an increasing complexity of program development.

The latter domain reflects what has been traditionally referred to as "real-time", "interactive", or "embedded" processing, and most recently AI technology. Performance of these systems, often completely autonomous, tends to be driven by the external environment. Therefore, their responses must match the rates of change of the environment. From this point of view then, "average" performance of the processor becomes of less significance, because it is its instan-

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

tenous response to an isolated event that will determine the system's behavior. This class of systems seldom relies on the average instruction execution rates, and often depends on fast and predictable behavior of the instructions modifying the code execution flows (such as branches or procedure calls). A processor's behavior in the control environment is also indicative of its suitability for an efficient declarative language implementation, since it usually also reflects on the efficiency and flexibility of its instructions set, code compactness and reentrancy, and the ability to handle recursion.

3. HARDWARE REQUIREMENTS OF DECLARATIVE PROGRAMMING LANGUAGES

Prolog and Lisp share certain common characteristics in terms of the flow of their excution. Both rely heavily on efficient execution of procedure/function calls, and utilize stack-based structures for the exchange and/or passing of run-time parameters. In their reliance on the model of an abstract stack-based machine they also structurally resemble another stack-oriented language: Forth. As noted by Hoffman [Ho87], Lisp, similarly to Forth is not only an interpreted language, but also its own operating system. Moreover, both languages are extensible, i.e. they allow to define new funtions/words that subsequently become parts of the language.

Similarly, Prolog implementations rely on the abstract *Prolog Virtual Machine*. Again, as noted by Odette [Od88], Forth' inherent ability to emulate stack-oriented abstract machines leads to very efficient implementations of Prolog, particularly those based on the Warren Abstract Machine concept.

3.1. Prolog Virtual Machine's (PVM) Execution Flow

The main objectives of a Prolog compiler are to take a collection of Prolog's clauses, and to produce an executable description of their structure. According to Odette [Od87], combination of the validation process and the clause structure leads to the fact that the internal representation of each atomic proposition is the program that realizes the validation process over the proposition.

Although the declarative semantics of Prolog is quite straightforward, the search mechanism related to the procedure invocation, and the unification mechanism representing the passing of information between procedures are considerably more complex.

There are two main stacks in the Prolog Virtual Machine (PVM): the Structure Stack, and the Control Stack.The Structure Stack holds temporary structures created during the computation; it is quite simple with one top pointer only. The Control Stack holds the state information, procedure arguments and variables.

Prolog's execution flow utilizes the Control Stack in the following manner: it pops the top goal for reduction, and pushes the derived goals on the resolvent Control Stack. Clause reduction is accomplished by sequential search and backtracking. Goal reduction is achieved by attempts to unify the goal with available clauses. When the computation progresses forward, the flow of a Prolog program is quite straightforward, and similar to that of a conventional procedural language. Procedure call corresponds to the goal invocation, and statement sequencing is equivalent to the ordering of subgoals in a body of the clause. In the case of recursion, the execution flow also resembles that encountered in conventional, recursion-oriented languages. The difference becomes evident when the computation cannot proceed further: in the conventional approach a runtime error occurs; in the case of a Prolog program the computation backtracks to the last undone choice and another computation path is attempted.

It becomes evident then, that the implementation of the Prolog language on a general purpose computer with a conventional, register-based architecture will not be the most efficient one. There is a need for developing a specialized processor architecture, which would more closely reflect the concept of the PVM, and which would be more effective for AI/Prolog based applications. Specifically, architectural support for efficient goal invocation, backtracking, and recursion would significantly boost the performance of Prolog programs on such a processor.

3.2 Lisp Virtual Machine's (LVM) Execution Flow.

Moon [Mo87] proposes to categorize computer architectures into three subsets. The system architecture defines the user interface in the broad sense. The instruction architecture defines the interface between the compilers and the hardware, and deals with the instruction set, memory management, interrupt mechanism, etc. The processor architecture defines the implementation of the instruction architecture describing the interface between the firmware and hardware. It also details the interface between individual parts of the processor hardware. Although the processor architecture is the main topic of our considerations, it is essential to discuss here also some related issues.

Conseptual objects are the fundamental form of data manipulated by a Lisp system. The variables' values, function arguments and results, and list elements are object references. An object reference usually contains the address of the object representation. Every instruction in the instruction set is contained in a compiled function, which comprises three elements: a fixed overhead, table of constants, and the actual sequence of instructions (a striking resemblance to the way Forth' inner interpreter works). The instructions operate on an abstract model of a stack machine. The 0-address instructions pop their operands off the stack and push their results onto the stack. Other 1-address instructions are

capable of accessing any location in the current stack frame.

A Lisp instruction can use its argument field either as an operand address or as the immediate operation field. Many instructions are simply Lisp functions which can be directly implemented in hardware or firmware. Other instructions, ("built-ins"), are implemented as compiled instructions. They take a fixed number of arguments from the stack and their argument field. Subsequently they return a fixed number of values to the stack. Therefore the ability to efficiently implement such a Lisp-oriented instruction set would significantly enhance the performance of Lisp-based systems. LVM storage allocated to Lisp's functions is divided into three stacks. The Control Stack contains function-nesting information, arguments, local variables, function return values, and temporary objects. The Binding Stack records dynamically bound variables. Finally, the Data Stack contains allocated temporary objects. The protocol for a function call is to push arguments on the stack, and then to execute a call with the function specification (number of arguments, and what to do with the returned values). After the return the arguments are popped off the stack and the function return values pushed on.

For every function call a new stack frame is built on the Control Stack. The frame consists of caller's and callee's copy of arguments, header words, local variables, and arguments for calling the next function. The frame is delimited by pointer registers. A compiled function additionally starts with a sequence of the entry vector instructions. A function return is accomplished by a return instruction with the values to be returned as operands.

The efficiency of the function calling operations has always been a major bottleneck for the Lisp implementations. Using stack-based processor architectures can immensely enhance Lisp implementation and performance. The single calling sequence with the argument patterns matched up at run time simplifies the compiler. Since all registers reside in the stack no register saving and restoring overhead would be required. Additional advantage is derived from stack-based machines' more compact instruction formats and load/store patterns of operation.

4. STACK-BASED PROCESSORS AND FORTH VIRTUAL MACHINE

The Forth language has been developed specifically for high speed real-time control applications, and has always offered the programmer absolute control of the underlying hardware. Forth' concept of a *Virtual Machine* (FVM) encourages writing highly structured, and exceptionally compact and reentrant code. This is accomplished through a two-stack oriented architectural concept of the FVM, with one of the stacks reserved for operand manipulations, and the other reserved for subroutine call return addresses and loop counters.

Forth provides some unconventional features which make it particularly attractive from the declarative languages' point of view. Development of an application program in Forth constitutes a process of defining new "words" in an interpretive way. These words then become new commands in the language's dictionary. Thus the process of formulating the task for a machine is by definition highly structured and modularized, and the code resembles an application-specific language, tailored to the problem at hand. The ability to define new "words" also offers a potentially unlimited repertoir of definable data types and structures.

Because of its threaded nature Forth programs are inherently reentrant and handle recursion very well. Consequently the compiled Forth code tends to be very compact and fast even on conventional processors.

Due to the stack-oriented structure, the overhead associated with such operations as subroutine calls and returns, branches and interrupts, and parameter passing is minimal. Moreover, since Forth does not favor register-to-register operations, the overhead traditionally associated with saving the task context in a register-oriented language (such as C) is also minimal.

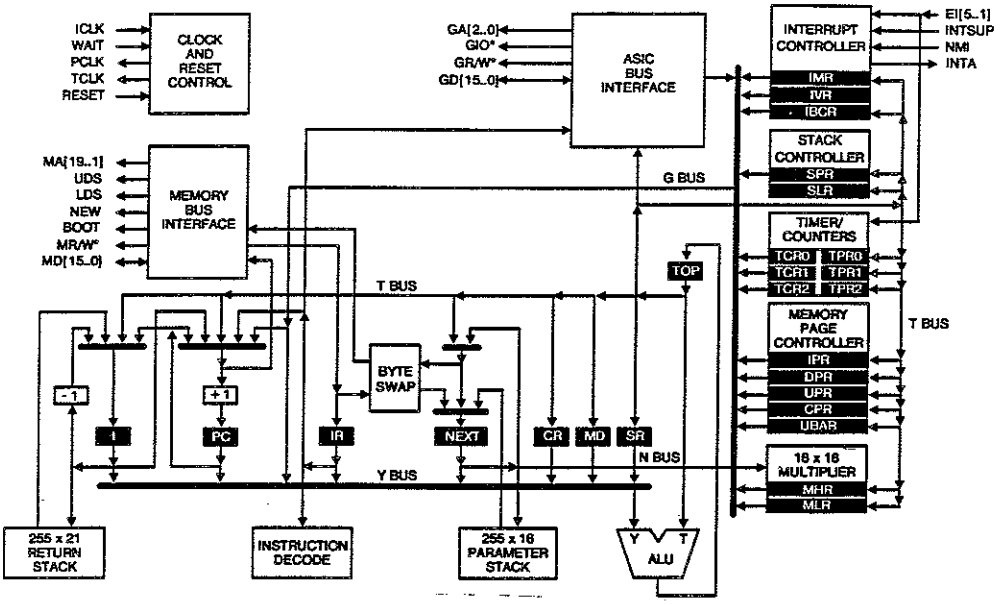
A number of real-time software developers have pointed out certain structural similarities between Forth and Lisp [Ho87], or Prolog and Lisp [Od87]; subsequently they elected Forth as a platform language for the development of experimental real-time expert systems.

Implementing the Virtual Machine in hardware (RTX-2000 processor) offers another dramatic performance improvement by virtue of transferring the majority of the Virtual Machine's internal constructs into the hardware. It also illustrates the performance gains that can be attained by the development and use of a truly integrated software-hardware environment, long advocated by many researchers.

5. THE RTX-2000 PROCESSOR ARCHITECTURE

The central part of the RTX-2000 is the "RTX engine" originally architected by Charles Moore, the inventor of the Forth language. The engine's internal data flow very closely reflects the structure of Forth' Virtual Machine, with its two data- and address-related LIFO stacks, and heavy emphasis on implicit addressing [Ma88A], [Ma88B].

The RTX engine, in many respects reflects the most desirable features of an ideal RISC processor. It's internal data paths are highly parallel, and directly routable to the ALU in a single machine cycle. Any of



RTX 2000 FUNCTIONAL BLOCK DIAGRAM

the internal 8 directly addressable registers can feed its contents through one of the inputs to the ALU -- usually in parallel with another one or two non-ALU related operations. Moreover, the engine offers parallel access to the ALU from an external general purpose input port.

All arithmetic instructions are performed on the two top locations of the engine's Parameter Stack, and contained in the core's Top (T) and Next (N) registers. The remaining locations of the Parameter stack are in the on-chip stack RAM. Similarly, the top location of the Return (or Index) stack is stored in the engine's I register, with the extension of the stack in the Return Stack RAM. The Return Stack's major purpose is to ensure a quick means of providing the return addresses for subroutine calls and the values of loop counters. Both stack RAMs appear to the engine as LIFOs, allowing the engine to utilize a very efficient implicit (0-address) addressing scheme.

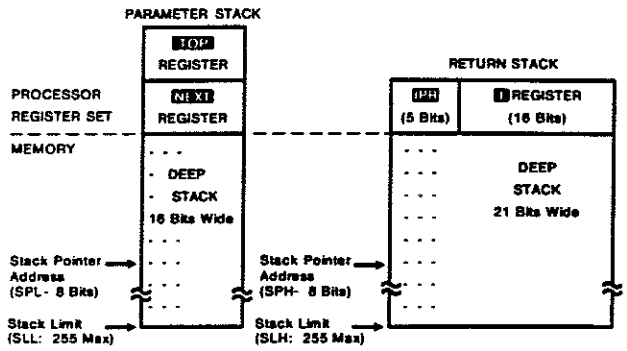
In addition to the stack registers, the core also employs two registers dedicated to the support of arithmetic step operations: multiply step, divide step and square root step. The first two are assisted by the MD register, while the third one is assisted by both the MD and SR registers. While not in use, both registers can be used as scratchpad registers.

The instruction address is kept in the Program Counter register (PC), while the actual instruction, upon fetch is stored in the instruction register (IR). Finally, the specific status bits of the processor, such as values of carries, interrupt status, etc, are stored in the engine's configuration register (CR).

With the exception of memory reference instructions, which consume two machine cycles each (unless executed sequentially), the processor executes its instructions in a single clock cycle. If one bears in mind that a single RTX instruction may contain as many as four FORTH primitives, the computing power of the engine becomes obvious. Moreover, instructions which actually consume many clock cycles on other processors -- such as subroutine calls and returns, branching or interrupts -- are reduced to a single machine cycle, thus contributing to a very high degree of predictability and observability of the execution flow, essential for efficient realtime code development and execution.

Fast stack pointer generation and control are essential for efficient execution of data/address pushes and pops executed by the RTX-2000 during almost every machine cycle. The on-chip stack controllers are optimized for handling these tasks within a half of the engine's machine cycle. Stack controllers internally maintain the absolute address of the current 16-bit "top" value of the memory location onto which the next datum will be pushed. This value, stored in the stack controller's Stack Pointer Register, can be read by the processor at any time.

A second register associated with each task is a Stack Limit Register, which allows the programmer to establish a virtual limit on the depth of the stack (less than its physical limit and equal to 256 words in the case of the RTX-2000). Consequently, the Stack Overflow flag can be set by the controller several words in advance, thus giving the engine an extra stack space on which to store intermediate data during the process of moving the data to main memory. Whenever the "bottom" of the stack is reached, a



PARAMETER AND RETURN STACKS - FUNCTIONAL STRUCTURE

Stack Underflow flag is set.

In the RTX-2000 processor implementation, the Parameter Stack, is 16-bits wide. The Return Stack width is 21 bits in order to provide the processor with a total address range of 1 Mbyte. The total addressable space is divided into sixteen 32 kiloword code/data pages. A special class of memory reference instructions permits rapid access to 32 word groups of "user memory" by embedding a short literal address in the instruction itself.

The on-chip interrupt controller uses 14 prioritized interrupts (including the NMI).Of these 14 interrupts, 9 are used by the internal peripherals of the processor, including stack underflow and overflow interrupts, three timer/counter interrupts, one software interrupt and a non-maskable interrupt (NMI). The remaining 5 interrupts are available as inputs to the processor, with three of them sharing the input pins with the counter/timer clock inputs. The function of these shared inputs is programmed internally by setting respective flag bits in the processor's IBCR register. Upon receipt of an interrupt, it takes the processor only 4 machine cycles (400 ns) to reach an interrupt handler.

The RTX-2000's on-chip 16x16 bit parallel multiplier is capable of performing a single multiplication of either signed (two's complement) or unsigned integers in a single machine cycle (100 ns), with additional overhead involved in fetching the operands. A full 32-bit precision result is stored in the multiplier's output registers. The operands for multiplication are supplied by the T and N registers, and the double precision result of the multiplication can be subsequently placed in these locations if so instructed by the user program.

The RTX-2000 uses three on-chip 16-bit down-counters, which can be individually configured as either timers or event counters. In their initial mode, all three counters are configured as timers, with the source clock applied directly from the RTX-2000's internal TCLK. The count limits for each timer can be individually preloaded by the processor, by loading each timer's preload register.

6. CONCLUSIONS

Experimental implementation of a Prolog compiler compilerported to the NC4000 processor (the predecesor of the RTX-2000) was based on the Warren Abstract Machine implementation, and executed 132,000 PVM instructions per second [Od88]. A new version of the compiler running on the RTX-2000 processor with a 100 ns machine cycle is expected to execute at least 300,000 PVM instructions per second, or an equivalent of 30,000 LIPS (logic inference per second) -- approximately 5 times faster than a C-based implementation of Prolog running on a VAX 11/780.

7. REFERENCES

[Ha85] H.M. Harris, "Forth As the Basis for an Integrated Operations Environment for a Space Shuttle Scientific Experiment", *Journal of Forth Applications and Research*, 4(2), 1985.

[Ho87] U. Hoffman,"A LISP Kernel for the NC4000", *Proceedings of the euroFORML Conference*, Heilbronn, West Germany, 1987.

[Li88] J. Liebowitz, "Introduction to Expert Systems", Mitchell Publishing, Santa Cruz, CA, 1988

[Ma88A] C. W. Malinowski, "RTX-2000: A High Performance Controller for Embedded Robotic Applications", *Proceedings of the 13th Intnl. Conference on Microelectroelectronics: ELECTRONICA 88*, Munich, West Germany, 1988.

[Mo87] D. A. Moon, "Symbolics Architecture", *IEEE Computer*, pp.43-52, January 1987.

[Ma88B] C. W. Malinowski , "A New 16-bit Realtime Controller Sets New Performance Standards Through Direct Execution of Forth", *Proceedings of the First Australian Forth Symposium*, Sydney, Australia, 1988.

[Pa86] W. Paloski et al., "Use of the Forth Based Prolog for Real-Time Expert Systems", *Proceedings of the Rochester Forth Conference*, Rochester, NY, 1986.

[Od87] L.L. Odette, "Compiling Prolog to Forth", *Journal of Forth Application and Research*, pp. 487-534, 1987.

[Od88] L.L. Odette, W. Wilkinson, "Forth and Prolog on the Forth Machines", *Proceedings of the First Australian Forth Symposium*, Sydney, Australia, 1988.

A STUDY OF THE GENERALIZATION IN MULTI-LAYER FEED FORWARD NEURAL NETWORKS

Francis J. Gerrity
Martin Marietta Corporation
Orlando Division
Post Office Box 5837
Orlando, Florida 32855

Michael Georgiopoulos
Department of Electrical Engineering
University of Central Florida
Orlando, Florida 32816

George Papadourakis
Department of Computer Science
University of Crete
Iraklion, Crete GREECE

ABSTRACT

This paper examines the capability of a 3-layer feed forward network, trained with the back propagation algorithm to study the capability of the network to generalize. Using English characters as inputs, the network is analyzed for its ability to identify characters that have translation offsets and characters that have been rotated. Also studied is the networks immunity to three cases of noise. In the first case the network is trained with pure characters and noisy characters, then the networks response to characters with an intermediate amount of noise is studied. In the second case the network is trained to pure characters, then the networks response to increasing noise is observed. Finally an effort is made to study the network response to noise when the network is trained with noisy characters.

INTRODUCTION

A type of neural network which has been used successfully as a classifier is the multi-layer perceptron [Lippman, 1987]. Multi-layer perceptrons [Minsky, 1969] are feed-forward networks with one or more layers of nodes between the input and output nodes. A node in a layer sums N weighted inputs, plus offsets, and passes the result through a nonlinearity to the next layer.

Much of the current research in the field of neural networks is focused on the design of efficient learning algorithms to train a neural network to perform a specific task. One of the most successful algorithms developed so far to train a multi-layer perceptron is the back propagation algorithm [Kung, 1988, Rumelhart, 1986].

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

This paper investigates the capability of a 3-layer, feed forward, neural network using the back propagation algorithm to recognize patterns for which it has not been trained. Without such generalization, neural networks would be of little practical significance. Merely learning the training patterns can be accomplished by storing these patterns and their desired outputs in a look-up table. Ordinary English alphanumeric characters are used as the input medium. Indeed, although English characters are used, any type of characters could have been used with comparable results, although it is safe to assume that some types of characters, for example Chinese characters, would probably require more pixels in order to provide the resolution required. Other types of input could be used as long as the representation is suitable to the network.

The ability of the network to generalize is based on a set of carefully chosen experiments. In particular, this paper examines the ability of the network to generalize under conditions of translation, rotation, and noise corruption. The input patterns are 12 by 12 binary pixel values representing handwritten characters from the Canadian Standards Association Alphanumeric Character Set For Handprinting [1983] standard. Five of these characters are shown in Figure 1.

EXPERIMENTAL RESULTS

Preliminaries

The results shown in this paper use a 3-layer network with a learning rate (σ) of 0.35. This net contained 144 nodes and 750 associated weights in the first hidden layer and 5 nodes and 25 associated weights in each of the second and third hidden layers.

The threshold limit is set to 0.1. In other words for target values of 1 and 0 the output values must be at least 0.9 and at most 0.1 respectively. Other

threshold limits could have been used, but the less restrictive the threshold limit, the less tolerant the trained network is to input errors. Convergence occurs when all of the outputs fall within the threshold limits for all patterns.

The network output error (NOE) is one means of examining network performance. The network output error is defined as follows:

$$NOE = \frac{\sum_{p=1}^{M} \sum_{i=1}^{N_L} | t_{pi} - a_{pi} |}{M * N_L}$$

where 'M' is the number of patterns, 'N_L' is the number of nodes in the output layer, 't_{pi}' is the target output at node 'i' for pattern 'p', and 'a_{pi}' is the actual output at node 'i' for pattern 'p'.

Translation

This experiment determines whether or not the network can recognize characters that have translational offsets. The network is taught five characters, a left justified 'A', 'B', 'C', 'D', and a right justified 'A' as shown in Figure 2. After the network is trained to recognize the patterns in Figure 2, it is then presented with the translated patterns shown in Figure 3. The network is NOT run to convergence, rather the characters are presented and the output is measured. Results indicate that the network has the tendency to recognize translated patterns which it has not been taught.

Table 1

Input: Pattern	Output: 'A'	'B'	'C'	'D'
'A' Shifted 1-bit	0.78	0.07	0.04	0.03
'A' Shifted 2-bits	0.91	0.04	0.07	0.01
'A' Shifted 3-bits	0.78	0.01	0.31	0.01
'A' Shifted 4-bits	0.29	0.07	0.44	0.01

Network Output Error = 0.119

The above results show that all translational positions of the character 'A' are recognized except for the 'A' that is shifted 4-bit positions to the right.

This experiment was run several times. Before each run, the weights were given new random values. The network was trained to the first set of patterns as before, then the second set of patterns containing translated 'A's was presented. The network output error of the runs varied between 0.109 and 0.268. This small error seems to indicate that the network was good at recognizing translated 'A's. In reality, however, there are some cases when the network fails to recognize certain translated 'A's (see also Table 1). In general, we observed that the capability of the network to recognize translated characters is sensitive to the initial set of weights, as well as, the learning rate.

Rotation of Characters

The goal of this experiment is to determine whether or not the network can recognize characters that have been rotated. The network is taught five characters, 'A' in four positions of rotation (see Figure 4), and a non-rotated 'B'. After the network has learned these patterns it is presented with the characters of a non-rotated 'A', and 'B's in four rotational positions as shown in Figure 5 with the following results.

Table 2

Input: Pattern	Output: 'A'	'B'
Non-Rotated 'A'	0.90	0.09
Non-Rotated 'B'	0.10	0.91
'B' Rotated 90°	0.71	0.20
'B' Rotated 180°	0.94	0.05
'B' Rotated 270°	0.23	0.77

Network Output Error = 0.420

This experiment was run several more times, each time with at different set of initial conditions. The results show that the network is not able to generalize what is has learned about the rotation of one character and apply it to another character. For example, the network calculated the 'B' rotated 180° as being an 'A'. Furthermore, our results indicated that the capability of the network to recognize rotated characters is sensitive to the initial set of weights, as well as, the learning rate.

Random Noise 'A'

The goal of this experiment is to determine if the network has the ability to recognize characters corrupted with noise. The network is first trained with five pure characters, as shown in Figure 1, as well as with characters containing 10% random noise, as shown in Figure 6. The network is then presented with characters corrupted with an intermediate (5%) amount of noise, as shown in Figure 7 with the following results:

Table 3

Input: Pattern	Output: 'A'	'B'	'C'	'D'	'E'
5% Noisy 'A'	0.91	0.03	0.07	0.00	0.01
5% Noisy 'B'	0.06	0.90	0.03	0.04	0.00
5% Noisy 'C'	0.09	0.03	0.89	0.03	0.00
5% Noisy 'D'	0.00	0.06	0.05	0.91	0.08
5% Noisy 'E'	0.01	0.01	0.00	0.27	0.87

Network Output Error = 0.057

The results show that the output is not good enough for convergence. But if we relax the criteria for convergence, such that an output greater that 0.5 represents a '1' otherwise the output is a '0', then the network does very well at recognizing characters with an intermediate amount of noise.

In this experiment the 5% noise was carefully controlled to be a subset of either the pure characters or the characters with 10% noise. The case of purely random noise is dealt with in a later experiment.

Random Noise 'B'

The goal of this experiment is to determine how much random noise corruption a network trained to pure characters can accept.

Initially, the network is trained to recognize pure characters, as shown in Figure 1. Then the network is presented with different noisy versions of each character and the network output error is observed. One hundred samples are presented to the network at each increment of 0%, 5%, 10%, 15%, and 20% noise. The average network output error (ANOE - the network output error averaged over the 100 performances of the experiment) is shown in Figure 8 as a function of the amount of noise introduced. The plot shows that the performance of the network deteriorates as the percentage of the noise corruption increases. But, the fact that the average network output error increases gracefully shows that the network is able to generalize to some extent.

Random Noise 'C'

An effort to improve the sensitivity of the network to characters containing noise is conducted in this experiment. First, the network is trained with characters containing 3% noise. Training continues until a full iteration of all patterns (containing 3% noise) produces outputs, all of which, meet the threshold limit criteria. After training, the network is presented with different noisy versions of each character and the network output error is observed. One hundred samples are presented to the network at each increment of 0%, 5%, 10%, 15%, and 20% noise and the average network output error is computed (the network output error averaged over all these samples). This procedure is repeated for training patterns containing 5% and 7% noise.

The results are shown in Figure 9 with the average network output error being a function of the amount of noise introduced. The plot shows that the performance of the network deteriorates as the percentage of the noise corruption increases. However, a comparison between figures 8 and 9 shows that a network which has been trained with noisy characters is more immune to noise than a network which has been trained solely with pure characters.

CONCLUSION

This paper has shown that a 3-layer, feed forward neural network using the back propagation algorithm has some ability to generalize characters containing translational offset but not rotated. The network does well with random noise if it is trained with patterns containing random noise. In all cases the performance of the network is dependent upon initial conditions (initial set of weights and the learning rate).

REFERENCES

Canadian Standards Association, "Alphanumeric Character Set For Handprinting", Standard Z 243.34-M1983, July 1983.

Kung, S.Y., Hwang, J.W., "Systolic Architectures For Artificial Neural Nets", submitted to IEEE Journal of Robotics and Automation, 1988.

Lippman, Richard P., "An Introduction to Computing with Neural Nets". IEEE ASSP magazine,4: pp 4-22, April 1987.

Minsky, M., and Papert, S., "Perceptrons". MIT Press, Cambridge, Massachusetts, 1969.

Rumelhart, D.E., McClelland, J.L., and the PDP Research Group. "Parallel Distributed Processing (PDP) Exploration in the Microstructure of Cognition (Volume 1)". MIT Press, Cambridge, Massachusetts, 1986.

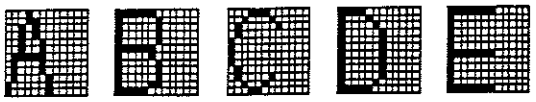


Figure 1. Input Patterns A, B, C, D, E.

Figure 2. Left and Right Justified A's.

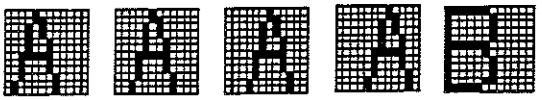


Figure 3. 'A's Shifted 1, 2, 3, 4 Bits.

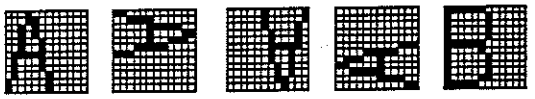


Figure 4. Input Patterns of Rotated A's.

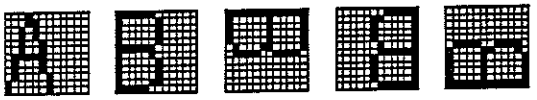


Figure 5. Input Patterns of Rotated B's.

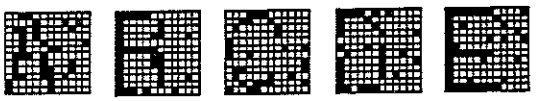


Figure 6. A, B, C, D, E, With 10% Noise.

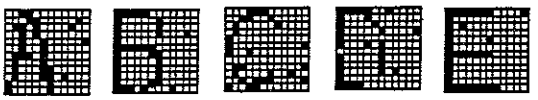


Figure 7. A, B, C, D, E, With 5% Noise.

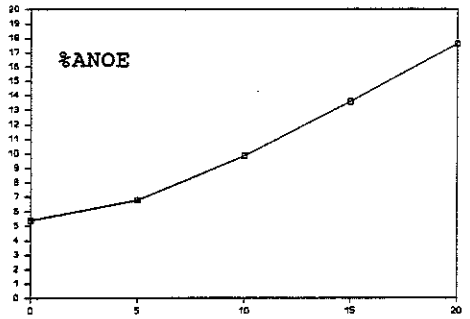


Figure 8. Average Network Output Error Versus Noise.

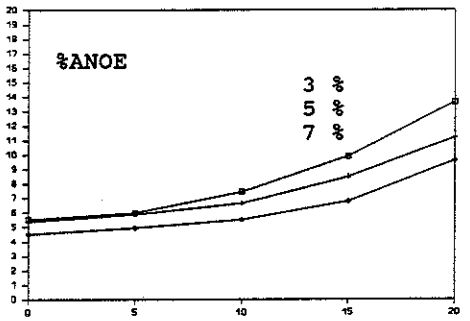


Figure 9. Average Network Output Error Versus Noise For Different Noise Training.

CLASSIFICATION OF EVOKED POTENTIALS USING MULTILAYER FEEDFORWARD NEURAL NETWORKS

Ozcan Ozdamar and Dogan Alpsan
University of Miami
Dept. Biomedical Engineering
P.O. Box 248294
Coral Gables, Florida 33124

ABSTRACT

Multilayer feedforward neural networks are investigated to classify auditory brainstem evoked potentials for hearing threshold determination. Simulations using limited number of training patterns show that such networks can successfully classify these recordings for automated electrophysiologic audiometry. The experiments show that the major factor that affects the classification rate is the size of the training pattern set. Number of hidden layer elements or addition of noise to the training data does not appear to change performance. Such manipulations, however, affect training time considerably.

INTRODUCTION

Recognition and classification of biomedical signals such as electrocardiograms (ECG) and electroencephalograms (EEG) present a great challenge due to their great complexity and variability. These signals, typically recorded from subjects with electrodes using high amplification are classified and diagnosed by medical experts. There is, however, a great need for automated recognition and classification of such recordings for the development of intelligent medical devices

In this study classification of evoked potentials (EP), which are computer averaged brain waves in response to sensory stimulation, is investigated using neural networks. Early portion of the auditory EPs, known as auditory brainstem responses (ABR), are routinely used to estimate hearing thresholds in newborns, infants and otherwise behaviorally untestable patients. Numerous methods have been proposed and tested with various success rates to automate this process (Delgado et al. 1988; Mason and Adams 1984; Salvi et al. 1987). These methods essentially classify ABRs with traditional statistical pattern recognition procedures. They do not, in general, reflect the way human experts make their decisions based on the generalizations of their previous experience with these signals.

Recently syntactic methods are investigated for the classification of ABRs (Madhavan et al. 1986; Ozdamar et al. 1987). Syntactic methods are essentially rule based systems and come closer to the reasoning approach used by experts. These methods, however, require generation of hundreds of rules by an expert and cannot be tailored easily to satisfy the conflicting views of individual experts.

This study explores if a neural network model can classify ABRs for hearing threshold detection using only a limited number of recordings for training. The performance of various feedforward neural network types are evaluated for best performance. The effects of the network configuration and the composition of the training set on learning and convergence are also examined.

METHODS

In this study feedforward multilayer neural networks with one hidden layer are used. These networks are trained to classify any ABR test signal into two classes: "Response" and "NoResponse". Training is accomplished by using the backpropagation procedure (Rumelhart et al. 1986). The objective of this type of learning is to try to minimize the difference between the desired and the actual outputs by feeding a measure of this error back to the inputs. Each weight is adjusted in proportion to its contribution to the error.

In the present model, an input layer consisting of 64 processing elements is used. The amplitude values of the preprocessed ABRs are fed into the input layer in temporal order. Preprocessing consists of smoothing and compressing the original waveform to 64 points. Magnitudes are

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

range translated and normalized to vary between 0.0 and 1.0 thus eliminating absolute wave size information. The number of processing units in the hidden layer is varied between 2 and 15 for different simulation experiments. The output layer consists of 2 units representing "NoResponse" and "Response" classes.

The system is trained and tested using 253 ABR recordings from 8 normal hearing adult subjects (10 ears). The recording were obtained by using a dual processor based microcomputer system using 100-3000 Hz band-pass filtering. Alternating click stimuli at 20 Hz repetition rate were used. Responses were obtained at many subthreshold and suprathreshold click intensities with 10 dB intervals. 1024 sweeps each 12.8 msec duration were averaged and stored on magnetic media for later analysis.

Before training, all weights and biases are initialized to random values between -1.0 and 1.0. The network is presented with a set of training waveforms consisting of subthreshold and suprathreshold responses . The ratio of the number of the subthreshold to suprathreshold recordings is kept at 1:2 in all the simulations. Training set is prepared by an EP expert who examined all the recordings and classified them into the two groups. Examples of "NoResponse" and "Response" recordings are shown in Fig.1. As observed, these signals contain a fair amount of noise and suprathreshold recordings form a heterogeneous group due to acquisition at different sound intensities.

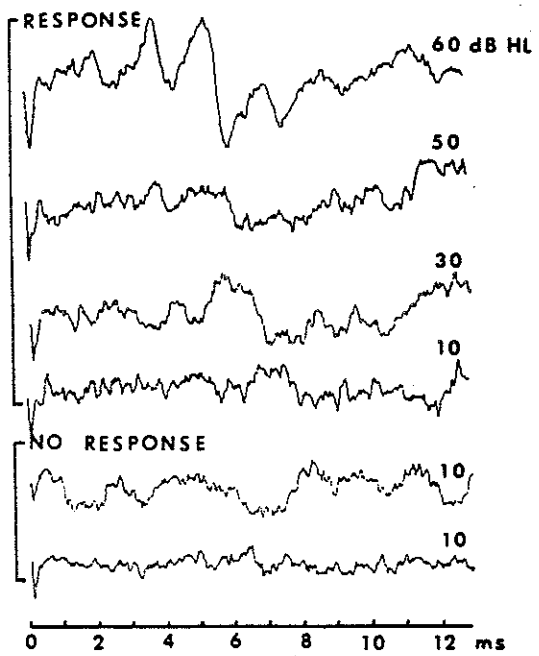


Fig.1 - Examples of ABR recordings used in the training set.

Training is started by presenting one of the patterns from the training set to the input layer and specifying the desired output pattern. The outputs of the processing units in the hidden and output layers are calculated by

$$y_j = \frac{1}{1 + \exp\left[-\left(\sum_i x_i w_{ij} + b_j\right)\right]}$$

where y_j is the output of the unit j, x_i is the output of a unit that is connected to j, w_{ij} is the weight of the connection from i to j and b_j is the bias of the unit j.

The computed output pattern is then compared with the desired output and the discrepancy between them is calculated according to a specified error function. In this study the following improved error function (Franzini 1987) is used to increase the learning rate:

$$E = -\sum_j \ln\left(1 - (y_j - d_j)^2\right)$$

where y_j and d_j are the actual and desired values of an output unit , respectively.

The resulting error is used to modify the weights of the processing units using the rule

$$w_{ij}(t+1) = w_{ij}(t) + \varepsilon \delta_j y_j + \alpha\left(w_{ij}(t) - w_{ij}(t-1)\right)$$

where ε is the learning rate parameter and α is the momentum term. In all simulations the learning rate was 0.01 and the momentum term was 0.9.

All the training inputs are presented iteratively until the errors in both outputs fall below a specified tolerance level (0.2). One pass through the whole set of training patterns is defined as an iteration. In some simulations random noise is added to each training input pattern according to the following formula, x=x+x*r where x is the response amplitude and r is a random number between -0.5 and +0.5 (Elman and Zipser 1988). Each simulation is repeated 10 times with different random initial weights. A simulation was terminated as not converging after 10,000 iterations and a new one was initiated until 10 converging simulations are obtained. After training is complete, the remaining recordings which are not included in the training set are used to test the performance of the network.

RESULTS

In general, all the networks tested gave satisfactory classification rates. In order to determine the hidden layer size for best performance, networks were simulated with hidden units varying from 2

to 15. The results of these simulations are summarized in Table I. The performance of these networks were independent of the number of hidden units with an overall average of 75.3% correct classification.

The number of iterations needed for convergence varied with the number of hidden units. As seen in Table I, the network with 6 hidden units was slowest to converge while the one with 15 units converged most quickly. The relationship of convergence with hidden unit size was complex. Nevertheless, a general trend could be observed such that networks with fewer hidden units converged more slowly. Failure to converge was also observed only for networks with fewer hidden units. The network with 2 hidden units was particularly difficult to converge with 6 failures in 16 trials with different initial random weights.

Table II summarizes the results of the experiments to study the effect of the size of training patterns on the classification rate. In these experiments the number of hidden units was kept constant at 10 and the number of patterns included in the training set were varied. The network performance increased with the number of training patterns up to 45 patterns. A training set larger than 45 did not improve performance any further. The training set size also affected the rate of convergence. The number of iterations required for convergence initially increased and then decreased with increasing training set size.

The effects of adding noise to the training patterns on performance and convergence are shown in Table III. Learning with noise produced no improvement in network performance but slowed down convergence by about 28%.

In order to investigate learning strategies among different networks, learning curves were plotted. A learning curve is a plot of the percentage correct classification as a function of training iteration (Alpsan and Ozdamar 1989). Learning curves showed that variations in both the hidden unit size and training set size resulted in different learning behaviors. Two such learning curves are shown in Fig.2. While networks with 2 to 6 hidden units exhibited one type of learning, networks with 10 to 15 units showed another type. Likewise there were differences in learning for different number of training patterns. While training with the set of 21 examplars resulted in one learning pattern, the other sets showed different learning curves.

DISCUSSION

This study shows that feedforward neural networks can successfully classify ABRs into "Response" and "NoResponse" classes for hearing threshold determination purposes. Classification rates are comparable to those of previous studies (Delgado et al. 1988). Performance of all the networks is satisfactory such that any one of them can be installed in an on-line threshold-tracking device.

Table I. Effects of the number of hidden layer elements on the network performance. Numbers show percentage means and (standard deviations).

Network Class. Expert Decision	Correct Response	Correct NoResponse	Incorrect Response	Incorrect NoResponse			
No. of HL Units	(TP)	(TN)	(FN)	(FP)	Total Correct	No. of Iterations	No. not Converged
2	68.2 (1.2)	8.2 (0.4)	12.6 (1.2)	11.0 (0.4)	76.4 (1.0)	2433 (1470)	6
3	65.7 (3.2)	9.1 (1.5)	15.1 (3.2)	10.0 (1.5)	74.4 (1.6)	2078 (1209)	1
4	68.2 (1.4)	8.4 (0.5)	12.6 (1.4)	10.8 (0.5)	76.6 (1.5)	2482 (1297)	1
6	63.2 (3.9)	10.3 (1.8)	17.6 (3.9)	8.9 (1.8)	73.5 (2.9)	3032 (1722)	-
10	65.4 (2.9)	9.5 (1.2)	15.4 (2.9)	9.7 (1.2)	74.9 (2.2)	1244 (521)	-
15	66.9 (2.4)	9.1 (1.6)	13.9 (2.4)	10.1 (1.6)	76.0 (2.3)	909 (64)	-

Table II. Effects of the number of examplars on the network performance. Numbers show percentage means and (standard deviations).

Network Class. Expert Decision	Correct Response	Correct NoResponse	Incorrect Response	Incorrect NoResponse			
No. of Examplars	(TP)	(TN)	(FN)	(FP)	Total Correct	No. of Iterations	No. not Converged
21	58.9 (2.2)	7.1 (1.3)	19.6 (2.2)	14.5 (1.3)	66.0 (1.8)	1185 (672)	-
30	62.7 (3.3)	9.5 (2.5)	15.9 (3.3)	11.9 (2.5)	72.2 (1.3)	2917 (2563)	2
45	65.8 (2.9)	9.0 (0.9)	14.0 (2.9)	11.2 (0.9)	74.8 (2.4)	1396 (563)	2
60	65.4 (2.4)	9.5 (1.2)	15.4 (2.9)	9.6 (1.1)	74.9 (2.2)	1244 (521)	-

In this study many simulations were conducted with different number of hidden layer elements to improve the pattern recognition performance of the system and its learning rate as measured by the number of iterations required for convergence. Past studies show that the effects of the number of hidden units on the network performance and the speed of convergence depend on the order of the task being learned ("the number of regularity features"). In most pattern recognition applications the percent correct classifications improve with increasing number of hidden units up to a critical number (Gorman and Sejnowski 1988). Above this critical number, adding more units to the hidden layer does not improve recognition any further. Some problems, on the other hand, are insensitive to the number of hidden units and no improvement is observed (Bound and Lloyd 1988). Similarly, no improvement on classification rates with different hidden units is observed in the present study.

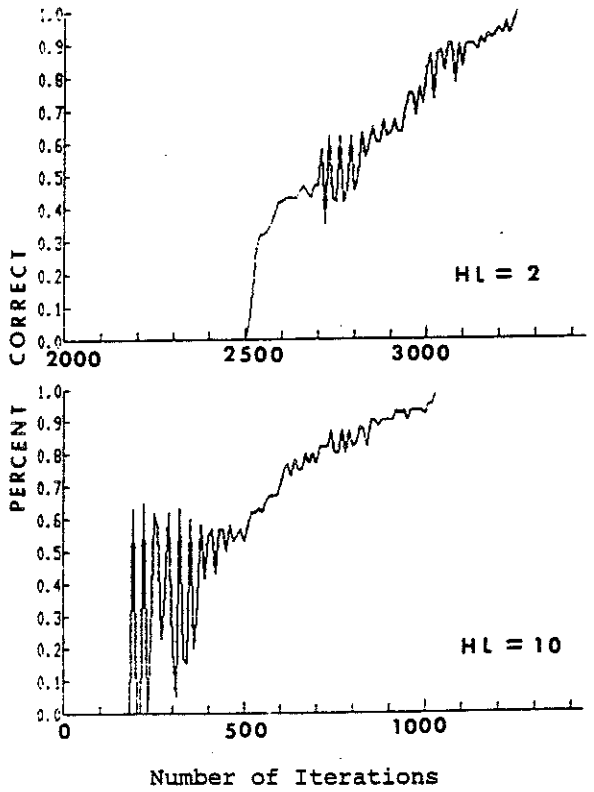


Fig.2 - Examples of two different learning curves observed in the study.

The performances of the networks with fewer hidden units show less variability as seen in standard deviations. This suggest that these networks are less sensitive to initial conditions.

The number of iterations necessary for learning has been found to decrease with increasing number of hidden units (Kung and Hwang 1988; Rumelhart et al. 1986). For the XOR problem, learning time was found to be linearly reduced with the logarithm of the number of hidden units (Rumelhart et al. 1986). For some pattern association tasks, however, the reduction in the number of training iterations was abrubt. The abrubt decline occurs at a hidden unit size corresponding to the order of the task (Kung and Hwang 1988). The difference in the convergence rates between networks with small (2-6 units) and large (10-15 units) hidden layers suggests that the number of regularity features in the ABR sample studied must be between 6 and 10. This conclusion is also supported by the differences in learning

Table III. Effects of noisy traing set on the network performance.
Numbers show percentage means and (standard deviations).

Network Class. Expert Decision	Correct Response	Correct NoResponse	Incorrect Response	Incorrect NoResponse			
Type of Examplars	(TP)	(TN)	(FN)	(FP)	Total Correct	No. of Iterations	No. not Converged
Normal	65.4 (2.9)	9.5 (1.2)	15.4 (2.9)	9.7 (1.2)	74.9 (2.2)	1244 (521)	-
Noisy	65.8 (3.5)	9.1 (1.8)	15.1 (3.5)	10.1 (1.8)	74.9 (2.6)	1589 (990)	-

curves which show an abrubt change when the number of hidden units increase from 6 to 10.

It has been observed that noise added to the training patterns affects network preformances in various ways. Random noise introduced into the training patterns improved performance in certain tasks (Sietsma and Dow 1988) while degraded performance in others (Elman and Zipser 1988). However, even for those tasks in which performance was degraded by random noise, certain kinds of noise (e.g., noise proportional to each input value) significantly improved performance (Elman and Zipser 1988; Franzini 1987). In the present study, noise of this latter type had no effect on network performance, but slowed learning. This could be due to already noisy nature of ABR signals. Thus adding external noise does not have any further effect on performance.

Acknowledgement

This study was partially supported by Florida High Technology and Industry Council (FHTIC). Dr. Alpsan was a Fulbright scholar from the Middle East Technical University, Ankara, Turkey.

REFERENCES

Alpsan, D. and O. Ozdamar. 1989. "A back propagation network for classifying auditory brainstem evoked potentials: Input level biasing, temporal and spectral inputs and learning patterns", submitted.

Bounds, D.G. and P.J. Lloyd. 1988. "A multilayer perceptron network for the diagnosis of low back pain." IEEE Ann. Int. Conf. Neural Networks, II:481-489.

Delgado, R.E., O. Ozdamar, E. Miskiel. 1988. "On-line system for automated auditory evoked response threshold determination." IEEE, Proc.10th Ann. Int. Conf. Eng. Med. Biol. 1472-1473.

Elman, J.L. and D. Zipser. 1988. "Learning the hidden structure of speech." J.Acoust. Soc. Am., 83:1615-1626.

Franzini, M.A. 1987. "Speech recognition with back propagation." IEEE, Proc. 9th Ann. Int. Conf. Eng. Med. Biol., 1702-1703.

Gorman, R.P. and T.J. Sejnowski. 1988. "Learned classification of sonar targets using a massively parallel network." IEEE Trans. Acoust. Speech Signal Proc., 36:1135-1140.

Kung, S.Y. and J.N. Hwang. 1988. "An algebraic projection analysis for optimal hidden units size and learning rates in back propagation learning." IEEE Ann. Int. Conf. Neural Networks, I:363-370.

Madhavan, G. P., H. De Bruin, A.R.M. Upton, M.E. Jernigan. 1986. "Classification of brainstem auditory evoked potentials by syntactic methods." Electroenceph. Clin. Neurophysiol., 65:289-296.

Ozdamar, O., C.N.Lopez, R.E. Delgado. 1987. "Syntactic recognition of auditory brainstem evoked potentials for neurological diagnosis." IEEE, Proc. Miami Technicon '87, 264-267.

Rumelhart, D.E., G.E. Hinton, and R.J. Williams. 1986. "Learning internal representations by error propagation." In Parallel Distributed Processing, D.E. Rumelhart, J.L. Mcclelland (Eds.), Vol:1, MIT Press, Cambridge, MA.

Salvi, R.J., W. Ahroon, S. Saunders, S.A. Arnold. 1987. "Evoked potentials: Computer automated threshold tracking procedure using an objective detection criterion." Ear and Hearing, 8:151-156.

Sietsma, J., R.J.F. Dow. 1988. "Neural net pruning- Why and how." IEEE Ann. Int. Conf. Neural Networks, I:325-333.

FUZZY DATA COMPARATOR WITH NEURAL NETWORK POSTPROCESSOR A HARDWARE IMPLEMENTATION

Paul Basehore, Joseph Yestrebsky, Gerald Reed
Micro Devices
Special Products Division Of Chip Supply
5643 Beggs Road, Orlando, FL 32810-2603

ABSTRACT

In this paper we describe a Fuzzy Set Comparator (FSC) designed for adaptive ranking, and ranking fuzzy data in groups by certain predetermined characteristics (i.e., modified single index ranking). The FSC is intended to simplify the implementation of systems where decisions must be made rapidly from inaccurate, noisy or real-time variable data. Either Hamming or Linear distance may be selected as the comparison metric. A neural network postprocessor ranks the comparisons, providing a simple hardware implementation and superior rank calculation speed. The FSC is implemented in 1.5 micron CMOS technology. As it is bus-oriented, the FSC may be incorporated into a microprocessor environment, or operated autonomously.

INTRODUCTION

In practice, many data sources, such as those from optical, speech, telemetry, and judgmental sources, tend to be inaccurate, noisy, or otherwise variable. Such data are therefore difficult to use directly in decision making or pattern recognition applications. The action of evaluating (ranking) 'fuzzy' data defuzzifies those data. Defuzzification must take place to make the data useful in any decision making process. However the defuzzification of data by single index ranking (Kandel 1986) may result in reduced resolution and loss of potentially vital information. Techniques for handling this problem are being investigated (e.g., multiple indexed ranking (Mabuchi 1988)), but these techniques are in their early stages of development. It is expected that a modified single index ranking will be adequate for most practical problems.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

In this paper we describe the theory and performance of a recently developed Fuzzy Set Comparator (FSC) integrated circuit (IC). The processing element of the FSC is based upon the mathematics of Fuzzy Set Theory. A postprocessor comprised of a tandem pair of neural networks computes the Fuzzy Set ranking. The FSC is designed for adaptive ranking and ranking fuzzy sets in groups by certain predetermined characteristics (i.e., modified single index ranking). The neural network features a simpler hardware implementation and a superior rank calculation speed than more conventional hardware implementations. The device is implemented in 1.5 micron Complimentary Metal Oxide Semiconductor (CMOS) technology, using about 20,000 transistors.

The FSC accomplishes limited "learning" by allowing input data streams to be stored in memory for subsequent use as comparison data.

TYPICAL APPLICATIONS AND DEVICE OBJECTIVES

Since many practical data are fuzzy, there are numerous potential applications for fuzzy data comparison. Most notable among these are text

and voice recognition, robotic control, security, surveillance, and Computer Aided Manufacturing (CAM) systems. A primary design goal of the FSC is compatibility with a broad range of applications. This includes applications where high data throughput, high speed processing, or both, are necessary. To meet these objectives, the FSC architecture provides expansion to allow simultaneous comparison of up to 256 fuzzy sets (Figure 1). In addition, we employ a high speed neural network post-processor whose processing speed is independent of the number of fuzzy sets being analyzed. Fuzzy set ranking analysis and control are accomplished through a microprocessor interface.

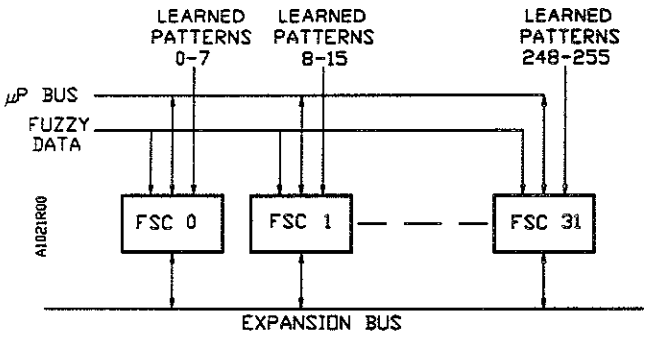


Figure 1. FSC System Expansion

The FSC does not preprocess or transform the input data in any way before performing the fuzzy ranking, though this may be required for certain applications. This elimination of preprocessing circuitry lowers device cost and helps maintain our compatibility goals.

DEVICE ARCHITECTURE

The FSC is composed of eight data comparison circuits, two neural networks, and circuitry to perform overhead functions (Figure 2). Each comparison circuit compares a "fuzzy" data set with a "learned" data set. Each comparison circuit generates an error value using one of two externally selectable comparison metrics. Once obtained, the error values are passed on to the neural networks.

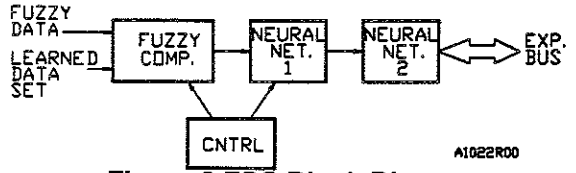


Figure 2.FSC Block Diagram

The ranking post-processor is configured as two tandem neural networks. The first network receives inputs from each of the eight data comparison circuits (Figure 3) and from the threshold register (ranking value). This network settles on the lowest value of the nine 16-bit inputs in two clock periods. (During the first clock time, the high bytes of the nine inputs are compared. The low bytes are compared during the second period. While slightly decreasing throughput, this approach was chosen to reduce circuitry.) The second neural network provides for expansion to other FSC devices. Comparison errors less than the ranking values

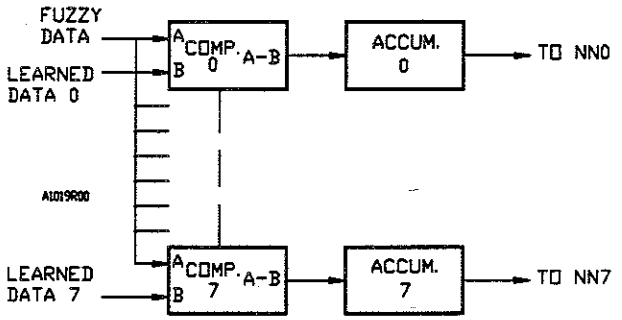


Figure 3. Comparison Circuit

contained in the corresponding FSC devices are compared on the expansion bus, which forms a part of the second neural network. The output, from as many as 32 FSC devices, settles on the lowest of the up to 256 total inputs within two additional clock periods. Arbitration circuitry is provided to handle tie conditions. This arbitration consumes one additional clock period. A solution is thus reached within five clocks from the time data are initially presented to the first neural network.

The FSC is bus-oriented and may be operated autonomously or under microprocessor control (Figure 4). Any single accumulator or control register from any FSC is addressable through a (4 bit) address bus and Chip Select control signal. Data may be moved to and from accumulator and control registers via an 8-bit data bus and the Read/Write control signal. Certain device operating conditions and alarms are signaled via an open-collector output interrupt signal.

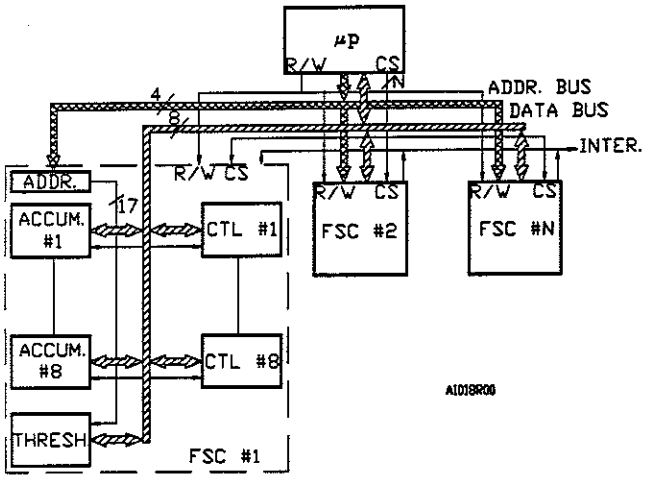


Figure 4. FSC Bus Structure

FUZZY COMPARATOR PERFORMANCE

Comparison of a "fuzzy" data set with a "learned" data set quantifies the fuzzy data. This defuzzification process is controlled by the ranking threshold (known as the α - cut) of the membership characteristic function μ associated with the fuzzy set (Kandel 1986). For those applications which require dynamic adjustment of this parameter the ranking threshold may be externally controlled in the FSC. A threshold value of zero would be used in ranking a non-fuzzy data set; hence, only exact matches will pass the comparison. The largest value of the ranking threshold in the FSC is 65535.

A number of metrics exist for ranking fuzzy digital data. In general, metric choice is application dependent, but two choices prevail (Linear and Hamming distance). The FSC comparator segments both the "fuzzy" and "learned" data into identical word lengths of *n*-bits ($1 < n < 8$). It then computes and sums the linear distance (L_1 norm) associated with *m* each *n*-bit words,

$$E = \sum_{i=1}^{m} | (FDW) - (LDW) |$$

where both *m* and *n* are externally selectable. FDW and LDW are Fuzzy and Learned Data Word, respectively, and E is the resultant error. The value for *m* may be any integer greater than or equal to 1. For example, if pixel data were being compared, *m* would be equal to the number of pixels in the image. Colors or grey levels, represented by bits/pixel, would determine the value of *n*. For applications where the Hamming Distance is the more appropriate metric, simply set *n* equal to 1.

NEURAL NETWORK DESIGN AND PERFORMANCE

Starting from the position that a logic element is a limiting form of a neuron, (Hopfield 1982; Widrow 1987), we constructed a three layer asynchronous competitive neural network from simple logic elements. The network contained in one FSC device consists of 128 neurons and solves the problem of identifying the lowest 8-bit value from up to eight error accumulators. (See Figure 5.)

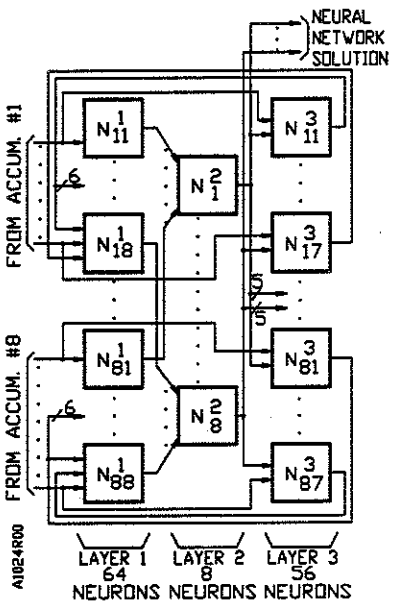


Figure 5. Neural Network Structure

Information from the accumulators feeds forward through the network, while feedback from layer 3 to layer 1 provides reinforcement or inhibition of the input data, as appropriate. Some feedforward of the input data directly to layer 3 also occurs. The lowest 16-bit word from among all accumulators in a particular FSC eventually settles at the layer 2 output.

To support applications involving multiple FSC devices (i.e., more than eight input data patterns), a second 3 layer neural network is attached to the output of the first network (Figure 6). The structure of these two networks is similar, and the second network identifies the lowest 16-bit accumulator result from a group of FSC devices. Connectivity between neurons across FSC devices in the second network is provided through an (open-collector) expansion bus.

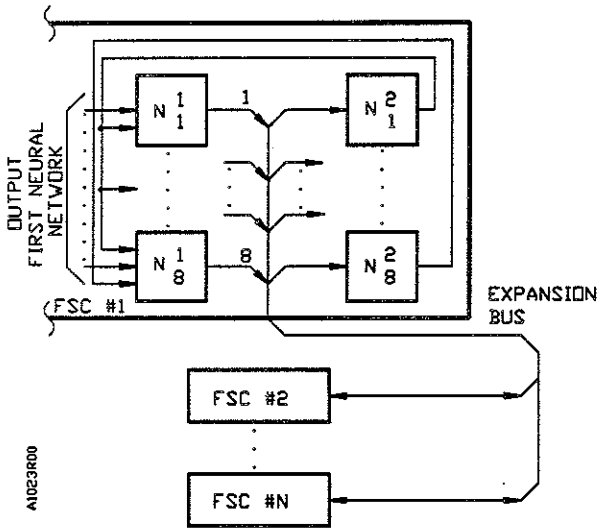


Figure 6. Neural/Expansion Bus control

The interconnection weights of the networks are fixed at 0 or +1, so the neural networks can perform only the tasks for which they were created, and cannot learn other tasks. The advantage of fixed interconnection weights for this application is the high speed parallel data processing which is achievable. Though employing positive feedback, simulation has shown these networks to be stable. As any of the 2^8 possible inputs may occur, the stable solution space for each neural network is an 8-dimensional hypercube, with stable solutions appearing at the corners of the cube.

The global neural network selects the minimum of up to 256 error values in five cycles of a 20MHZ clock, illustrating the parallel-processing nature of neural networks. The network also exhibits reduced hardware implementation relative to more conventional approaches. Each neuron of the network may be individually deselected, thus allowing fuzzy sets with a particular range of ranking values to be grouped.

CONCLUSION

A monolithic IC has been developed to aid in the ranking of fuzzy data sets, providing:

1) dynamic control of the ranking parameters
2) flexibility in choice of the comparison metric
3) simple expansion capabilities

A postprocessing neural network computes ranking comparisons rapidly, allowing the device to operate in high-speed applications. An IBM-PC compatible evaluation PC board with video interface and supporting software have been developed, aiding prototype evaluation and continuing research into fuzzy information processing.

REFERENCES

Hopfield, J., "Natural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proc. Nat Acad of Science*, Vol 79, April 1982.

Hopfield, J., "Neurons with Graded Response have Collective Computational Properties like those of Two-State Neurons", *Proc. Nat Acad of Science*, Vol. 81, May 1984.

Kandel, A. "Fuzzy Mathematical Techniques with Applications", Addison-Wesley Publishing, 1986.

Mabuchi, S., "An Approach to the Comparison of Fuzzy Subjects with an α - Cut Dependent Index", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 18, No. 2, 1988.

Widrow, B., R. G. Winter, and R. A. Baxter, "Learning Phenomena in Layered Neural Networks"., *Proceedings IEEE First International Conference on Neural Networks*, June 1987, pp.II-411 through II-429.

A Language for Knowledge Representation With Frames

Holmes S. Liao, R. C. Lacher
Department of Computer Science
Florida State University
Tallahassee, FL 32306

Abstract: *The result of a research project exploring features of a frame-based language, ALF, is presented. Several new perspectives for frame languages are proposed.*

Keywords: Frame-based Languages, Frame, Knowledge Representation, Demon, Reasoning

1. Introduction.

In this article we discuss a prototype knowledge representation language, A Language for Frame (ALF). Intended for the use of knowledge engineers in constructing frame-based expert systems, ALF is built upon a LISP environment and has a similar syntax to that of LISP, but ALF emphasizes particularly the power of frame manipulation.

A frame is usually described as a data structure which can be used to represent a stereotyped object, act, or event (Minsky 1975). In general, a frame can be interpreted as a set of slots which are filled with values. A frame-based knowledge base consists of a forest of frames.

In ALF's terminology, a *pool*, which is in the form of frame, is a global data structure that can be accessed by all modules. There are two types of data sub-structure in pool. One is called *frame pool*, the other *reasoning pool*. Frame pool keeps track of the status of all frames and sub-frames hierarchical relationship. Reasoning pool is used to deposit all the verified properties for each frame. The operation of reasoning pool is similar to that of truth maintenance system (Filman 1988). The frame pool and reasoning pool are each managed by a demon that is attributed to the pool and activated by all manipulation attempts.

2. Reasoning Capability.

ALF provides the capability of goal-directed reasoning, or backward chaining. For instance, on receiving the statement

(reason 'Frame)

the system searches the reasoning pool for properties in "Frame" that have not yet been confirmed and queries the user to request additional data. The search process is in a depth-first fashion.

The system user queries through the "reason" predicate. For example, a physician should be able to enter the name of a disease and then answer the questions associated with that disease. This could be awkward in practice because the system would in effect require the user to have all the knowledge in the knowledge base; that is, the physician would be expected to have immediate recall of all possible diseases. To get around this difficulty, ALF provides a predicate

(diagnose 'Kbase)

which effectively verifies all properties associated with all the frames in the "Kbase". This predicate effectively implements a forward- reasoning facility. In ALF, we experiment with self-adjusting methods of restructuring the knowledge base, wherein the frequency of a verified frame affects its order of appearance in the knowledge base. The more times a frame is verified, the closer the frame is to the beginning of the search path. (See also (Sleator and Tarjan 1986).)

Another predicate,

(match 'Frame 'Kbase)

is used to match each property in "Frame" against every property in every frame inside the "Kbase". The response to this predicate is a list of frames with a plausibility value in the range of the real interval **(0, 1]** indicating the percentage of properties in each frame inside the "Kbase" that have been matched with "Frame".

To provide an explanation facility, ALF makes use of the structured nature of frame to provide a predicate

(why 'Frame)

which can be used to explain the reasoning process for a certain frame. The explanation task is accomplished by showing all properties associated with that frame. Moreover, the frame's generic(s) and instance(s) are shown in depth-first order.

3. Procedure Attachment.

The knowledge in a frame is in declarative form. Attaching a procedure (or several procedures) to a frame can contribute to a resolution of the controversy between declarative and procedural knowledge representations (Winograd 1985).

In ALF, a procedure can be attached at either the slot level or the frame level. Once the value in

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

a slot (or in a frame), which is attached to a procedure, is accessed (or modified), the dormant procedure, which acts as a demon, could be activated to perform the predefined tasks. Message passing between frames can be carried out by demons. Therefore, a frame is called a "live" object; that is, the demon can be awakened once the precondition is satisfied. Through the use of the demon, the style of object-oriented programming can be achieved. Other usages of the demon include directing the execution sequence (or reasoning process) in frame-based systems and handling exceptions raised in a frame. For example, a demon attached to a frame can be used to trigger the execution of another frame group, thus changing the execution sequence.

The semantics of demon activation can be interpreted as

presumption → conclusion

where presumption is "touching" a slot which is attached to a demon, and conclusion is the execution of the attached demon. Here, "touching" means retrieving, modifying, deleting, or inserting values in the slot. We can perceive that this semantics is similar to that of production systems. This characteristic equips frame-based systems with the capabilities of the rule-based systems, that is, to preform reasoning.

Though LISP supports arbitrary attachment of procedures to lists, the procedure attachment in frame-based language should be more disciplined (Stefik 1979). The discipline of procedure attachment in ALF is that one must use the predicate

(set-demon 'Frame 'Property 'Procedure-name) or
(set-demon 'Frame 'Procedure-name)

explicitly to provide a standardized point of attachment in the system. The procedure will then act as a demon to guard the designated slot or the designated frame. Generally speaking, procedure attachment in ALF redraws the traditional boundary between program and data by giving unusual primacy to data structure. Nonetheless, the definition of the procedure to be attached is left to programmer.

4. Hierarchical Taxonomy.

ALF's specialization machinery is used to structure the frame into taxonomic lattices, a network that is used to represent the hierarchical relationships between frames. From the internal point of view, a frame can be sub-divided into slots that contain values. In addition, the language supports two other external acyclic hierarchical organizations. One is called *physical taxonomy*, the other, *inheritance taxonomy*. Theoretically, the number of layers in both taxonomies is not limited.

4.1. Physical Taxonomy.

The semantics of the physical taxonomy is essentially a *part-of* relationship. The physical taxonomy of frames is constructed by using

(set-subf 'Frame 'Property 'Sub-Frame)

predicate, where "Frame" and "Property" must have already been created before this predicate is executed. As for the "Sub-Frame", which is essentially another ordinary frame, it can be either have already been created or have not yet been created. The system automatically keeps track of the frames information by using the *frame pool*. The advantage of allowing a user to set pre-defined or un-defined subframe is that the user can build up a knowledge base in either top-down or bottom-up approach.

4.2. Inheritance Taxonomy.

The inheritance taxonomy represents *is-a* relationship. Though there could be many roles of inheritance, as that of Unit package (Stefik 1979), ALF provides two types of inheritance, *static inheritance* and *dynamic inheritance*. Acyclic inheritance taxonomy is enforced in both static and dynamic inheritance.

In ALF, a frame inherited by another frame is called a *generic*. A frame that inherits another frame (or frames) is called an *instance*. In a hierarchical frame-based system which is several levels deep (*e.g.* three or more levels), a *generic* frame can be an *instance* of another frame. The terminology, *generic* and *instance*, is borrowed from semantic networks. Both dynamic inheritance and static inheritance correspond to *is-a* links in semantic networks (Brachman 1983).

Considering the time factor, we declare that the static inheritance is time-dependent while the dynamic inheritance is time-independent. The reason is that the values inherited from static inheritance relationship depend upon the environment at the time when inheritance occurs. On the other hand, the effect of the dynamic inheritance, that is, the values in an instance shall change according to its generic, will last until the dynamic inheritance relationship is destroyed.

By employing a default mechanism, values in frames can have four levels of priority. That is, the dynamic inheritance has priority over static inheritance and default value has priority over dynamic inheritance. Ultimately, "normal" resident value has the highest priority. The effect of adopting multiple priorities can contribute to the resolution of the ambiguity of multiple inheritance (Touretzky 1986).

5. Summary.

ALF is a prototype higher level language for knowledge representation that explores several new issues: We distinguish physical taxonomy and inheritance taxonomy; the distinction is not explicit in any other frame-based systems. Four levels of value priorities are supported by ALF, hoping that this can resolve part of the complexity in multiple inheritance. Self-adjusting frame structure is used to increase the system's efficiency. A "traditional" demon mechanism, along with reasoning and explanation facilities which are common in most of the frame-based systems, are also included. The accommodation of production rules with frames makes ALF a powerful tool to construct knowledge-based expert systems.

In the next iteration of ALF, some enhancements would be appropriate. First, the ambiguity of multiple inheritance, which prevails in most of the semantic networks and frame formalism, is not solved in ALF. Second, the construction of frame objects is still too liberal. No solid methodology is devised to identify and construct frames. Lastly, to make ALF a real frame-based expert system shell, a user-friendly environment must be provided. This requirement will be more appropriate if we can shift ALF to a micro computer. Adding these capabilities would complete

ALF to a full knowledge engineering language.

Bibliography

Brachman, R.J. *What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks*, **IEEE Computer**, Vol. 16, No. 10, 1983, pp.30-36.

Filman, R.E. *Reasoning with Worlds and Truth Maintenance in a Knowledge Based Programming Environment*, **Communications of the ACM**, Vol.31 , No.4, 1988, pp.382-401.

Minsky,M. *A Framework for Representing Knowledge*, **Psychology of Computer Vision**, Winston,P. (Ed.), McGraw-Hill, New York, 1975.

Sleator, D.D., and Tarjan, R.E. *Amortized Efficiency of List Update and Paging Rules*, **Communication of the ACM**, Vol. 28, No. 2, pp. 202-208.

Stefik, M. *An Examination of a Frame-Structured Representation System*, **Proceedings of 8th International Joint Conference on Artificial Intelligence**, Tokyo, Japan, 1979, pp.845-852.

Touretzky, D.S. **The Mathematics of Inheritance Systems**, Morgan Kaufmann Publishers, 1986.

Winograd, T. *Frame Representation and Declarative / Procedural Controversy*, **Readings in Representation**, Brachman,R.J., Levesque,H.J. (Eds.), Morgan Kaufmann Publishers, 1985.

EXTENDING GRAPHICAL KNOWLEDGE REPRESENTATION LANGUAGES

Peter Creasy
Key Centre for Software Technology
Department of Computer Science
University of Queensland
St. Lucia Qld 4067
Australia

ABSTRACT

Representation languages which permit a graphical form are becoming widely recognized as a useful tool in the area of knowledge representation, the graphical forms permitting relationships which exist in the universe of discourse to be seen. The semantics that can be represented by most languages is generally fairly restricted, to a few common classes of integrity constraints, with no general mechanism to express semantics.

This paper examines a technique for extending the classes of semantics that can be graphically represented, shows how this can be applied to an existing knowledge representation language and suggests ways of extending this to other languages.

INTRODUCTION

Within AI knowledge engineering typically involves an informal approach to the formalization of the Universe of Discourse (UoD). There is often a large step between the UoD and the end result of data structures and associated procedures.

A large amount of research has been done within the information systems community in developing prescriptive methodologies for the information analysis task. This procedure, conceptual schema design, produces a formal description of the logical structure of the information of the UoD. This formal description, expressed by a conceptual schema language, can then be mapped to the appropriate data structures and procedures.

The intermediate form has the advantage of being well understood by the expert of the UoD and the database engineer. This intermediate representation, being independent of any data structures, can be mapped to a variety of database structures.

Typically, knowledge-based systems are (at least partially) fact-based and thus can take advantage of the techniques of fact-based methodologies and languages such as NIAM (Nijssen and Halpin 1989) and E-R (Chen 1976). The associated languages, however, are somewhat limiting in what can be represented. To overcome these problems NIAM has been extended (Creasy 1989) to permit a wider class of semantics to be represented.

Conceptual schema languages with a graphical form have become increasingly significant in information systems (e.g. Harel 1988,) as the importance of being able to see what the data structures are, becomes accepted.

NIAM has a graphical form and thus any extensions to the language should also have a graphical form. We have extended the language to have the power of FOPL, through existential graphs. We show that this concept can be extended to any fact-oriented language.

In the second section we discuss existential graphs. The next section discusses ENIAM, an extended version of NIAM. NIAM is informally described with an example and the extension is discussed. The extension of this idea to other fact-based languages is then shown.

EXISTENTIAL GRAPHS

Existential graphs were proposed by Peirce (1960) to represent logic and resulted from the work he did over a number of years. The graphs are simple with few concepts. They are one of many graphical representations for logic which have been proposed over the years, e.g. Venn diagrams (see Gardner 1983). The graphs use few operators (user symbols, lines and closures e.g. circles), are easy to read and write and offer the power of FOPL. Peirce proposed a number of what he called *parts* to the existential graphs. He called these the Alpha, Beta and Gamma parts. The parts of interest to us here are the Alpha part (equivalent to propositional logic) and the Beta part (equivalent to FOPL). Roberts (1973) has shown completeness and consistency for existential graphs. In particular Roberts uses Quine's (1955) ML logic system. Sowa (1984) has demonstrated the usefulness of existential graphs for representing FOPL.

The Alpha part has only three basic symbols: the sheet of assertion, the cut and the graph. The graphs are laid out on the *sheet of assertion* (SA), an arbitrarily large area, which equates to a model of the universe of discourse. A graph instance is something which asserts a possible state in the universe, viz. proposition. The sheet of assertion is also considered to be a

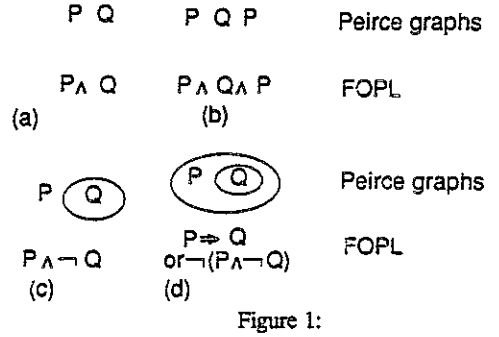


Figure 1:

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

c. 1989 FLAIRS 0-9620-1731-0/89/0400-0184

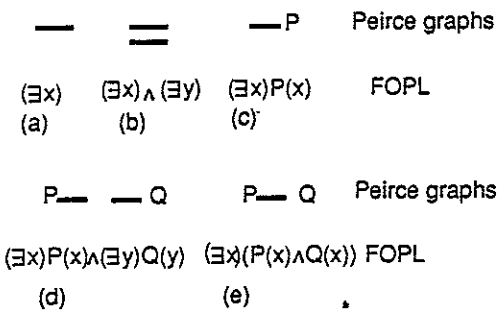


Figure 2:

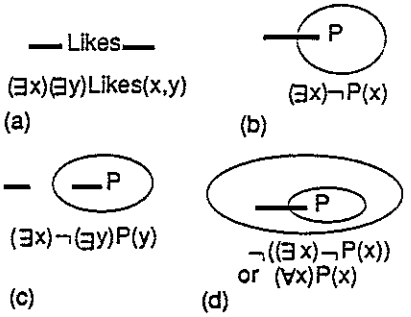


Figure 3:

graph, as the blank SA expresses whatever initially holds in the UoD.

Graph instances written on the SA are asserted to be true. By a graph is meant "any sign (symbol) which expresses in a proposition some state in the universe" (Roberts, p 32). Some examples are shown in figure 1. It should be noted that the shape of the graphs (e.g. whether lines are straight or curved or the shape of the closures) is not important. Figure 1(a) contains two graphs which together state the propositions represented by P and Q hold, i.e. the conjunction of these holds. Peirce distinguished graph and graph instance. In his terms there is strictly only one graph P of which there may be a number of occurrences (graph instances); e.g. we may have P repeated a number of times such as in figure 1(b). In the following we shall use the term graph rather than graph instance, except where we wish to make the distinction.

A single line, with no crossings, can be drawn around a graph (a cut) to indicate the negation of the graph. For example figure 1(c) is equivalent to $P \wedge \neg Q$. For material implication (e.g. $P \Rightarrow Q$, or "if P then Q") the term "scroll" is used. This example is shown in figure 1(d). The graphs are read from the outside inwards (*endoporeutic*). In this example (figure 1(d)) reading from the outside inwards we have the antecedent ("if P") and then the consequent ("then Q").

The Beta part additionally has a *line of identity*, used to represent an individual in the universe, and a *spot* which is equivalent to the FOPL predicate, e.g. P of figure 2(c). In the Alpha part the graph and the (propositional) symbol were one and the same. However in the Beta part a graph may consist of a number of symbols, hence the need to distinguish the symbol and the graph. The line of figure 2(a) indicates the existence of something, e,g, $(\exists x)$, while figure 2(b) indicates $(\exists x)(\exists y)$, i.e. two individuals which may be the same.

We still have the convention that two graphs drawn on the same SA represent the conjunction of the graphs. Thus figure 2(d) represents $(\exists x)P(x) \wedge (\exists y)Q(y)$. In figure 2(e) the line of identity between P and Q indicates that the two individuals represented by the ends are identical, i.e. $(\exists x)(\exists y)(P(x) \wedge Q(y) \wedge x = y)$ or $(\exists x)(P(x) \wedge Q(x))$. This can be extended to n individuals.

Just as a predicate can have more than one argument, more than one line may be attached to a predicate (symbol). The places of attachment are called *hooks*, and the meaning of the hooks must be understood just as the positions of the predicate arguments in FOPL need to be understood. Figure 3(a) represents *someone likes someone*. In this case we arbitrarily decided to consider the hook in front of the predicate to be the person doing the liking and the other hook to be the person liked.

It is possible to have any nesting of cuts. Graphs are referred to as being evenly enclosed or oddly enclosed, depending on the depth of nesting. This is significant as those graphs evenly enclosed have been negated an even number of times and thus have a positive sign (no negation) in the expanded FOPL (or propositional logic) form, while those oddly enclosed have a negation sign. Anything not enclosed (i.e. on the SA) is considered to be evenly enclosed.

When a line (of identity) crosses a cut the nesting of the cuts determines the equivalent FOPL variable scoping, with the outermost extremity of the line determining the position of the existential quantifier (the endoporeutic view). For example figure 3(b) is equivalent to $(\exists x)\neg P(x)$, i.e. the existence is at the outer level; figure 3(c) is equivalent to $(\exists x)\neg(\exists y)P(y)$; figure 3(d), where the outer part of the line of identity is oddly enclosed and P is evenly enclosed, is equivalent to $(\neg\exists x)P(x)$, i.e. $(\forall x)P(x)$, in this case the existence is at level 1.

A line of identity passing through an empty cut indicates the non-identity of the individuals at the extremities. For example figure 4(a) is equivalent to $(\exists x)(\exists y)(P(x) \wedge Q(y) \wedge x \neq y)$. The negation of a ligature indicates the non-identity of the individuals denoted by the extremities of the ligature.

We shall look at some examples to introduce a methodology for drawing the graphs and to compare the graphs with FOPL. As we have seen material implication is drawn with what Peirce called a scroll. Thus "if something is a P then it is a Q" is shown in figure 4(b). In this type of structure the antecedent is always oddly enclosed and the consequent evenly enclosed.

With only negation and conjunction we need these to represent disjunction. While this may appear awkward, we emphasize that the visual representation is important and one can quickly learn to recognize and draw this picture as a disjunction.

Constants are not treated differently from other symbols, which we have so far thought of as predicates. We treat a constant the same as a unary predicate. If necessary, we can distinguish constants by enclosing them in quotes. However for the usage with NIAM, as we shall see, this is not necessary as there is no ambiguity.

ENIAM

NIAM is a methodology which involves taking a significant set of sentences from the UoD and from these abstracting to obtain the fact types.

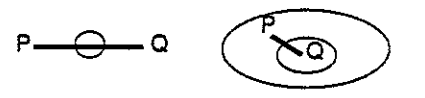


Figure 4:

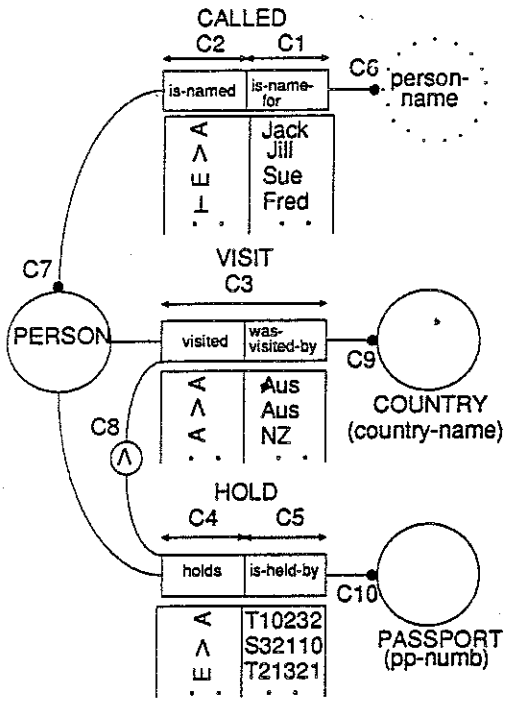


Figure 5:

The associated NIAM language is a fact-based language with a graphical representation. An example is shown in figure 5. Part of the population of the significant sample is shown, however this is usually only done either for explanatory purposes or to check the schema. The label types (PERSON-NAME and COUNTRY-NAME) are represented as dotted circles. The current extension of the label type PERSON-NAME is Jack and Jill. The entity types (PERSON and COUNTRY) are represented as circles. The current extension of PERSON is those people called Jack and Jill. These entities are non-lexical. We have represented them by arbitrary lexical markers. Often in the diagrams we use the corresponding lexical representation (e.g. Jack).

A relationship type between an entity type and a label type is a reference type. A relationship type between entity types is a fact type. Each fact or reference type consists of a number of roles. These describe the part (or role) each entity or label type (to which the role is joined) plays in the fact or reference type. For example the role played by PERSON in CALLED is "is-named". Reference types give the naming conventions to be used in lexically identifying entity instances. Every entity type must be linked directly or indirectly to a label type (or types) such that every entity has a unique reference.

Some of the constraints we can represent in NIAM are intrafact (and intrareference), interfact uniqueness, mandatory role, equality, exclusion and subset constraints. The arrows above the roles represent intrafact or intrareference uniqueness constraints which are placed on the instances of the role or roles. They indicate that the role or roles uniquely identify a single fact instance. In this case constraint C1 indicates that each PERSON-NAME is represented only once in the reference type CALLED and C2 that each PERSON is only represented once, viz. each person has (only) one unique name. Constraint C3 indicates that a person can have visited more than one country and vice versa, i.e. the only constraint is that there be no repetitions of any combination of person and country.

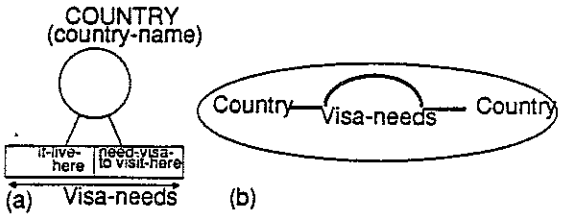


Figure 6:

The constraint C7 is a total role constraint which indicates that if an entity (or label) takes part in any fact (or reference) instance then it must take part in an instance of the fact (or reference) type corresponding to the constraint. The subset constraint C8 indicates that anyone who visits a country must have a passport.

Rather than represent the references as we did with the reference type CALLED, we can simplify the diagram by writing the corresponding label type in parentheses following the entity. We have shown this for the entity type COUNTRY and used the country name in the fact type VISIT.

For figure 5 it is a comparatively simple matter to write down the corresponding relation schema. In this case it will be VISIT(PERSON , COUNTRY). As each 'column name' in the relation schema must correspond to a label type, these strictly would be PERSON-NAME and COUNTRY-NAME.

The Extended Language

NIAM constraints are set oriented. In terms of sets, the existential graphs are "member oriented" in that we express in an exclusion constraint, say, that there is no element that is both sets involved in the constraint. This potentially gives greater scope for expressing constraints. For example, given the NIAM schema of figure 6(a), we may wish to express that *one does not need a visa to visit ones own country*. We cannot express this with NIAM's set constraints as a country will generally appear in both roles, i.e. in both sets, and we thus can't use the exclusion constraint. However existential graphs do allow us to express "there cannot be a country which acts as a lives-in and a requires-visa-for in the same row". This is shown in figure 6(b). The combination of the NIAM constraints and the existential graph constraints is thus potentially powerful and simple.

To be useful as a constraints language the logic first needs to be extended by introducing "inequality" predicates ($<$, $\leq$, $\geq$, $>$, $\neq$) for numeric values.

One of the advantages of a graphical language is allowing a user to see immediately the structures (e.g. relationships between object types). Over-complication of a diagram defeats this advantage; thus it is important to keep the number of graphical symbols to a minimum. By using existential graphs it permits the full power of FOPL to be used to express constraints with few symbols. We could use set constraints of NIAM when appropriate, i.e. to express what we shall call our "specialty" constraints (uniqueness, mandatory role etc.), and existential graph constraints when membership constraints are required or NIAM does not have the appropriate symbol.

We shall use figure 7 as our example in demonstrating how we can expand NIAM's constraints. As most of the examples don't apply to all of figure 7 we shall only present the relevant sections in the examples. In figure 7 people are scheduled to visit countries. Each country has certain vaccination (immunisation) requirements and each person is recorded as having certain vaccinations. It is required to ensure that people

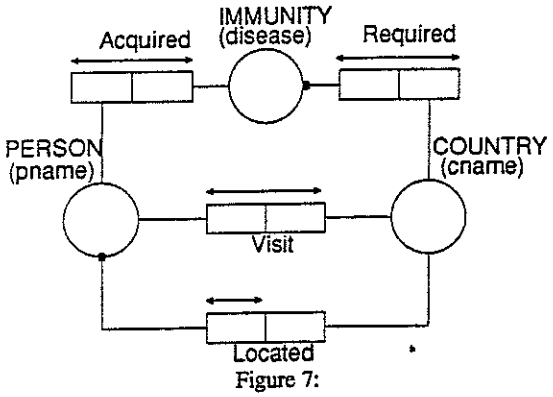


Figure 7:

have the necessary vaccinations to visit the countries they have been scheduled to visit.

Each of the fact types and entity types can be considered as a predicate. We need typing predicates in FOPL (as in figure 6(b)). However, as NIAM is a strongly typed language we have implicit typing of the fact type predicate arguments; e.g. with the Acquired fact type predicate, for $(\exists x)(\exists y)$Acquired(x,y) we have the implicit typing Person(x) $\wedge$ Immunity(y).

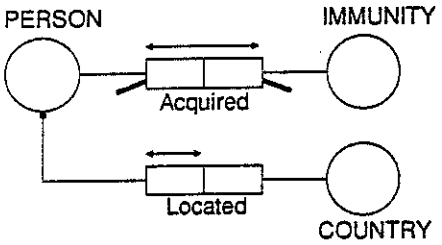


Figure 8:

The existential graphs can be incorporated directly into NIAM. We use the fact type symbol as the predicate. For example, if we wish to express the constraint that *there must be someone who has a vaccination* we use the fact type Acquired as an existential graph predicate as in figure 8.

As a further example suppose we have the subset constraint *anyone visiting a country must have a have a vaccination* we express this as the embedded existential graph in figure 9. However, if we use the fact types as predicates directly the diagram rapidly becomes unmanageable. For practical reasons we are excluded from using further instances of the predicates.

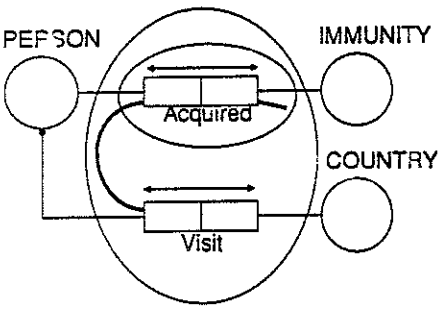


Figure 9:

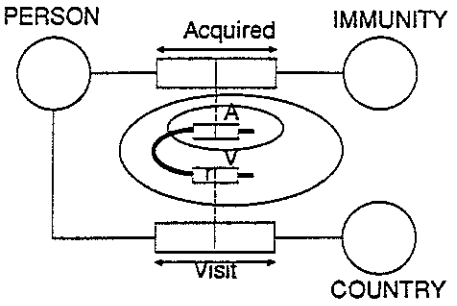


Figure 10:

We thus also propose taking a "copy" or "image" of the relevant fact types and use those as the predicates in an existential graphs diagram. The above constraint can then be rewritten as in figure 10. The image A is an "image" of the fact population Acquired (what we have previously called a predicate instance).

In the embedded existential graphs we explicitly represent constants. For example *all visitors from Australia have a Polio vaccination* would be represented as in figure 11.

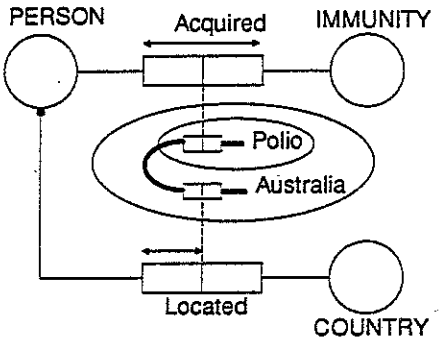


Figure 11:

Often we need constraints involving more than 2 fact types, in particular those involving a join. For example we may require that *people visiting a country must have the necessary vaccinations*, viz.

(Visit * Required)[person,immunity] $\subseteq$ Acquired.

We can express this in ENIAM in a more natural manner: *if a person is visiting a country which requires vaccinations then the person must have those vaccinations*. Using the scroll we express this in ENIAM as in figure 12.

We can also use the existential graphs to represent derivation rules. Where rules are of the form "consequent if antecedent",

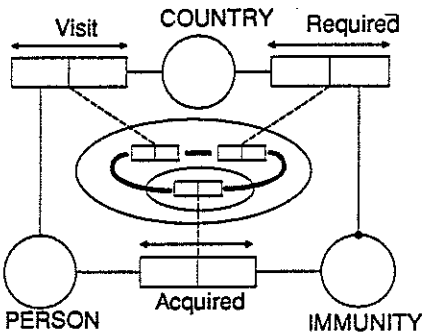


Figure 12:

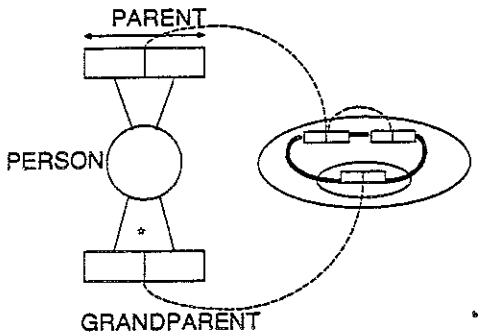


Figure 13:

e.g. *x is the grand-parent-of z if x is the parent-of y and y is the parent-of z* then we can use the scroll. This particular example is shown in figure 13.

It would be desirable to be able to represent an element of an entity type, i.e. to have the equivalent of unary type predicates. In this case we express the image of an entity type as a large blob. The constraint of figure 14 indicates that there exists at least one person in the information base (which does lead to problems in not permitting us to have an empty information base).

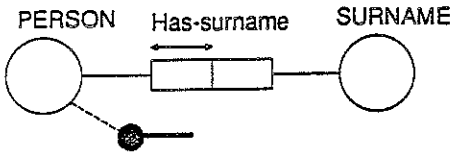


Figure 14:

Given this construct we have a representation for the mandatory role. We represent *every person has a surname* as in figure 15. This type of constraint is useful for expressing subtype definitions.

EXTENDING FACT-BASED LANGUAGES

We can use existential graphs to extend the graphical form of any fact-based language. You can take your favorite graphical knowledge representation language and use the relationship type as the relationship predicate. If the language is strongly typed (as most such languages are), the same approach can be taken as we did with NIAM.

In particular the entity-relationship model (Chen 1976) which is widely used is well suited for such an extension. E-R's relationship type and entity type are treated in the same way as NIAM's fact type and entity type. We simply use differently shaped image symbols to correspond to those of E-R.

We can express the example *people visiting a country must have the necessary vaccinations* of figure 12 as the extended E-R diagram of figure 16.

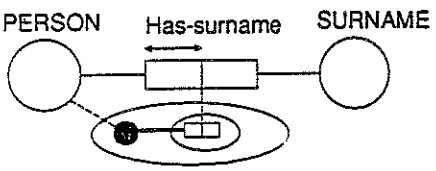


Figure 15:

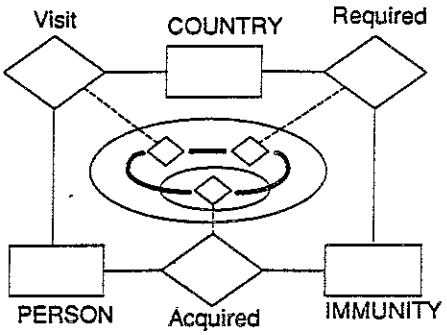


Figure 16:

SUMMARY AND CONCLUSIONS

Existential graphs have been shown to provide a simple, easy to read graphical representation for first order logic. While, for complex expressions, they may be difficult to read, the same could also be said of the usual Peano-Russell notation. Most expressions though are relatively simple, and of a standard form, involving integrity constraints. We have shown that it is comparatively simple to extend any fact-based language to include existential graphs.

We have shown elsewhere (Creasy 1988) that a technique suggested by Nicolas (1982) can be adapted to permit efficient computation of most constraints expressed with the graphs.

REFERENCES

Creasy, P.N. (1988) "Extending Graphical Conceptual Schema Languages", Internal Report, University of Queensland.

Creasy, P.N. (1989) "ENIAM: A More Complete Conceptual Schema Language," Technical Report No. 107, University of Queensland, February 1989.

Gardner, M. (1983) *Logic machines and Diagrams*, (first published 1958), The Harvester Press, Brighton.

Harel, D. (1988) "On Visual Formalisms", *Comm ACM* 31, 5 (May), pp 514-530.

Nicolas, J-M. (1982) "Logic for Improving Integrity Checking in Relational Data Bases," *Acta Informatica* 18, pp 227-253.

Nijssen, G.M. and Halpin, T.A. (1989) *Conceptual Schema and Relational Database Design: A Fact-Based Approach*, Prentice Hall.

Peirce, C.S. (1960) *Collected Papers of Charles Sanders Peirce*, vol 4, A.W. Burks (Ed.), Harvard University Press, Cambridge, Mass.

Quine, W. V. (1955) *Mathematical Logic*, (revised edition) Harvard University Press, Cambridge, Mass.

Reiter, R. (1988) "On Integrity Constraints," *Proccedings 2nd Conference of Theoretical Aspects of Reasoning about Knowledge*, Pacific Grove, California, March 7-9, pp 97-111.

Roberts, D.D. (1973) *The Existential Graphs of Charles S. Peirce*, Mouton, The Hague.

Sowa, J.F. (1984) *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Weseley, Reading, Massachusetts.

AUTOMATIC REPRESENTATION OF KNOWLEDGE IN MANUFACTURING PROCESSES

Carlos Segami
Computer Science Department
University of Central Florida
Orlando, Florida 32816

Ying-Kuei Yang
Expert Systems Group, CIM R&D
Harris Semiconductor
P. O. Box 883, MS 62-64
Melbourne, FL 32901

ABSTRACT

The need to automate manufacturing processes has made it desirable that manufacturing knowledge be kept in a database in such a way that the database system may be used to command the manufacturing mechanisms. Since conventional database models have been found to be inadequate for the representation of manufacturing information, a knowledge representation mechanism is proposed in this paper, which addresses the demands of the manufacturing domain. At the same time, the components of a system are described, which is able to read the English description of a process and build from it the proposed knowledge representation structures.

Key Words: Knowledge Representation, Process, Manufacture.

1. INTRODUCTION

Because of the requirements of data automation and integration (Simon, 1982; Groover, 1984), knowledge about manufacturing processes must be kept in a database system and it must be related to other pertinent knowledge so that the database system can be used to command the manufacturing mechanisms. We find, however, that record oriented conventional data models, which were originally developed for information processing in the business domain, are not adequate for the representation of manufacturing knowledge (Brodie, 1984; Kent, 1979; Su, 1986; Yang, 1988). The nature of knowledge in the manufacturing domain is very different and more complex than that of the business domain, which results in quite different needs with regard to modeling. For example, knowledge in the manufacturing domain consists mainly of collections of interrelated *events*, rather than collections of *records* with identical structures. These events, furthermore, are generally required to occur within given time constraints. Record-based models are so rigid that their modeling power is too limited to allow the whole semantics of manufacturing data to be directly represented in a database system. It is clear, then, that a more powerful approach for the representation and organization of manufacturing knowledge needs to be developed, and that appropriate tools must be provided so that manufacturing engineers can easily store this knowledge in a database.

In this paper, we propose a knowledge representation mechanism that addresses the demands of the manufacturing domain, and describe the components of a system that builds the representation structures of a manufacturing process from an English description of it. The knowledge representation structures proposed in this paper are similar to the structures introduced in the SNOWY system (Gomez, 1985; Gomez and Segami 1987, 1989a, 1989b). SNOWY is a system which understands scientific texts in the general domain of biology. It reads short paragraphs, assimilates the knowledge in them, and answers questions about the text it has read. When SNOWY starts reading new texts, it has very limited knowledge or no knowledge at all about the contents of the texts, something that differentiates it from other models of comprehension. Comprehension in SNOWY is viewed as a process involving the phases of *parsing*, *formation*, which takes the parser output and builds the internal knowledge representation structures; *recognition*, which identifies the concepts and relations that are already known to the system, and *integration*, which inserts in memory the new concepts and relations underlying the text just read. The problem we deal with in this paper is that of reading the English description of a manufacturing process, description which consists of sentences like the one shown in Figure 1 below, and building from it the representation structures corresponding to the process.

Insert the shaft into the main hole of the housing body using robot-1 so that the shaft head goes through the top hole of the housing body

Figure 1. English Description of an Event in a Manufacturing Process

Since knowledge representation structures have to be built from an English text, a phase similar to SNOWY's *formation* phase is needed. Once a representation structure has been built, it has to be appropriately integrated with the rest of the structures representing the process. Thus, an *integration* phase is also needed. Since we are mainly interested in the representation of a specific kind of processes, the type of integration needed will be somewhat different from SNOWY's, which does deal with processes. Thus, the system we propose consists of the phases of parsing, formation and integration. We will not discuss parsing in this paper. The parser we use in our prototype is WUP: A Word Usage Parser (Gomez, 1988). WUP is a syntactic parser which transforms a sentence into a case frame structure. For example, the sentence "red blood cells carry oxygen from the lungs to all parts of the body" is transformed into the structure below. The output of the parser becomes the input to the *formation* phase.

```
(subj ((adj red) (noun blood) (noun cells)) verb (carry)
 obj ((noun oxygen)) prep (from ((dfart the) (noun lungs)))
 prep (to ((adj all) (noun parts))) prep (of ((dfart the) (noun body))))
```

In section 2, we describe the types of structures used in the representation of processes, in section 3 we discuss how these structures are built from the parser output, and in section 4 we discuss the integration of new representation structures with already existing ones. Finally, the conclusions are given in section 5.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

2. REPRESENTATION OF MANUFACTURING PROCESSES

We view a manufacturing process as a collection of events connected to each other through temporal relations. Each event corresponds to an individual action to be performed during manufacture, and can be perceived as consisting of a primitive relation (the specific action to be performed), a number of arguments (the objects to be manipulated, how they will be manipulated, etc.), and some conditions that complete the definition of the event. Because the events in a manufacturing process must take place in a predetermined order, events will be related to each other through temporal relations such as *precede*, *concurrent* and *simultaneous*. Events are represented through *event-structures*, which are structures similar to the *b-structures* in (Gomez & Segami, 1987). Figure 2 shows an example corresponding to the text in Figure 1. The *actor* slot must be filled with an object capable of performing an action. These include the animate beings and other objects such as robots. The slot *pr* contains the primitive relation of the event, and the slot *condition* contains one or more relations that must be satisfied in the performance of the event. A second type of representation structures used in our system are *object-structures* (Gomez, 1985; Gomez & Segami, 1989a). These are frame type structures that are used to represent physical or abstract objects. They consist of slots corresponding to descriptive relations or to relations that attribute actions to physical objects. Figure 3 shows the *object-structure* corresponding to the object *shaft*. Each of the arguments of a given event has an *object-structure* associated with it. Thus, corresponding to the event shown in Figure 2, there will be *object-structures* for robot-1, shaft, housing-body-main-hole, shaft-head, and housing-body-top-hole which contain knowledge about these objects pertinent to the process. *Object-structures* are kept in memory connected to each other through *is-a* relations, which causes memory to have a hierarchical organization. When the system begins reading the description of a process, it accesses the knowledge in this hierarchy as it builds the representation structures. Figure 4 illustrates the organization of some *object-structures* in memory, and Figure 5 illustrates the *part-of* relation between some of the objects in the example of Figure 6.

```
(e1
   (actor (robot-1))
   (pr (insert))
   (object (shaft))
   (destination (housing-body-main-hole))
   (condition
      (shaft-head (goes-through (housing-body-top-hole))))
)
```

Figure 2. An Example of an Event-Structure

```
(shaft
   (is-a (physical-thing))
   (part-of (touch-sensor-top))
   (has-part (shaft-head))
   (weight (28))
   (hardness (30))
   (length (20))
)
```

Figure 3. Object-structure corresponding to the object shaft

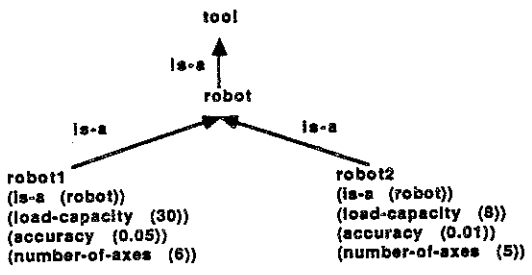


Figure 4. Organization of some object-structures in memory

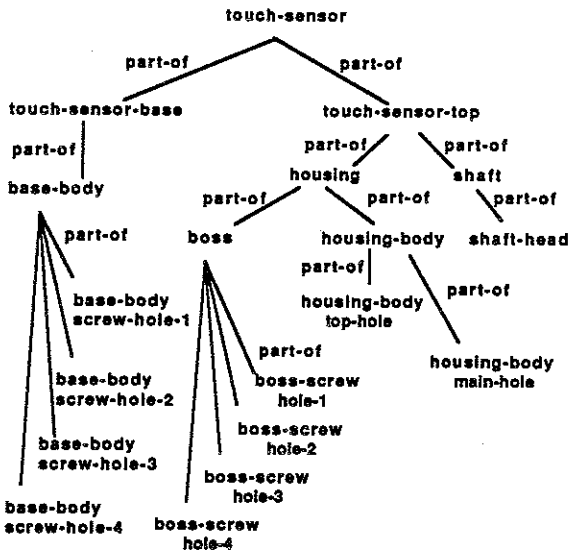


Figure 5. Part-of relations between some objects in memory

In summary, the representation of a manufacturing process consists of (1) a network of *event-structures* and, (2) a collection of *object-structures* corresponding to physical objects relevant to the process.

3. FORMATION

In this section we give an overview of the steps taken by the Formation phase. Once an input sentence is parsed, the Formation phase transforms the parser output into the representation structures of the event or events introduced by the sentence. When doing this, the system accesses information from several sources, which include the *object-structures* in memory, the definitions of the primitive relations, and *formation* rules associated with the primitives (Gomez, 1985). Primitive relations may be either *descriptive* or *action* primitives. Since each event in a manufacturing process is associated with an action primitive, when the system reads statements whose verbal primitive is an action, it builds an *event* structure. When the verbal primitive of a sentence is descriptive, the system will simply add the corresponding relation to the pertinent *object-structures* in memory. What structures to build and how their slots will be filled is determined by the *formation* rules associated with the primitives.

Each primitive has a definition similar to:

```
(insert
   (actor (agent))
   (object (physical-thing))
   (source (place))
   (destination (place))
   (tool (physical-thing))
   (condition (relation))
)
```

These definitions indicate what semantic features are required of each slot filler in the corresponding *event-structure*. Since in a manufacturing process all parts required in the process are known before hand, we assume that when the description of a process is being read, all objects referenced in it have their corresponding *object-structure* already present in memory. The system, however, is able to read the definition of a new object and to build and insert its *object-structure* in memory.

Given a parsed sentence, the main steps in the Formation phase are the following:

1. Identify the concept denoted by the subject of the sentence. Here, the Formation phase examines the noun phrase in the subject of the sentence and identifies the concept in memory that corresponds to it.

2. Identify the concept denoted by the object or predicate of the sentence.

3. Determine the primitive relation, and retrieve its definition and formation rules.

4. Build the structures as indicated by the formation rules.

5. Examine the prepositional phrases and subclauses in the statement and fill in the corresponding slots in the *event-structure*.

The *formation* rules associated with the primitives determine what structures must to be built. Rules associated with descriptive primitives direct the system to either create a new *object-structure*, when the definition of a new object is being introduced, or to add a slot to an existing *object-structure*, when a new relation is predicated of an existing object. Similarly, rules associated with action primitives direct the system to build the *event-structure* corresponding to a step in a manufacturing process. Building an *event-structure* involves the identification of the arguments of the event. For example, if the event is an instance of *insert*, then the actor, object, destination, and conditions, if any, need to be determined from the text. This is done with the help of the definition of the primitives, which includes the semantic features expected of each of the arguments. Each *event-structure* built by the Formation phase is identified with a gensym, and it is up to the Integration phase to appropriately link a new *event-structure* with the rest of the structures representing the process. Figure 6 contains the definition of an assembly process consisting of seven steps. After each sentence is read, the Formation phase builds the corresponding *event-structure*, which is saved until the user provides information that determines what temporal relations link the new event to previous events. Handling this information is the task of the Integration phase.

4. INTEGRATION

The Integration phase is concerned with two tasks. When new concepts or relations are introduced by the text, the integration phase makes sure that they are inserted in the correct place in memory. Similarly, when a new event in the manufacturing process is introduced, it is up to the Integration phase to appropriately link this event with other events in the process. The first task is discussed at length in (Gomez and Segami, 1989a). In this section we describe the integration of new events into the representation of a process. What this involves is the determination of the temporal relations existing between a newly formed event and those already existing in the process.

If e1 and e2 are two events, we say that they are in the relation "e1 precede e2" when during manufacture the step corresponding to e1 must be performed before the step corresponding to e2. Similarly, we say that "e1 concurrent e2" when e1 and e2 correspond to manufacturing steps that may be performed at the same time, while the relation "e1 simultaneous e2" means that e1 and e2 must be performed simultaneously. Other temporal relations may be needed as different texts are studied. After the representation of an event is built, the Integration phase examines the input sentence to determine if a temporal relation is defined by it. Suppose, for example, that the following passage is read:

> Insert the shaft into the main hole of the housing body. Next, insert the rod into the main hole of the housing body.

When the second sentence is read, the Integration phase determines that the first event must precede the second one, and proceeds to integrate the second one accordingly. Similarly, when the sentences below are read, the Integration phase determines that the two events must take place simultaneously.

Insert the shaft into the main hole of the housing body using robot-1, so that the head of the shaft goes through the top of the housing body.

Insert the rod into the main hole of the housing body using robot-1, so that the body of the rod is inside the housing body.

Insert the tail of the rod into the main hole of the base boss using robot-1, such that the housing boss is adjacent to the body of the base and such that the screw hole 1 in the body of the base is aligned with the screw hole 1 in the housing boss.

Use robot-2 to screw the screw hole 1 in the body of the base with the screw hole 1 in the housing boss using a hexagon head cap screw.

Use robot-3 to screw the screw hole 2 in the body of the base with the screw hole 2 in the housing boss using a hexagon head cap screw.

Use robot-4 to screw the screw hole 3 in the body of the base with the screw hole 3 in the housing boss using a hexagon head cap screw.

Use robot-5 to screw the screw hole 4 in the body of the base with the screw hole 4 in the housing boss using a hexagon head cap screw.

Figure 6. A Process Consisting of Seven Manufacturing Steps

> Screw hole 1 in the base with hole 1 in the housing boss using a hexagon head cap screw. At the same time, screw hole 2 in the base with hole 2 in the housing boss using a hexagon head cap screw.

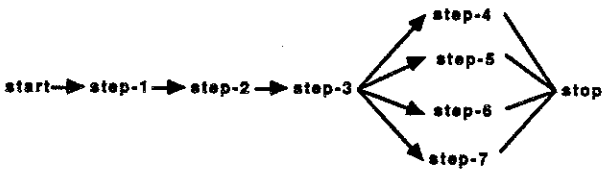
Alternatively, the temporal relations may be specified by the user in separate sentences, as illustrated below.

> Insert the shaft into the main hole of the housing body. Insert the rod into the main hole of the housing body. Step 1 precedes step 2.

Thus, if the system reads the definition of Figure 6, it will build the *event-structures* corresponding to the seven steps, but it will be unable to determine the temporal relations between the events. To complete the definition of the process, the user might include the following:

> Step 1 precedes step 2. Step 2 precedes step 3. Step 3 precedes step 4, step 5, step 6, and step 7. Step 4, step 5, step 6, and step 7 are concurrent.

With this definition, the events in the process will be organized as illustrated below.



5. CONCLUSIONS

The representation structures we have introduced can be thought of as a high level specification of a process, which can be transformed by an interpreter into low level specifications with enough detail to be recognized by manufacturing tools such as robots. This system can be particularly useful in manufacturing processes in the following ways:

(1) It can adequately represent every piece of manufacturing knowledge, including events and precedence relations, which would be difficult for conventional data models to represent. With the proposed data model, the database system can command and provide complete information to the manufacturing mechanisms.

(2) It is a user friendly system. Manufacturing engineers may interact with the system in plain English. The system is powerful enough to take an English sentence as input, transform it into the appropriate knowledge representation structures, and integrate them with existing knowledge.

(3) The representation relates all the information required for an event, a process or a product, which makes it much easier for an engineer to propagate the changes when required. This feature is particularly useful in the manufacturing domain which has more complex relationships among entities than conventional data processing domains.

The proposed system may be enhanced with other components, particularly, a question answering component to allow user to query the system in various ways.

REFERENCES

Brodie, M. L., Mylopoulus, J. and Schmidt, J. W. (1984). *On Conceptual Modeling,* Springer-Verlag, New York, NY.

Gomez, F. (1985). A Model of Comprehension of Elementary Scientific Texts. *Proceedings of the Workshop on Theoretical Approaches to Natural Language Understanding.* Halifax, Nova Scotia.

Gomez, F. (1988). WUP: A Parser Based on Word Usage, in *Proceedings of the 1988 IEEE Conference on Computers and Communication,* Scottsdale, Arizona, 445-449.

Gomez, F. and Segami, C. (1987). SNOWY: A System Which Understands Elementary Scientific Texts. Technical Report CS-TR-87-04, Dept. of Computer Science, University of Central Florida.

Gomez, F. and Segami, C. (1989a). "The Recognition and Classification of Concepts in Understanding Scientific Texts," *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 1, 1989.

Gomez, F. and Segami, C. (1989b). Classification-Based Inferences in Retrieving Information from a Database of Scientific Facts. To appear in *Proceedings of the Fifth IEEE Conference on Artificial Intelligence Applications*, Miami, Florida.

Groover, M. P. and Zimmers, E. W. (1984). *CAD/CAM: Computer Aided Design and Manufacturing*, Prentice-Hall, New York, NY.

Kent, W. (1979). "Limitations of Record-Based Information Models," *ACM Transactions on Database Systems*, Vol.4, No. 1, pp 107-131.

Simon, R. L. (1982). "CAD/CAM: The Foundation of Manufacturing Automation," in Proceedings of the Conference on CAD/CAM Technology in Mechanical Engineering, MIT, Cambridge, MA, pp. 221-235.

Su, S. Y. W. (1986). "Modeling Integrated Manufacturing Data with SAM*," *Computer*, Vol. 19, No. 1, pp 34-49.

Yang, Y. K. (1988). "The Semantic Network and the Representation of a Manufacturing Database," in Proceedings of the First Florida AI Research Symposium, Orlando, FL.

Representing Event Identity in Semantic Analysis: A Network Approach

Joseph M. Marchetti Robert A. Morris

Florida Institute of Technology
Dept. of Computer Science
150 West University Blvd.
Melbourne, FL 32901

Keywords: Temporal Knowledge Representation

1 Abstract

When an intelligent(?) being pulls a trigger, thereby killing someone, how many actions has it performed? The answer to this question has important consequences for a theory of event representation. In this paper we argue for an approach which distinguishes between an event occurrence and its possible descriptions. We motivate our approach by pointing out that inferences commonly made about performing actions *by* performing others is simply and cogently explained by it. We contrast our view with that of James Allen, who suggests that no distinction between a fully instantiated propositional form describing an event, and the event itself, ought to be made.

2. Statement and Significance of Problem

> I am inclined to think that no account of what it is to do something by doing something else will be satisfactory unless it makes room for, and indeed explains, those things that make us want to say that for a man to do something by doing something is for his doing one thing to be identical with his doing of the other, and in particular, that for Sirhan to have killed Kennedy by shooting him is for the shooting to have been identical with the killing. [8]

> When I say "I name this ship the *Queen Elizabeth* I do not describe the christening ceremony, I actually perform the christening; and when I say "I do" (sc. take this woman to be my lawful wedded wife), I am not reporting on a marriage, I am indulging in it. [5]

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

> I turn last to Kim's remark that it is not absurd to say that Brutus' killing of Caesar is not the same as Brutus' stabbing Caesar...We are inclined to say *this* stabbing might not have resulted in a death, so how can it be identical with the killing? Of course the death is not identical with the stabbing; it happened later. But neither this nor the the fact that not some stabbings are not killings shows that this particular stabbing was not a killing. Brutus' stabbing of Caesar did result in Caesar's death; so it was in fact, though of course not necessarily, identical with Brutus' killing of Caesar. [6]

Intelligent natural and artificial systems which plan and make decisions must learn and maintain the necessary information to make decisions and act on the basis of beliefs and desires. A robot, for example, may be given the command: "Hide block A". Given the right set of beliefs, it may be able to realize that putting block A inside box B is tantamount to doing the requisite action. A sketch of the reasoning process is summarized in the following inference:

1. I want to hide block A.
2. By putting block A inside box B, I am hiding block A.
3. Therefore, I will put block A inside box B.

Our focus in this paper is on the meaning representation of sentences like (2). In particular, an issue arises in the analysis of such sentences regarding the number of occurrences of events being described. We tend to agree with the philosophers that a proper account of intentional behavior is impossible without attempting to settle the problem of identifying and counting events. A cogent solution to the problem we are about to describe in more detail, will have significant ramifications for the representation of semantic information about events and their relations.

The problem may be restated in the form of a common sense query: if by pulling the trigger of a gun, I shoot someone, how many actions have I performed?

The answer to this query has implications for the semantic interpretation of English sentences such as the following:

- By stabbing Caesar, Brutus killed him.
- Sirhan fatally shot Kennedy.
- John followed Mary by walking a few feet behind her.
- Phyllis broke the bottle, thereby christening the boat.

There are sound reasons, we hold, in adopting what we'll call the *Davidsonian* solution to this problem. This solution can be formulated as follows:

> The proper semantic interpretation of sentences of the form, *By doing X, S did Y* is to maintain that there is a single event e such that e is the doing of X and the doing of Y. Consequently, e can be viewed as one event with possibly many descriptions.

This view implies drawing a distinction between the (occurrence of) an event and its description or representation. In this paper, we shall do the following:

1. Lay the theoretical foundation for a Davidsonian solution to the problem of event and action identity.
2. Justify the solution on logical grounds.
3. Compare the solution to the dominant opposing view, held by Goldman [7] and Allen [1]. Such a view holds that for each fully instantiated event representation there corresponds a unique occurrence. Hence there is no need (for a theory of meaning) to draw a distinction between event occurrences and their fully instantiated representations.
4. Describe a modification of the semantic network mechanism which allows for the interpretation of English sentences such as the ones listed above, based on our model.

2. Background

Over the past few years it has become increasing obvious to researchers in AI, database theory and software engineering that temporal logic can be used as a valuable formal tool. In AI, the characterization of events, change and reasoning about time is crucial. Systems that process text and engage in dialogue, plan, predict, and explain, must be cognizant of the role of time in accomplishing their tasks. This interest in action and event theory has sparked a radical change of focus of theories of meaning and representation from a focus on non-temporal objects and properties, to event types and their occurrences.

It is clear to anyone who has tried to debug a computer program that employs concurrency that reasoning about events offers difficult challenges. Our beliefs about events are non-monotonic: further information may overturn default assumptions. The pattern of practical reasoning sketched at the outset assumes a set of preconditions characteristic of causal reasoning. Our reasoning about when events occurred, or whether they are still occurring may rest on the basis of incomplete information. In [3,4], it was argued, somewhat reluctantly, that a temporal database which allowed for incompletely specified event occurrences should provide for a dual representation scheme for events. The problem posed at the outset of this paper offers no clear intuitive solution. We suggest a solution which, we feel, makes certain inferences transparent from the standpoint of first-order predicate calculus.

The most well-developed alternative to the approach taken in this paper is the viewpoint is set forth by James Allen in a series of articles. [1,2]. The goals of Allen's work on action and event theory and those of this paper are related, but different. Allen is proposing a general theory of event representation in order to design systems that predict, plan, and in general reason about, temporal entities like events and processes. He extends temporal logic to include an account of events, actions, belief, intention and causality. Our emphasis is at the "front end", i.e., the semantic processing of natural language sentences about events. Our goal is the design and implementation of a natural language interface to an "time-intelligent system", i.e., one that reasons about, and responds to, queries and assertions about temporal entities.

The basic response of Allen [2] to the problem of identifying events is based on the theory of action of Goldman [7], who holds that a sentence like "By shooting a gun, Jim killed Sam" should be interpreted as identifying two distinct event occurrences which are related by a relation Goldman calls "generation". The properties true of actions related by generation include: they are cotemporal, neither is a part of the other, and there is a context such that doing one action in that context implies doing the other. (It should be noted that Allen appears to drift into the opposing view when he states that "vastly different patterns of behavior can be classified as the *same* action. For example, turning on a light generally involves flipping a light switch...". [2, p.126]. (Our emphasis) This commonsense notion of sameness, it seems to us, goes unexplained in his view.)

Allen claims that "the notion of generation is crucial for considering how an action as performed, or how to perform an action (i.e. planning)." Generational knowledge about voting, for example, tells us how to vote, because it tells us that, given the right context, doing a (conceptually simpler) act of pulling a lever 'implies" the occurrence of voting. Allen's approach succeeds in making manifest many of the important semantic issues regarding event identity. It seems to us however, that

the rather obscure concept of "implying the occurrence of an action" is clearly explained in our view in terms of identity: the occurrence A and the occurrence of B are just the same occurrence. This makes the inferencing mechanism an instance of the first-order principle of "substitutivity of identity".

Allen claims that Goldman's view provides for a "simpler semantics" [1, p.125] Presumably, this is because his view does not require a distinction between a fully instantiated representation of an event and an occurrence of an event, whereas our view does. We feel that the solution of this paper offers a cogent explanation of the inferencing done in making practical decisions. The proof of this claim awaits its implementation, and hence for this paper should be regarding as a thesis.

3. Our Solution

There are many examples of the sort of semantic problem being addressed here. John pulls a lever, *thereby* voting. *By* uttering the right words (in the right circumstances) I marry someone. Generally, a connection is established between what we will loosely term "event types" (e.g., pulling and voting, uttering and marrying, walking behind and following). The correlation seems to be between a physical event type (uttering, pulling, stabbing) and an event of a "non-physical" type e.g., a political action such as voting. (We view actions roughly as events involving an agent with an intention to do something.)

It seems awkward to say that, in voting, two distinct acts were performed, a pulling of a lever and a voting. Furthermore, the counting need not stop here, for pulling a lever involves a grasping of the lever and a contraction of the muscles. We seem forced to conclude that there is a whole multitude of distinct occurrences going on. This, we hold is less intuitive than to maintain that for one event, we may identify a number of distinct descriptions. An analogy, familiar to philosophers, is that for one physical object there may correspond a number of distinct descriptions of the object (for example, the planet Venus as the Morning Star and the Evening Star). Thus, we hold that the proper semantic treatment of sentences like "By pulling the lever, John voted" should involve the identification of one event. In what follows, we will describe an approach to the semantics of sentences of the form "By doing A, B has been done" under the assumption that an occurrence of just one event has happened.

On Davidson's view [6], the logical form of simple sentences like "John voted" involves quantification over events. A logically perspicuous paraphrase of the sentence would be "There is an event e such that e is a voting by John". Thus, a sentence like "By pulling a lever, John voted" would be interpreted as involving a pair of propositional (sentential) forms connected by the non-logical sentential connective "by". An approach which interpreted this sentence as identifying one event with distinct descriptions would point to the connective "by" as indicating the identity. It is tempting, then, to simply rephrase sentence (2) above as:

Putting the block in the box at t = Hiding the block at t.

This approach won't quite do, however, without further analysis. Identity is symmetric, and thus this sentence means the same as

Hiding the block at t = Putting the block in the box at t.

which is the analysis of "By hiding the block, I will put the block in the box," which is semantically anomalous, and hence has a different meaning than, the original sentence. The order of the event descriptions in the sentence indicates a semantic relation between the voting and the pulling. This relationship obviously is not an "isa" link in the network, however, since neither of these event classes are contained in the other. (Not all votings are pullings of levers, and not all pullings of levers are voting.) The semantic relationship is a "long-distance" one between different events types. A proper treatment of the meaning of the connective "by" must be able to explain the anomaly.

It is common to think of the world as containing objects, including occurrences of events and properties or relations true of them. Thus, we can speak of the the event of Sirhan killing Kennedy, and of the state of Kennedy's being dead. Actions can be causally related to other actions ("When John fired the gun, Sam winced"), as well as to the instigation of a period when a property is true of an object ("John's promotion resulted in him having a raise in salary"). Let us use the notion of "state" to characterize this property-object relationship, and speak of actions resulting in new states of the world. The stabbing of Brutus by Caesar resulted in Caesar's death, the uttering of the words "I do", with the state of being married. In general, stabbings are causally related to deaths. For every state, in turn, there is an action which by definition leads to it. The act of marrying is inextricably entwined with the state of being married, being killed, with the state of being dead. More formally, for every event e which is a H-ing, there exists a state of being H-ed. The basic claim of this paper is that interplay of semantic and causal relations leads to an identification, although not a necessary one, between occurrences of actions: acts of stabbing with acts of killing, acts of hiding with the acts of putting into a large box, etc. (This form of reasoning is somewhat the converse of reasoning by default: with the latter we say that something is true unless we find out additional things to the contrary; with the reasoning that leads to an identification of event occurrence, we tend to say that this identification is not to be made *unless* certain other truths are known. A typical utterance of "I do" is not an act of getting married!). The identification of a hiding of a block (now) with the putting of the same block into a box (now) thus seems to be the

result of deliberation about causation and and meaning. A summary of this reasoning can be represented in a semantic network, pictured below (Figure 1). This network is a representation of two propositions:

- For all events e of X hiding Y there is a state s such that in s Y is hidden.
- An event e of putting X in Y causes the state s of Y being hidden.

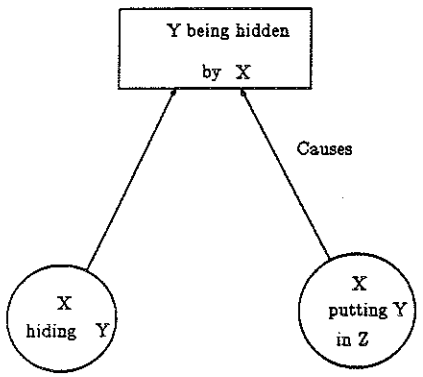


figure 1.

What we claim is that an occurrence of this pattern of relations between event types and state types in a knowledge base leads to an identification of occurrences of each event type (although, of course, not a necessary identity). This leads to a representation of sentences like (2) in terms of the network in Figure 2. In predicate calculus notation, this inference would be expressed as "there is an event e such that e is a hiding of block A in box B (by agent C) at t and e is a putting of block A in box B (by C) at t".

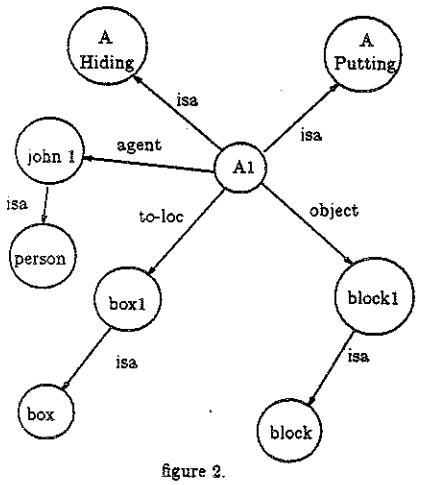


figure 2.

The quantification posits the existence of a unique event. Implicit in this reasoning process is what might be called "reasoning backward" from more complex actions like hiding, to more primitive ones, like putting. How this process might be incorporated into the search through the knowledge base is unclear. The point we have tried to make in this paper is that the realization that, for example, an act of putting solves a goal of hiding something, amounts to a recognition that an occurrence of one act is identical to the occurrence of the other. This, we claim, is missing from Allen's account.

4. Summary

We have examined a problem concerning the identification of event occurrences. We have noted the solution of this problem is fundamental for any representationn of knowledge about events. Our solution suggests an event ontology in which there are multiple descriptions of a single occurrence. Further work should be done in implementing this representation scheme into a system that makes decisions about actions. It has been the authors' claim that the result will be a system in which practical reasoning will involve reasoning about the identity of occurrences of events.

References

[1] Allen, J. F., *Towards a General Theory of Action and Time*, Artificial Intelligence 23 (1984), pp. 123-154.

[2] Allen, J. F., *Natural Language Understanding*, Benjamin/Cummings Press, Menlo Park, CA., (1987).

[3] Anger, F., Mata-Toledo, R., Morris, R., Rodriguez, R. "A Relational Knowledge Base With Temporal Reasoning," PROCEEDINGS OF THE FIRST FLORIDA ARTIFICIAL INTELLIGENCE RESEARCH SYMPOSIUM, Orlando, FL, (1988), pp. 147-151.

[4] Anger, F., Mata-Toledo, R., Morris, R., Rodriguez, R. "A Temporal Relational Calculus" PROCEEDINGS OF THE AUSTRALIAN JOINT ARTIFICIAL INTELLIGENCE CONFERENCE Adelaide, S.A. (1988), pp. 146-155.

[5] Austin, J.L.: *Performative Utterances*, in Philosophical Papers, Oxford University Press, (1979), p. 235.

[6] Davidson, D: *The Individuation of Events* in Essays on Actions and Events. Oxford: Clarendon Press, (1980), p.171.

[7] Goldman, A., *A Theory of Human Action* U. Pittsburgh Press, Pittsburgh, PA., (1970).

[8] Thomson, J. J. *Acts and Other Events*, Cornell University Press (1977), p. 19.

ON GRAMMAR INFERENCE AND ENTITY RELATIONS

by

PRATIK BISWAS AND ABRAHAM KANDEL

Department of Computer Science
and
The Institute for Expert Systems and Robotics
The Florida State University
Tallahassee, Florida 32306 - 4019
U.S.A.

Key Words.

Grammatical Inference, Relational Grammar, Entity Sets, Relationship Sets, ER Diagrams, Well-formed ER model.

Abstract

This paper offers a systematic way to deal with the grammar inference problem through an Entity-Relationship approach. A new algorithm is proposed for grammatical inference. A data model, known as the Entity-Relationship model [*Chen*, 76] has been used to represent and retrieve knowledge from a restricted domain. A domain comprising of one or two-dimensional hierarchical structures is examined and documented in natural languages. Entity-Relationship Diagrams (ERDs) are applied to extract the entity/relationship types from the English text. These pictorial representations are then analyzed semantically and syntactically to infer a grammar capable of generating these structures. The capabilities and limitations of the approach have been compared with other available techniques along with an insight to future research in this area.

* *This research has been supported in part by grant IST 8405953 and by the Florida High Technology and the Industrial Council grant UPN 85100316.*

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

1. Introduction

In the terminology of linguistics, learning from sample patterns is easily interpreted as the problem of learning a grammar from a set of sample sentences. This procedure commonly referred to as *grammatical inference*, is an important subject matter because of its automatic learning implications. However the area of grammatical inference is still in its infancy and there is no known scheme capable of attacking this problem in its general form. Instead numerous algorithms for the inference of restricted grammars have been proposed. Most of these algorithms are heuristic in nature. A study of these various algorithms motivated us to find a solution to this problem which is systematic, rational and reasonably accurate.

Formally speaking, the problem of grammatical inference is concerned mainly with the procedures that can be used to infer the syntactic rules of an unknown grammar G based on a finite set of sentences from L(G).

The inferred grammar should be capable of producing the given syntactic entities and in some cases many more. It is to be noted here that the input set can be a singleton.

2.Background

Next we review the material which is essential for the understanding of our problem.

2.1 Relational Grammar (structural grammar)

The grammars, that we attempt to infer, describe structures/sentences in terms of the relationships involved among various structural components/primitives. They would, in our case, tend to vary from context-free to regular.

A *relational grammar* G is a four-tuple (V_n, V_t, P, S) where V_n, V_t, and P and S represent the *nonterminals, terminals, productions* and the *starting symbol* as usual. A production in P is however of the form :

$$A \rightarrow r(x, y)$$

where $A \in V_n$ and $r \in R$ (a specified set of relationships) is any binary relationship satisfied by any x and y. We did not include R in the definition of the grammar, just to maintain uniformity with the standard definitions.

Example 2.1.1 :

As an example, consider the following grammar,

$G = \{\{S\}, \{a\}, P, S\}$, where P:

$S \rightarrow CAT(a, S)$

$S \rightarrow CAT(a, \lambda)$

Where CAT stands for the Concatenation relation between any two structures.

2.2 Entity sets, Relationship sets and ER diagrams

An *Entity set* is a collection of several objects which share some common property.

Example : A specific person JOHN is an element of the entity set PERSON.

Relationships are meaningful associations between entities.

A *relationship set*, R_i, is a mathematical relation among n entities, each taken from an entity set.

$$\{(e_1, .., e_n) \mid e_1 \in E_1, .., e_n \in E_n\}$$

Example : 'FATHER-SON' is a relationship made up of two PERSON entities.

The *ER (entity-relationship) diagrammatic* technique is a graphic way of displaying entity types , relationship types and attributes. There are many variations of ER diagrams in the litertature. We shall however be consistent with *P. Chen*'s work [*Chen*, 76], published in his introductory paper in 1976.

- an entity set is represented by a rectangular box.
- an entity is represented by a '•' inside the rectangle.
- a relationship set is represented by a diamond shaped box.
- a circle represents a value set, to which an attribute maps an entity.

2.3 Well-formed ER Model

Our enterprise consists of the input sample set of one or two-dimensional patterns.

Our well-formed ER model is defined by :

1. Object set types in an ER model.
2. The structural relationships between the different types of object sets.

Since the attributes of the object sets are of no use in our case, they do not form part of our ER model.

- **Object sets**

  The types of object sets that make up an ER model of our enterprise are :

  entity sets - structures (patterns).

  relationship sets - relation among subpatterns.

  dependent entity sets - recursive structures/patterns

  subsets - substructures (subpatterns).

  supersets - structures/substructures (patterns/subpatterns)

- **Structural relationships between object sets**

  Object sets in our model of enterprise must satisfy the following properties :

  - an entity set may be
    1. unconnected.
    2. connected to any number of relationship sets.
    3. own any number of dependent entity sets.
  - a relationship set must be
    1. connected to two object sets, of which none should be a relationship set.
    2. may be owned by one of the different entity sets involved in the relationship.
    3. all relationships in our model are one to one.

2.4 Entities and Relationships in a Grammar

It is interesting to note how the different components of a grammar relate themselves easily to the concepts of entities and relationships.

The starting symbol (S) represents a complete structure which is an entity set type by itself.

Other Nonterminals represent different syntactic categories and are substructures of different entity set types.

Terminals are primitive structures or basic building blocks that can be thought of as singleton entity sets.

The set P contains one or more relationships between the different entity set types. Each relationship in that set may or may not be a singleton.

Example :

Consider the grammar defined by

$G = \{\{S\}, \{a\}, P, S\}$, where P:

$S \rightarrow CAT(a, S)$

$S \rightarrow CAT(a, \lambda)$

Here the *entity sets* are S and a and the *relationship set* is CAT. The set a in this case is a singleton.

It may be noted at this point that we use sentence / string / structure / pattern / entity at different contexts to refer to one and same thing.

3. Algorithm for Grammar Inference

We propose a 6-step algorithm for grammatical inference from the set of input patterns, using the ER approach.

step 1 : A hierarchical description of the patterns in the language of the grammar to be inferred.

step 2 : Conversion of pattern descriptions to an easy to read natural language descriptions.

step 3 : Translation of the natural language descriptions into ER diagrams.

step 4 : Modification of the ER diagrams.

step 5 : Merging (if possible) of the ER di agrams to form the ER model of our enterprise.

step 6 : Inference of the relational grammar.

Next we perform an in-depth analysis of the various steps involved in the algorithm.

3.1 A hierarchical description of patterns in the language of the grammar to be inferred.

We describe the strings/patterns/structures hierarchically, as far as possible, using the given positional descriptors (relationships) on the given pattern primitives (terminals). The purpose is to make the descriptions match sentences in the language of the grammar to be inferred.

- Example 3.1.1 :

  structure -

  $$\frac{a+b}{c}$$

  (mathematical expression)

  Given primitives - a,b,c,+,-

  Given relationships - ABOVE, LEFT

  Description -

  ABOVE (ABOVE (LEFT (a, LEFT(+,b)), -), c)

3.2 Conversion of pattern description into a natural language description.

The pattern description at step 1 is converted into an easy to read English language description to facilitate the translation process at step 3. A structure /pattern /string/ sentence (S) is described in terms of its constituent subentities in line with the pattern description at step 1. Subentities are characterized by their complexities. Complexity of a subpattern is defined by the number of primitives it has (terminals under consideration), e.g. the number of sides, the number of characters, etc. The symbol O with a subscript, for its complexity, is used to denote the substructure under consideration. Superscripts are used to distinguish one sample from others in the input set.

- Example 3.2.1 :

  Pattern description :

  ABOVE (h, (ABOVE (LEFT (v, v)), h))

  Natural language description :

  The structure (S) has an entity 'h' (horizontal bar) above O_3. O_3 has O_2 above a 'h'. O_2 has an entity 'v' (vertical bar) to the left of another entity 'v'.

3.3 Translation of the natural language descriptions into ER diagrams.

The underlying philosophy is a well-known saying that states , "*A picture is better than a thousand words.*" When a person looks at a picture, he grasps the information in a two-dimensional way. But when he reads a page of words, the one-dimensional (sequential) orientation limits his speed of comprehension.

The scheme is an adaptation from *P. Chen*'s work [*Chen*, 83] for translating English sentences into ER diagrams. Some of his rules have been modified to suit our work. The following are explanations of the translation rules used for our purpose.

- **Rule 1**

  A *common noun* in English corresponds to an entity type in an ER diagram. A *proper noun* is however an instance (occurrence) of an entity from an entity set.

- **Rule 2**

  Transitive Verbs correspond to relationship types in ER diagrams.

- **Rule 3**

  Positional descriptors (mostly Prepositions) correspond to relationship types. ABOVE (*to the above of*), LEFT (*to the left of*) , etc form relationship sets.

- **Rule 4**

  A *clause* in English is a high-level entity type abstracted from a group of interconnected low-level entity and relationship types, in ER diagrams.

- **Rule 5**

  A *sentence* in English corresponds to one or more entity types connected by a relationship type, in which each entity type can be decomposed (recursively) into low-level entity types interconnected by relationship types.

- **Rule 6**

  When a pattern description is translated sentence by sentence each relationship is made unique. Otherwise we would have the same relationship set type appearing more than once.

3.4 Modification of the ER diagrams.

We now present a set of rules to simplify the ER diagrams obtained at the previous step. These rules employ important entity-relationship concepts, consistent with the literature in this area. Any modification made on the ER diagram of any pattern is used subsequently to modify the ER diagrams of the other elements of the input set.

- **Rule 1** : Ownership
  In case of a relationship arising out of a verb like '*has*' the entity set type on the left *owns* the entity set type on the right. The entity on the right is a *subset* of the entity set type on the left. In terms of pattern grammar the owner is the name of the *syntactic category* that owns the *substructure* on the right. A *dashed rectangle* is used to represent the syntactic category.
  The idea is to make the ER diagram of the pattern hierarchical.

- **Rule 2** : Redundancy
  In this we attempt to relate semantic concepts from Entity Relationship set theory to remove redundant information from our domain.

  i) Instances (occurrences) of *identical structures* belong to the same set. Elements belonging to the same entity set are identified and merged. Arrows are used to resolve ambiguities.

  ii) If the domain involves different instances of the *same relationship* type then the diamonds can be removed from the diagram as we do not consider any attribute of the relationship.

  iii) When an entity set type is a *subset of a subset*, the superset owns the innermost entity set type. When two syntactic categories are equivalent, eliminating the lower one in hierarchy helps in reducing a rule.

  iv) Once the basic entity set types (*subsets*) and the syntactic categories are (*supersets*) are idetified, the *supersets* are substituted for the *subsets* uniformly in all the sample patterns. This helps to increase the generating power of the grammar and eliminates duplication in step 5.

  Rules *iii)* and *iv)* follow from the Generalization-Association [*Su*, 88] between *supersets* and *subsets*. Since we do not consider any specific attributes of the subsets and since the subsets inherit all the properties of the supersets, their substitution or elimination will not result in any loss of information in our case.

- **Rule 3** : Dependency (dependent entities)
  This rule relates recursivity of patterns to dependency of entity sets. The more recursive a structure is, the more dependent it becomes on the basic structure. A *recursive* structure can be built recursively from a basic entity, such that both the basic structure and the recursive structure belong to the same syntactic category.
  The *dependent entities* are identified and the ER diagram is amended to remove all intermediate entities leaving behind the inner and the outermost dependent entity set types, the inner set type belonging to the outer syntactic category but the outer set type being recursively dependent on the basic structure. This follows from general recursion theory where we use a base and a general case to define recursive functions.

3.5 Merging the ER diagrams to form an ER model of our enterprise.

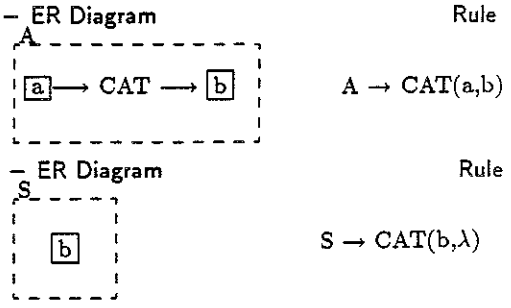
Once the ER diagrams of all the input patterns (in case of more than one) have been drawn and amended, they are combined to form the ER model of our enterprise, which in this case is the input sample set. The ER model describes in a graphic way the language of the grammar to be inferred.

We start with the innermost entity types and reach out to the outermost syntactic categories. Every entity and relationship set type encountered in the ER diagrams of the various input patterns is represented only once in the ER model. However if too many entity or relationship set types are involved, merging can become very complicated and clumsy and will not serve the purpose of graphic representation. In such cases it is advisable to avoid this step without loosing any information.

3.6 Grammar Inference

We infer the *relational grammar* from the ER model of our enterprise, using the following rules :

- **Rule 1** : *Firm rectangles* represent *terminals.*
- **Rule 2** : *Dashed rectangles* represent *non-terminals.*
- **Rule 3** : *Production rules* are inferred from the following guidelines.



4. Conclusions

In this work, we have attempted to use the ER modelling aid in grammar inference. The foregoing analysis has demonstrated that the ER approach can be successfully applied as a knowledge-base both for representing and retrieving knowledge. The result obtained compares favourably with other standard inference algorithms available in the literature, and in some cases is identical to the output of *Feldman's* or *Evans's* algorithm on which it is modelled. However a few major differences with other algorithms need to be pointed out here.

In general, our algorithm tends to infer a grammar which has a high generating power as recursion is incorporated at a higher level than most other algorithms. An advantage of this happens to be the elimination of redundant *non-terminals (Syntactic Categories)*. But this may run into a problem when a negative set (consisting of patterns that the grammar should not generate) is specified alongside the input sample set (positive). In such cases recursion would not be incorporated and the objective would be to infer a canonical grammar, capable of generating only the given sample set.

Secondly the ER diagrammatic technique is a graphic representation of the various relationship types among the different object types, intended to bring out better comprehension in a two-dimensional way. So its effectiveness tends to get blurred if the domain is extended and involves too many entity and relationship types.

Lastly but not the least the precision of the approach is very much dependent on successful conversion of natural language descriptions into ER diagrams. But using natural language as a system specification tool has its own drawbacks. For example, natural language statements tend to be ambiguous and may in such a case result in an erroneous grammar. But since we exclude attributes of objects from our ER model, we leave out much of the ambiguities with it.

This research analyzes and illustrates a new approach manually and opens up a new vista of interest which can be explored with computerized aids. The idea is still in an experimental stage with many rough edges and improvements can still be suggested. There is more scope for refinement in the natural language descriptions and the modification of ER diagrams. It has been a modest effort in exploring a hitherto unknown area.

The application of the entity-relationship approach to picture-representation offers what appears to be a fertile field for further study. Much further work remains to be done.

The presence of ambiguous pictures and images calls for *fuzzy entities* and *fuzzy relationships*. This necessitates the use of *fuzzy ERDs* and *fuzzy languages*. With so much uncertainty around the next logical step could be the inference of a *Fuzzy grammar* capable of recognizing *fuzzy structures*.

Areas for future research where our ideas can have useful applications include data engineering, knowledge engineering, pattern recognition, artificial intelligence, fuzzy language theory, computer graphics and pictorial database design.

4. References

[Chomsky 64] N. Chomsky. *Aspects of the Theory of syntax. MIT Press, Cambridge, Massachusetts.* (1964)

[Evans 71] T. G. Evans. *Grammatical inference techniques in pattern analysis. Software Engineering* (J. T. Tou, Ed.), *Vol. 2. Academic Press, New York.* (1971)

[Feldman 72] J. A. Feldman. *Some Decidability Results on Grammatical Inference and Complexity. Information Control* **20**, 244-262. (1972)

[Chen 76] P. P. Chen. *The entity-relationship model: Toward a unified view of data. ACM Trans. Database Systems 1, No. 1.* (March, 1976)

[Chen 83] P. P. Chen. *English Sentence Structure and Entity-Relationship Diagrams. Information Sciences* **29**, 127-149. (1983)

[Hawryszkiewycz 83] I. T. Hawryszkiewycz. *A Computer-Aid for E-R Modelling. IEEE transactions on Entity-Relationship.* (1985)

[Su 88] S. Y. W. Su, V. Krishnamurthy and H. Lam. *An object-oriented semantic association model (OSAM* −)*. AI in Industrial Engineering and Manufacturing : Theoretical Issues and Applications, AIIE.* (1988)

COIMPLICATION AND THE RESOLUTION PROCEDURE IN FUZZY EXPERT SYSTEM

Kyung-Whan Oh
Department of Computer Science and Supercomputer Computations Research Institute
Florida State University, Tallahassee, FL 32306, U.S.A.
and
Abraham Kandel
Department of Computer Science and The Florida SUS Center for Artificial Intelligence
Florida State University, Tallahassee, FL 32306, U.S.A.

ABSTRACT

Many of the present expert systems use two-valued logic and probability. Since much of the information in the knowledge base of a typical expert system is imprecise and vague, it is well known that we can not handle those kinds of uncertainties using those tools.

In a fuzzy conditional inference of the fuzzy expert systems, the problem is that, given values, for A->B and A', we have to find a consistent value for B' through the modus ponens. In this paper we propose to use the equivalence relation for modus ponens of the inference in fuzzy expert systems instead of fuzzy implication and introduce the concept of coimplication as a new approach to approximate reasoning of expert systems using the new concept of modus ponens.

Keywords: Fuzzy implication operator, modus ponens,Coimplication, Fuzzy logic, Fuzzy sets, x-tautology,Fuzzy expert systems, Approximate Reasoning

* This research has been supported in part by NSF grant IST 8405953, by the Florida High Technology and Industrial Council grant UPN 85100316, and by the Florida State University Supercomputer Computations Research Institute which is partially funded by the U.S. Department of Energy through Contract No. DE-FC05-85ER250000.
** Permanent address after March, 1989:
Department of Computer Science, Sogang University,
Sinsudong, Mapoku, Seoul 121-742, Korea

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

INTRODUCTION

In a domain with little imprecision an expert system can likely be developed. But,the development of an expert system is very difficult in a domain with a large amount of imprecision. There are several sources of imprecision and uncertainty in expert system areas and when a solution is imprecise it must be presented to the system user in a manner which indicates the uncertainty in it.

By fuzzy expert system(Hall 1986; Kohout and Bandler 1982) we mean an expert system in which the uncertainty management is based on the theory of fuzzy sets. Thus fuzziness is used in handling imprecision and uncertainty for the inference process and knowledge represenation scheme. The theory of fuzzy sets and fuzzy logic is well-founded and strong. The theoretical basis behind fuzzy techniques will allow us to deal with uncertainty in a manner that is well-supported.The fuzzy logic is good for imprecise applications but it is still somewhat cumbersome.

Management of uncertainty is related to a computational analysis of uncertainty from the premises to the conclusion(Zadeh 1983). Uncertainties in the premises will be transmitted to the conclusion. In other words, there is a relation between the premise and the conclusion in an inference. We attempt to establish such a computational framework based on fuzzy logic to deal with uncertainty in fuzzy expert systems.

In a fuzzy conditional inference in fuzzy expert systems the problem is that if we are given values for A -> B and A', we have to find a consistent value for B' through modus ponens. In the real fuzzy logic system we propose to use the equivalence relation for modus ponens of the inference on fuzzy expert systems instead of implication, survey a property in equivalence and introduce a new approach to approximate reasoning using this new concept of modus ponens. Based on the equivalence method, we have developed a theory of dealing with vagueness in expert systems which is called

coimplication. The purpose of this paper is to provide guidance for modeling general domains.

A NEW REASONING METHOD AND COIMPLICATION

In fuzzy expert systems, the problem is that, given x->y and x, we must find a value y which is consistent i.e. to find the value of the conclusion of an implication relation by the use of fuzzy modus ponens(Bandler and Kohout 1984).

DEFINITION 1:

1) symmetrical difference:$v(P|Q)=|v(P->Q)-v(Q->P)|$ $=|p-q|$
2) asymmetrical difference:
   i) $v(P|-Q) = v(P->Q) - v(Q->P) = q - p$
   ii) $v(P-|Q) = v(Q->P) - v(P->Q) = p - q$

The concept of symmetrical difference of fuzzy sets is discussed in (Dubois and Prade 1980), and (Kandel 1986). Theorem 1 shows a relation between the equivalence and the symmetrical difference of P and Q.

THEOREM 1:

$v(P<->Q) = 1 - |p - q| = 1 - v(P|Q) = v(\text{not}(P|Q))$

proof:

$v(P<->Q)=v(P->Q \text{ and } Q->P)$
$=\min(\min(1-p+q,1),\min(1-q+p,1))$
i) $p \geq q : v(P<->Q) = \min(1-p+q,1) = 1-p+q = 1-|p-q|$
ii) $p < q : v(P<->Q) = \min(1,1-q+p) = 1-q+p = 1-|p-q|$

Consider a method applying these operators to an inference through modus ponens. The classical and plausible modus ponens is the formula "IF P->Q AND P THEN Q" which is a tautology.

The new approach is to use the equivalence operator, which is a similarity relation (Kandel 1986), in modus ponens in expert systems i.e. the formula "IF P<->Q AND P THEN Q" which is still a tautology.

Let us consider a specific example for the fuzzy conditional inference.

Ant 1 :If tomato is red (0.7) then tomato is ripe(0.8)
Ant 2 : tomato is red (0.8)

---

tomato is ripe (?)

When we use Lukasiewicz's fuzzy implication operator(Bandler and Kohout 1980), the truth degree of Ant 1 is 1; thus, given x -> y and x, then y is determined. In this case, the truth degree of "tomato is ripe" is greater than or equal to 0.8. Consider the above example by the equivalence operator :

Ant 1 : tomato is red (0.7) <-> tomato is ripe (0.8)
Ant 2 : tomato is red (0.8)

---

tomato is ripe (?)

If we use the equivalence operator then the truth degree of "tomato is ripe" is 0.9. Therefore, 0.9 in the equivalence method is more specific than 0.8 in the implication method. If the truth degree of "tomato is red" is greater than the truth degree of"tomato is ripe", then both methods have the same results.

The definition of inference from a set of premises(Yager 1985)] can be improved in a fuzzy real environment system using the equivalence method.

DEFINITION 2:

A fuzzy statement form which is always greater than or equal to x, $x \in (0,1]$, as a truth degree, no matter what the truth values of its fuzzy statement letters may be, is called a x-tautology.

DEFINITION 3:

p

P1,P2,...Pn fuzzily coimplies P, denoted $(P1,..,Pn)|-|P$, if for every fuzzy consistent interpretation of the atoms in the propositions P1,,,Pn and P, $p=\sup_{P,Q}(|v(Q) - v(P)|)$ and $p \leq 0.5$, where p is a maximum possible symmetrical difference between Q and P, and $Q=P1 \wedge .. \wedge Pn$.

THEOREM 2:

$1-x$

In fuzzy logic, $(P1,..,Pn)|-|P$ if $(P1 \wedge .. \wedge Pn)<->P$ is a x-tautology.

Proof:

Assume $(P1 \wedge .. \wedge Pn)<->P$ is a x-tautology, where $x \geq 0.5$. From the property of <->, this requires that $1- |v(P1 \wedge .. \wedge Pn)-v(P)| \geq x$, i.e. $|v(P1 \wedge .. \wedge Pn) - v(P)| \leq 1 - x$.
Thus $p = \sup_{Pi,P}(|v(P1 \wedge .. \wedge Pn)-v(P)|) = 1-x \leq 0.5$.

THEOREM 3 (Syllogistic Reasoning):

Let P, Q and R be propositions.
If $(P)|-|_{p1}Q$, $(Q)|-|_{p2}R$ and $p1 + p2 < 0.5$ then $(P)|-|_{p1+p2}R$.

Proof:

Let $p = v(P)$, $q = v(Q)$ and $r = v(R)$. Then $|p - q| \leq p1$ and $|q - r| \leq p2$. $-p1 \leq p - q \leq p1$ and $-p2 \leq q - r \leq p2$. Therefore
$|p - r| \leq p1 + p2$. Thus $(P)|-|_{p1+p2}R$

COROLLARY 1:

Let P1,P2,..,Pn be propositions. Let [b1,B1],..,[bn,Bn] be truth intervals of P1,..,Pn, respectively. Then the truth interval [b,B] of the conjunction P1∧P2∧..∧Pn is defined by b=inf(bi) and B=inf(Bi), and the truth interval [b,B] of the disjunction P1∨P2∨..∨Pn is defined by b=sup(bi) and B=sup(Bi).

In the resolution process, first of all, we have to find the truth interval of the antecedent and find a subpartition covering it. Algorithm 1 shows the splitting procedure of the truth interval of the antecedent Ia, calculated by Corollary 1, into a subpartition.

ALGORITHM 1: Splitting

1. Find a minimum subpartition {Ii}i=s,t covering the truth interval Ia of the antecedent such that $Ia \subseteq \bigcup_{i=s}^{t} Ii$, $\bigcup_{i=1}^{s-1} Ii \cap Ia = \emptyset$ and $\bigcup_{i=t+1}^{n} Ii \cap Ia = \emptyset$.
2. Set the left bound of Is to the left bound of Ia and the right bound of It to the right bound of Ia i.e. a subpartition {Ii'}i=s,t

From {Ii'} i=s,t, and the property of asymmetrical difference between the antecedent and the conclusion we can infer that conclusion has the possible interval Ic=[bc,Bc] of the truth degree of the conclusion as in Algorithm 2.

ALGORITHM 2: Resolution

1. From {Ii'}i=s,t, find {Ii"}i=s,t by Ii" = Ii' + di
2. Find the truth degree [bc,Bc] of the conclusion by bc $=\inf(\bigcup_{i=s,t} Ii")$ and Bc$=\sup(\bigcup_{i=s,t} Ii")$.
3. Find the symmetrical difference SD by SD $=\sup_{i=s,t}(sdi)$.

In expert systems, several conclusions concerning a variable may be obtained from different rules. The structure for facts and rules defined in the previous section will be applied to compute a final truth degree of the conclusion and combine several different pieces of information

Basically, $A1 \vdash_{(d1)} C, \ldots, An \vdash_{(dn)} C$ are combined into

$A1 \vee .. \vee An \vdash_{(d)} C$, where $d = \inf(di)_{i=1,n}$.

ALGORITHM 3 : Combination

(Case of 2 rules i.e. $A \vee B \vdash_{(d)} C$)

1. Go to Algorithm 1 with the disjunction interval [b,B] and find {Ii'}i=s,t.
2. Find di[A]i=s,t and di[B]i=s,t, and if di does not exist for some i, i=s,t then set di to 0.
3. Calculate di[A ∨ B] by min(di[A],di[B]) for i=s,t.
4. Go to Algorithm 2 for the combined truth interval of C.

CONCLUSIONS

A new reasoning method with the equivalence operator instead of the implication operator in modus ponens has been described in this paper. It demonstrates a fuzzy-logic-based computational framework. Based on this method, the concept of coimplication in the inference process has been developed.Coimplication method has the advantage of being computationally simple and efficient even for multiple rules.

Many expert systems operate with vagueness in various ways although they do not make use of fuzzy techniques. We have modelled vagueness through the coimplication method. The models developed thus far have no theoretical base and have been inadequately analyzed. We have attempted to provide guidance for modelling general domains.

We have developed a General Purpose Fuzzy Expert System(GPFES) using the concept of the coimplication in a fuzzy environment which is an attempt to handle uncertainties in general domain(Oh 1988). GPFES has been applied to passive multi-channel microwave precipitation retrieval by means of remote sensing(Oh 1988).

For an actual application in fuzzy expert systems we are using the following definition.

DEFINITION 4:

P1,,Pn fuzzily coimplies P approximately with an asymmetrical difference d, denoted $(P1,..,Pn) \vdash_{(d)} P$, if for every fuzzy consistent interpretaion of the atoms in the propositions P1,,Pn and P, p1 = inf(v(P) - v(Q)) and p2 = sup(v(P) - v(Q)), and p2 - p1 $\leq$ 0.5 and d = (p1 + p2) / 2, where Q = P1$\wedge$..$\wedge$Pn and d $\doteq$ v(P) - v(Q).

THEOREM 4:

If $(A) \vdash_{(d1)} C$, $(B) \vdash_{(d2)} C$ then $(A \vee B) \vdash_{(\min(d1,d2))} C$

Proof:

Let a = v(A), b = v(B) and c = v(C). v(C) - v(A) $\doteq$ d1 and v(C) - v(B) $\doteq$ d2. Then v(C) - max(v(A),v(B)) $\doteq$ min(d1,d2).

KNOWLEDGE REPRESENTATION AND RESOLUTION PROCEDURE

In this section we show the structure of facts and rules to represent vague information and handle propagation of uncertainty in expert systems using the coimplication.

A coimplication relation may have an a priori determined asymmetrical difference representing a certainty on the interval [-0.5,0.5], which is given by the domain expert. Coimplication relations are the primary relations used for the inference process. They are very flexible relations which can be used for any application.

For representation of facts the closed interval [0,1] of truth degree is divided into n disjoint intervals; I1=[b1,B1],I2=[b2,B2],,,In=[bn,Bn], where Ii and Ij are disjoint (i$\neq$j), and b1=0,B1=b2,,Bn-1=bn,Bn=1.

Let {fi} be the frequencies of items taking fuzzy membership values in the interval Ii (i=1,n) related to a fuzzy subset F. The basic form of an uncertain and/or imprecise fact is (predicate,F,{(Ii,fi,ui)},b,B), where b and B are the lower bound and the upper bound of the truth values of the fact, respectively,

and $ui = \sum_{k=1}^{fi} \chi_F(xk)/fi$, $\chi_F(xk) \in Ii$;

ui is an usual value of the interval Ii; see (Zadeh 1986). Suppose that a universe of discourse U has 100 elements, i.e. x1,..,x100. A fuzzy subset F in U is characterized by a membership function χ_F which takes the value in the interval [0,1], i.e. χ_F:U -> [0,1].

Because U is discrete, F is represented as

$F = \chi_F(x1)/x1 + .. + \chi_F(x100)/x100.$

Let Ii and Ij be the intervals containing b and B, respectively, where i $\leq$ j, and $M = \sum_{k=1}^{j} fk$. Then the certainty degree p is approximately defined as

$$p = 1 - P(UIk) + 1/100, \quad 1 \leq k \leq j$$
$$= 1 - M/100 + 1/100.$$

If b=B=1 and fn=1 then p = 1. Given an imprecise truth value r and its certainty degree p, we can transform them into b and B. The basic idea is to count for fi upward and downward, alternately, from a partitioned interval Ia containing r as 101 - 100p.

In many cases, given an uncertain truth degree of the antecedent, it is possible that the conclusion has several different truth degrees. For solving this kind of problem, we are applying the asymmetrical coimplication to the rules on the partitioned intervals.

A rule is composed of two parts, called antecedent and conclusion, and usually is denoted by "Antecedent -> Conclusion". A relation is suppoed to hold between them. The approach to represent rules is to use the coimplication $(P1,,Pn) \vdash_{(d)} P$ and, instead of finding a coimplication on the closed interval [0,1], the goal is to find the pieces of coimplication of the predicate on partitioned intervals Ii, i=1,n.

The interval [0,1] of the antecedent will be divided into {Ii}i=1,n. As in facts, we wish to find every possible asymmetrical difference between Ii and some values in the closed interval [0,1] of the conclusion.

DEFINITION 5:

Let Di = {d:d=ci - ai, ai$\in$Ii and ci is every possible truth degree of conclusion from ai}. Then di is called an approximate asymmetrical difference on Ii and is defined by di = (inf(Di) + sup(Di))/2 if sup(Di) - inf(Di) $\leq$ 0.5. Let SDi = {sd:sd= ci - ai , ai$\in$Ii and ci is every possible truth degree of conclusion from ai}. Then sdi is called a symmetrical difference on Ii and is defined by sdi = sup(SDi) if sdi $\leq$ 0.5.

The structure of a rule in the knowledge base is constructed by (A,C,[{(Ii,di,sdi)}i=s,t],b,B).

Before we discuss how to resolve a conclusion with the truth interval, let's consider the conjunction operation and the disjunction of the propositions on the antecedent and splitting scheme of the truth interval into partitioned truth intervals.

REFERENCES

Bandler, W. and L.J. Kohout. 1980. Fuzzy power sets and fuzzy implication operators, Fuzzy sets and systems 4(1), July, 13-80.

Bandler, W. and L.J. Kohout. 1984. The Four Modes of Inference in Fuzzy Expert Systems,Cybernetics and Systems Research 2, R. Trappl(ed.), North-Holland.

Dubois, D. and H. Prade. 1980. Fuzzy sets and Systems:Theory and Applications, Academic press, NY.

Hall, L.O. 1986. A Methodological Approach to a Re-Usable Fuzzy Expert System, Ph.D Dissertation,Dep. of Comp. Sci., Fl. St. Univ., Tallahassee, FL.

Kandel, A. 1986. Fuzzy Mathematical Technics with Applications, Addison-Wesley Pub. Co..

Kohout, L.J. and W. Bandler. 1982. Fuzzy Expert Systems, Proceedings Second Technical Conference of the British Computer Society's Expert System Specialist Group, Brunel Uni., Runnymede, England.

Oh, Kyung-Whan. 1988. New methodologies in the design of a general purpose fuzzy expert system: Applications with AI Based Precipitation Retrieval Designed for Satellite Microwave Measurements, Ph.D. Dissertation, Florida State University, Tallahassee, FL, U.S.A., December.

Yager, R. 1985. Inference in a multivalued logic

Zadeh, L. A. 1983. The role of fuzzy logic in the management of uncertainty in expert systems, Fuzzy Sets and Systems, 11, 199-227.

Zadeh, L. A. 1986. Outline of a theory of usuality based on fuzzy logic, A. Jones et al(eds.), Fuzzy set theory and applications, 79-97, D. Reideh Publishing Co.

A BI-DIRECTIONAL INFERENCE ENGINE USING AN OBJECT-ORIENTED APPROACH

Taha A. Sidani and Avelino J. Gonzalez
Department of Computer Engineering
University of Central Florida
P.O. Box 25000
Orlando, Florida 32816

ABSTRACT

An inference engine is the mechanism by which an expert system program uses the domain specific information in the knowledge base to solve a problem at hand. Most of the current systems apply one of three control strategies to perform the reasoning process. These strategies are better known as forward chaining, backward chaining, and hybrid control. However, some of these systems restrict the user by confining him/her to use only the type of control provided.

This paper describes the research aimed at designing a prototype shell that is flexible and simple to use. The form of inferencing mechanism within this system easily permits the reasoning to be performed using any of the types of controls mentioned previously. The user has the capability to select any of the strategies without having to reform the structure of the knowledge base. This system was implemented using an object-oriented approach.

INTRODUCTION

The inferencing methodology behind several well known expert systems and shells were researched. These include MYCIN (Shortliffe 1976), PROSPECTOR (Duda et al. 1979), and PERSONAL CONSULTANT PLUS by Texas Instruments. Each of the systems follow a strict inferencing strategy. Whether it is forward, backward or mixed the user is limited to the method in which the knowledge engineer represented the expert knowledge. If a section of this knowledge is meant to be processed in a forward manner, the knowledge engineer must encode this information using forward chaining rules. After the creation of the domain specific knowledge base, the reasoning process can only perform in a manner dependent on the structure representation of this information. The user is restricted to the method available. He/she cannot configure the system to perform the inferencing process using any of the other methods. In order to provide this kind of capability, the knowledge engineer must reorganize the knowledge to handle this change.

To allow this kind of flexibility, a system was designed to help the knowledge engineer represent the knowledge in a form independent of the control strategy. This system gives the user the capability to select the method of control without having the need to manipulate the knowledge in a different way. The implementation of this system was done on the Symbolics LISP machine using an object-oriented approach.

OBJECT-ORIENTED APPROACH

The object-oriented approach was well suited for implementing this type of control scheme. Different types of objects were used to describe the nodes and rules that constitute the inference net. Generally, in an object-oriented approach, each object modeling a real-world entity can perform actions associated with its type. The actions of each object are defined using generic functions that are associated with each type of object. Flavors is the object-oriented programming system provided on Symbolics computers. The data type, Flavors, was used to define the structure of each object. The implementation of each object is performed by making an "instance" of a defined structure, a flavor. The generic functions that perform the specific actions for each type of object are defined as "methods." The state of each object is stored using instance variables associated with each flavor.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

KNOWLEDGE REPRESENTATION

Three types of flavors were used to represent the knowledge of the designed system (see Figure 1). These structures model the nodes, rules and parameter-values. An object of type node describes one piece of information, like temperature. This object can have different values which characterize its varied states. Each node value is formed by making an instance of the parameter-values flavor. An example of this instance would be values for the above mentioned temperature node, such as low, warm, or high. The instance variables associated with the parameter-value flavor capture its measure of belief, disbelief and confidence about the truthfulness of its value. On the other hand, rule objects are those instances that hold the links of the inference net. They attach the different related nodes together and allow the propagation of belief to traverse down the network.

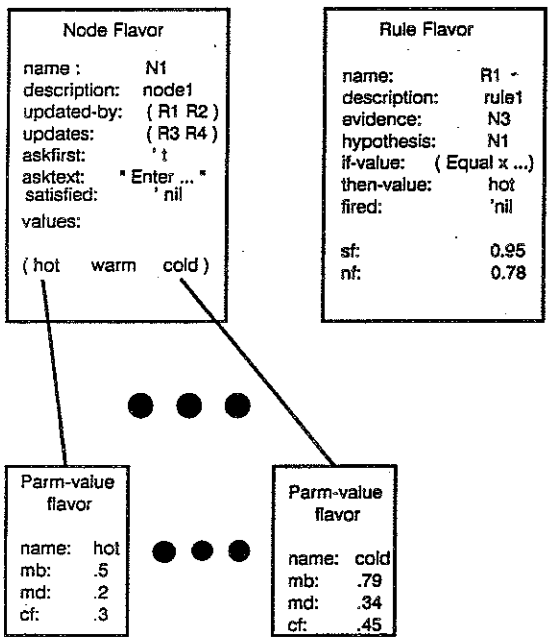


Figure 1. Structure of the Flavor Objects.

The propagation of belief is similar to that used by MYCIN (Shortliffe 1976), and in PDS (Fox et al. 1983). Overall, each node has associated with it a measure of belief (MB) and a measure of disbelief (MD). The difference between the MB and the MD constitutes the confidence factor (CF) for this node. The rules also carry belief and disbelief factors known as sufficiency function (SF) and necessity function (NF), respectively. When a rule fires, it propagates belief or disbelief down to the downstream connected node, also called hypothesis. The updating of the MB, MD, and CF for the hypothesis node are calculated using the following formulas:

$$(3.1) \quad MB = MB + (1 - MB) * SF * CFevid$$

$$(3.2) \quad MD = MD + (1 - MD) * NF * abs(CFevid)$$

$$(3.3) \quad CF = MB - MD$$

The CFevid used in both equation (3.1) and (3.2) represents the confidence factor of the node mentioned in the antecedent part of the rule, also known as the evidence of the rule. The confidence factor for each node can range from -1 to 1, whereas the MB and MD range from 0 to 1. If the CFevid is positive, the rule will only update the MB and CF of the hypothesis using equations (3.1) and (3.3). On the other hand, the rule will revise only MD and CF of the hypothesis if the CFevid happens to be a negative number.

SYSTEM IMPLEMENTATION

The system was developed using the Symbolics LISP machine. The use of menus and mouse sensitive icons make the system user friendly and easy to use. Error checking is performed at all times making the system intolerable for wrong input. The knowledge engineer can simply build the domain knowledge by following prompts and entering responses using both the keyboard and the mouse. It is totally menu driven. Only one type of rule is used by the system, so the knowledge builder does not have to worry about whether he/she used the correct type of rule to perform the required control. The friendly user interface of the system makes it possible for any person with little or no experience to enter the domain specific information into the knowledge base and perform the inferencing process using any of the three strategies mentioned. The developed system is divided into three major parts: Creation of the Knowledge Base, User Support, and Inference Mechanisms.

Creation of the Knowledge Base

This section of the system allows the person to enter the expert knowledge into the knowledge base. Since the information is represented as rules and nodes, the knowledge base builder will require the capabilities to add, edit and delete. With one click on the mouse, he/she can select to perform any of the above

mentioned functions. The **add** function will cause another mouse sensitive submenu to appear on the screen that allows the addition of either rules or nodes. After the addition of a particular node or rule, an instance of that object is created and stored under the name given by the user.

Selecting the **edit** or **delete** function also puts the user in different menus and performs a check on the existence of the chosen object. During the editing process, only certain instance variables, slots, of either nodes or rules can be updated by the user. The system takes care of revising the rest. The delete function reacts differently when trying to delete a node than a rule. Before the deletion of either, it displays for the user the current nodes and rules available. It always permits the user to delete present rules. On the other hand, only those nodes that are not connected to any other rules can be deleted. Otherwise, the connected rules must be deleted before the user can delete the node. Error checking is performed throughout the creation processes. For the ease of creating a large knowledge base, each submenu stays on the screen to allow multiple addition, deletion or editing operations.

User Support Facilities

These support facilities help simplify the operations for creation of the domain knowledge by giving the user the ability to view the current status of one or more objects. One of the operations available displays all of the instance variables of any currently present object. This is important to give the person running the program an inside look into each object. Another useful function is the power to inspect the names of all of the objects present. To start with a fresh knowledge in the knowledge base, a utility is required that can erase any instance of the old information. This purpose is provided by the **clear** operation. But in order to have accurate results for multiple consultations on the same domain in the knowledge base, the nodes and rules must be reinitialized to their default values before each run. Selecting the **reset** provides this without altering the structure of the information. In other words, the nodes and links are not changed. Along with these support facilities is a debugging aid known as **trace**. The trace is a very powerful function, since the user will be able to view the name of each active node during the execution. This is critical if one likes to examine the flow of reasoning performed by the program under different control strategies.

Inference Mechanisms

Three operations make up the heart of the inferencing part of the system. These are configuration, execution and output. The system has the capability of operating under forward, backward or mixed (hybrid) control mechanism. The user can select the method of operation at any time before performing the execution. Unless chosen otherwise, the program will continue to run using the same control that was configured last. When the forward mechanism is chosen, the system will start the reasoning with the top level nodes of the inference net. In other words, it processes from the initial data in the upper part of the network to the goals at the bottom. The system can also operate in the opposite manner when a backward chaining strategy is requested. Here, it begins its flow of inference from the goals to the needed data trying to verify each goal. Whereas, if the hybrid control is selected, the program will prompt the user for some volunteered information about the problem being solved. This information will give the system some hints on where to start reasoning. Each piece of information given will relate to the value and confidence factor of a certain node in the network. The system will keep track of these nodes and use them to trigger its hybrid control. This strategy will declare the node with the highest CF active and start the reasoning from this point. It will backward chain trying to "satisfy" the current node, and then forward chain once that node is satisfied. The process is continued until the list of active nodes is empty.

Using the object-oriented approach, each object node in the network has associated with it a generic function that can perform the required action needed when the node is active. This generic function is defined using the defmethod for the node flavor. The execute function has the overall control on selecting the current active node. It starts out with a list of possible nodes. This list is composed of the nodes that were hinted by the user during the initial startup of the hybrid control state. Once a node is selected, becomes active, the appropriate method linked to its flavor will take part. Generally speaking, the method defined for the node flavor will perform the actual "thinking" for the active object node and decides whether it needs to move forward or backward. The active object starts its action by first checking if it has been satisfied. Once satisfied the node will attempt its forward chaining, unless it happens to be a goal node, in which case it will call the execute function recursively with the rest of the possible active nodes. If found unsatisfied, the node will check itself for the capability to ask the user

questions before trying to backward chain for additional belief.

The output function will finally come into action to select those meaningful goals and sort them using the highest confidence factor criteria. The name, description, and the CF for the sorted goals will be displayed for the user to analyze. For an illustration of the previously described menus and functions, see Figure 2.

CONCLUSION

The limitations behind a system using only one control strategy, were discussed in detail. The system implemented by this thesis was developed with an objective to overcome this limitation. The program can perform either forward chaining, backward chaining, or hybrid inferencing and the user is given the opportunity to select which type of reasoning the program should use during a consultation. The knowledge base is not required to be restructured between selections of different reasoning modes. The implemented system is very user friendly and menu driven. Exhaustive error checking was performed during every user/program interaction.

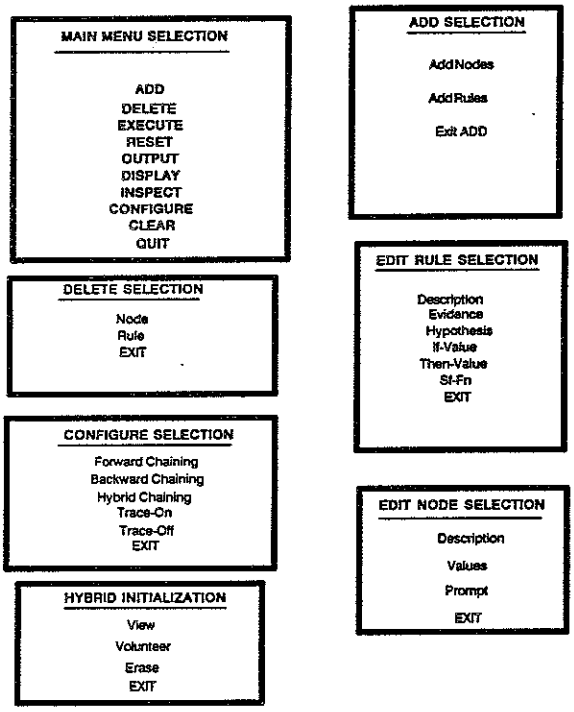


Figure 2. Menus Provided by the System.

The three different control strategies were tested and compared by applying them to the same knowledge base. The implementation of the trace function provided a means to run executions of the knowledge base and examine the flow of reasoning.

Different applications are better suited for using one type of inferencing control over another. Providing the opportunity for the user to select from three inferencing strategies, broadens the range of domains in which the system may be utilized.

REFERENCES

Buchanan, Bruce G., and Edward H. Shortliffe. 1985. The context of the MYCIN experiments. In Rule-based expert systems, edited by Bruce G. Buchanan and Edward H. Shortliffe. Reading and Menlo Park: Addison-Wesley Publishing Company.

Duda, Richard O., Peter E. Hart, Kurt Konolige, and Rene Reboh. 1979. A Computer-based consultant for mineral exploration, final report. Artificial Intelligence Center, Computer Science and Technology Division. SRI Project 6415.

Forgy, Charles L. 1982. RETE: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence 19: 17-37.

Fox, Mark S., Simon Lowenfeld, and Pamela Kleinosky. 1983. Techniques for Sensor-Based Diagnosis International Joint Conference on Artificial Intelligence, Frankfurt.

McDermott, John. 1982. R1: A rule-based configurer of computer systems. Artificial Intelligence 19 (September).

Shafer, G. 1976. A mathematical theory of evidence. New Jersey: Princeton University Press.

Shortliffe, Edward H. 1976. Computer-based medical consultations: MYCIN. New York: American Elsevier Publishing Company.

Szolovits, Peter. 1987. Expert systems tools and techniques: Past, present and future. In AI in the 1980s and beyond, edited by W. Eric L. Grimson and Ramesh S. Patil. Cambridge: MIT Press.

Van Melle, William, Edward H. Shortliffe, and Bruce G. Buchanan. 1981 EMYCIN: A domain-independent system that aids in constructing knowledge-base consultation programs. Machine Intelligence 9; Infotech State of the Art 3: 249-263.

Waterman, Donald A. 1986. A guide to expert systems. Reading and Menlo Park: Addison-Wesley Publishing Company.

Waterman, Donald A. and Frederick Hayes-Roth. 1983. An investigation of tools for building expert systems. In Building expert systems, edited by Frederick Hayes-Roth, Donald A. Waterman, and Douglas B. Lenat. Reading and Menlo Park: Addison-Wesley Publishing Company.

Winston, Patrick Henry. 1984. Artificial intelligence. Reading and Menlo Park: Addison-Wesley Publishing Company.

AN APPROACH TO MULTIPLE EXPERT KNOWLEDGE ACQUISITION THE OPERATIONS ANALYST (OPERA) SYSTEM

R. Bruce Hosken
Grumman Corporation
Kennedy Space Center, FL 32899

ABSTRACT

OPERA is an integrated suite of expert systems under development at Kennedy Space Center by a NASA, Grumman, and Mitre team. OPERA analyzes fault events in the Launch Processing System (LPS) computer network and issues fault summaries to assist the test team to troubleshoot problems. OPERA knowledge bases support advisories to a variety of technical users: operations team, test conductors, engineers and technicians. As a result, the practical knowledge of multiple experts is required. This paper describes one approach to the problem of gathering the domain knowledge of multiple experts.

INTRODUCTION

Knowledge acquisition for building expert systems typically involves a single domain expert. This may be satisfactory in a domain where the considered input of one expert encompasses the majority of technical alternatives. His knowledge built into the system knowledge base provides a place to start, but flavors all recommendations with only one point of view based on his expertise and background. The bias of the Knowledge Engineer can compromise the knowledge acquisition process, resulting in a one-dimensional knowledge base.

The OPERA application, which includes multiple systems, requires the capture of a body of knowledge with considerable breadth and volume. No one individual in the LPS support team possesses expertise in all of the required domain specialties (Adler, Heard & Hosken, 1988). Even specialists within a domain have difficulty scoping the volatile nature of the LPS. As a result, an expert system to assist in LPS troubleshooting should be based on inputs from multiple experts.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

THE PROBLEM

The problem, then, is to collect knowledge from many domain experts and combine the varied approaches to problem-solving into one comprehensive knowledge base. The task of the Knowledge Engineer, to interview each domain expert and extract his knowledge, can be extremely time-consuming and may not have the desired result. Meister (1987) suggests this process to be the biggest bottleneck in the development of expert systems. He further states that it is unrealistic to seek to capture an expert's mental processes in a computer. The only way to determine if a system has captured knowledge is to develop it and evaluate its performance against the experts. He indicates the desirability of using several different inputs in addition to those of the system expert. The input of several experts is preferable to any single one. If the Knowledge Engineer is also a domain expert then chances are improved that a useful product may be achieved.

The problem of extracting expert knowledge is well known in the field of Experimental Psychology. Classifications and methods useful in the process are presented by Hoffman (1987). He does not ignore the needs of the system developer, however, and provides practical criteria to compare methods in relation to these needs. After delineating many methods and approaches, he concludes with tabular estimates of the amount of time required to perform the knowledge acquisition process. If all viable steps are employed in the process, the time required is many months. The process is good, but lengthy. A shortcut is needed for OPERA.

Varying approaches to alleviating the knowledge acquisition bottleneck have been taken in expert system projects. The development of SOY-EX, a rule-based expert system to analyze the suitability of data sets for a soybean growth model (Everett and Dankel II, 1988) employed a combination of techniques. An important part of this project was directed to the development and evaluation of knowledge elicitation tools. The tools helped, but again, knowledge acquisition was still a major time-consuming process.

Limited resources available to the OPERA Project necessitated proceeding with system nucleus development, then beginning the task of knowledge acquisition later.

The Knowledge Engineer/Domain Expert also performs tasks of the System Developer and Designer. This allows little time for the lengthy process of interviewing. A method of conducting remote "interviews" has been employed to gather knowledge from multiple experts in parallel with other development activities.

THE OPERATIONS ANALYST APPROACH

In an attempt to address these problems and prepare the Operations user community for the new system, an approach to remote multiple-expert knowledge communication is underway. Several methods are now used in the collection of domain knowledge about the fault events in the LPS computer network. Computer-system troubleshooting knowledge is being collected in a unique process using what is termed a Knowledge Engineering Bulletin. This bulletin, called the OPERA-TIONS ANALYST, is composed of four sections: an Editor's Note, an OPERA Expert System Status Report, a MINDBENDER problem to collect knowledge information, and an analysis of the previous MINDBENDER. Figures 1A and 1B show a typical bulletin. The Editor's Note introduces new expert system concepts and performs the task of marketing the idea and the tool to gain acceptance of the new process by the users. The Status Report provides information on new features incorporated into the system. The MIND-BENDER Problem of the week is the key to collecting domain knowledge about trouble-shooting the computer network. The problem is posed with a realistic operational setting, a valid system error message, and "speculation" on probable causes by "fictitious experts" at the scene. Questions about the situation follow this presentation and are designed to gather as much troubleshooting knowledge as possible in a small space, but allow for individual creativity. Responses are solicited from more than a hundred domain experts working in support roles in the Launch Processing System. As problem solutions are mailed back, they are screened and edited by another domain expert and entered into the knowledge bases.

To make this system work properly, much domain knowledge must be collected in a short time. Compressing the knowledge acquisition process in this manner has its drawbacks just in the normal logistics of mailouts and processing of the returns.

Figure 1A. OPERA-TIONS ANALYST

****** THE OPERA-TIONS ANALYST ******

---- THE LPS KNOWLEDGE ENGINEERING BULLETIN ----

EDITOR'S NOTE: December 18, 1987

A new TROUBLESHOOTING TOOL is coming to the firing rooms!!! This one will knock your socks off..... it is an EXPERT SYSTEM called OPERA. OPERA means OPERations Analyst, and it is designed to make PROBLEM SOLVING tasks easier and faster. This news bulletin will inform you of the status of the OPERA SYSTEM and allow a means for you to feed back some vital information OPERA needs to function properly. WE NEED YOUR BRAINPOWER!!! Part of the task of developing an expert system is KNOWLEDGE ENGINEERING. We must collect the combined knowledge of engineers, technicians, test conductors and OPERations Analysts, to apply to the problems that occur in the LPS firing rooms. YOU ARE THE DOMAIN EXPERTS!!! WE WANT YOUR INPUT AND PARTICIPATION IN THIS PROJECT!!! An OPERA machine cannot replace a single expert supporting the firing room now, or in the future. But it can MAKE YOUR JOB EASIER!!! How easy depends on how much of your EXPERT KNOWLEDGE can be captured in the OPERA Expert System KNOWLEDGE BASES. Part of this news bulletin will be devoted to some real problem solving based on an event that occurred in the firing room. Anyone who wishes to try solving the problem of the week may send their approaches to THE OPERA-TIONAL ANALYST, GTS-675. All inputs will be screened, compiled and incorporated into the OPERA Knowledge Base, if possible. Those of you who have provided the "best" approaches will be mentioned in a subsequent newsletter. Some problems will be easier than others, but none are trick questions. THIS IS A REAL KNOWLEDGE ENGINEERING TASK!!! When you help us in this way, you also help yourselves by making the tool useful. THANKS.....

The Phantom (of the OPERA)

OPERA Project STATUS ----

The design and development of OPERA software is taking place at the NASA Development Lab in the CIF Building. Astrid Heard of NASA TE-LPS-11, Richard Adler of Mitre Corporation, and Grumman's Bruce Hosken have put together an expert system prototype for experimentation and demonstration. Informal demonstrations to select NASA and GTSI management have been favorable. Technical demonstrations to LPS support personnel are planned. Two expert systems, the Real-Time System Error Monitor (RTSEM), and the Problem Impact Analyst (PIA) are currently running under the control of a global OPERA Controller system, on the prototype LISP machine, a TI Explorer. The RTSEM monitors an incoming CCMS System Message stream (simulated), decodes messages and provides the user with message explanations and fault analyses. The PIA will process RTSEM requests for analyzed historical CCMS fault data on recent or recurring types of failures. Two more expert systems, the Configuration Requirements Analyst (CRA) and the Configuration Manager (CM) are scheduled for later development. These will assist in reconfiguration of the CCMS Set.

1

Figure 1B. OPERA-TIONS ANALYST

****** THE OPERA-TIONS ANALYST ******

THE CCMS "MINDBENDER" PROBLEM OF THE WEEK ----

This is it, folks..... the BIG PROBLEM of the week for you to TRY to solve!!! All troubleshooting approaches and solutions (your best guess is as good as any) will be screened by Prof. Schmartz for inclusion in OPERA Knowledge Bases. Go to it, and GOOD LUCK!

PROBLEM #1 ---- Solutions due: January 8, 1988

A CCMS Systems Engineer was attentively monitoring the SM page at the Master Console during nominal operations. Suddenly, without warning, the following message was displayed.....

45/01 FEP 132 (GS1A) ERROR SIGNAL FROM T/R, MDT/CDT PTR.=743C, T/R STATUS REG=94CF, T/R ADRS REG=020170

The Test Conductor on station asked the SE if he thought a HIM Engineer should be called. The SE mumbled unintelligibly as he pondered the problem, so the TC grabbed a passing HIM Engineer. The HIM Engineer said, "I know what that is! The T/R detected a parity error coming from the GSE Ground Data Bus!" The SE went to look up the T/R STATUS (this would be decoded in OPERA) and returned. He said, "That problem is a memory parity error on the T/R transfer of data to memory in the GSE FEP!" Who has the right answer to this problem??? You do, of course... but, why???

1. WHAT ARE THE FIRST THREE (3) THINGS YOU WOULD DO TO SOLVE THE PROBLEM?

2. WOULD YOU RUN DIAGNOSTICS? IF SO, WHICH ONES?

3. WOULD YOU RECOMMEND SWITCHING TO A STANDBY MACHINE? WHY?

PLEASE PRINT YOUR NAME AND MAIL CODE _____

Are you (circle one) an SE HE TC TECH ANALYST SUPV OTHER ?

Prof. Schmartz is awaiting your reply. SEND THIS RESPONSE TO OPERA-TIONS ANALYST, GTS-675

2

The editorial and technical effort that goes into the newsletter consequently cannot also be applied to the interviewing process. The results, however, have been encouraging. Responses seem to be representative of the domain expertise available, and come from multiple experts in multiple domains.

To illustrate how OPERA handles the multiple domain problem, Figure 2 relates the domain expert to the LPS functional task and specifies the OPERA Expert System which assists the expert. The chart shows the Real-Time System Error Monitor (RTSEM) to be primarily an error detection and evaluation tool used by all experts. The Problem Impact Analyst (PIA) relates previously known problems and contains historical problem data, of interest to all experts except the Analyst involved in operational procedures. A Configuration Requirements Analyst (CRA) which assists the operations team to recover and reconfigure the computer network when a failure occurs, is important to the Analyst but not to the engineers responsible for error isolation and troubleshooting functions. A Configuration Management (CM) system assists the Test Conductor and Set Manager to perform recommended actions in the recovery process. The five domain experts and six functional tasks represent the multiple domain nature of OPERA.

THE OPERA RESEARCH ASSOCIATE

Additional domain experts are currently performing single-error analysis of system messages. These experts are compiling knowledge data in a format compatible with the knowledge base structure of frames and slots. At present, more data is gathered using the research associates than with the slower bulletin and MINDBENDER approach. The primary advantage of this approach is the gathering of the knowledge of multiple experts in parallel, with many experts concurrently performing the research and knowledge compilation required to analyze individual error events. Again, the bias of the single Knowledge Engineer is avoided, since these experts perform their own independent research. No transcript interpretation of interviews is necessary, resulting in less misinterpretation of the technical facts. Domain experts can perform the task of error analysis without the pressure of an interview with the Knowledge Engineer.

Expert	Tests: Preps/ Rqmnts	Problem: Detect	Eval	CPU: Stat/Hist	Action: XQTD	Log/Track
Set Mgr	PIA CRA		RTSEM	PIA	CM	PIA
Test Conductor	PIA CRA	RTSEM	RTSEM	PIA	PIA CM	PIA
Analyst	CRA	RTSEM				
Support Engineer			RTSEM PIA	PIA	PIA	PIA
HW Eng (Tech)			RTSEM		PIA	PIA

RTSEM = Real-Time System Error Management
PIA = Problem Impact Analysis
CRA = Configuration Requirements Analysis
CM = Configuration Management

Figure 2. Operations Support Tasks Assisted by OPERA

FUTURE OPERA KNOWLEDGE ACQUISITION

The OPERA Project is only just beginning to build a multiple expert knowledge base. The continuing compilation of a large volume of knowledge data based on input from multiple experts may hinge completely on a faster method of gathering knowledge. Automated forms of knowledge engineering have been tried with some success, depending on the target domain and the degree of complexity of the requisite knowledge. The development of rule-based systems using automated knowledge acquisition methods is currently under experimentation and progress is being made. Applying the repertory grid as a useful tool from clinical psychology to the problem of eliciting data from experts (Ford, 1988) is allowing development of initial production rules to start an ongoing machine-learning process. This is a successful technique in the domain of Radiology.

Kahn, Nowlan and McDermott (1984) developed an automated approach to elicit knowledge in a system called MORE. This system evolved from a handcrafted predecessor, MUD, that used compiled diagnostic knowledge linked to the hypothesis in question. MORE is designed to elicit knowledge which may be used to perform diagnostic tasks, and as such has a function similar to that of the OPERA System. The generation of rules by MORE requires a domain model replete with hypotheses, symptoms, conditions, links and paths. To use these for OPERA could negate the OPERA performance charter to respond with near-real-time fault analyses.

The development of the Expertise Transfer System (Boose, 1984) may have application to the OPERA knowledge problem. Heuristic knowledge is gathered by ETS to construct and analyze an initial set of heuristics and problem parameters. Most of the domain expert interviewing process is eliminated to allow a significant saving in time. Grid methodologies are also applied by ETS to develop a matrix of truth values to analyze in a subsequent step. A future OPERA knowledge tool could incorporate similar featues.

SUMMARY

The combined approach to knowledge acquisition represented by the OPERA-TIONS ANALYST Bulletin and the OPERA Research Associates is an experiment in the field of Knowledge Engineering. While it employs several recognized techniques for gathering domain information it also incorporates those which are breaking new ground in technical communications. The responsiveness of the expert in the field will ultimately determine the effectiveness of this process. As problem solutions arrive, similarities and differences in technical approach can be categorized and compiled before updating of knowledge bases. If several fault-isolation approaches to a particular problem are incompatible, follow-up interviews with selected domain experts can determine the best approach for this situation. Brief interviews with any and all experts who respond with problem analyses can help to complete the communication cycle and add useful additional domain information.

A representative mix of system error analyses is highly desirable in this particular domain, due to the multiple-domain nature of the LPS data processing network. The top-level view of the System Engineer is adequate for an initial determination of the type and criticality of the problem, but when it is time to isolate the failure to a specific component the expert advice of the Field Technician is needed. Hardware failures may best be analyzed by a Hardware Engineer with assistance from a Technician. Compounding the confusion of a real-time analysis are the reconfiguration and recovery requirements of the Computer Operations Crew. All of these points of view need to be addressed when the LPS is processing a Space Shuttle in system testing, or during a Launch Countdown. A primary goal of the OPERA Expert System is to satisfy the needs of multiple experts and users in the launch control room by providing expert consultation or advisories. The Knowledge Base must incorporate multiple alternatives in some cases, when there is a valid reason to do so. Variations in individual error and problem analyses may uncover unexpected heuristic approaches that can save time and troubleshooting resources. The requirement for OPERA to respond in near-real-time, less than five minutes from the time of the error to be analyzed, makes use of an extensive rule-base cumbersome in terms of execution time and processing overhead. Typically, rule-based systems do not attempt to perform a real-time support function. OPERA focuses primarily on top-level heuristic insights into the problem domain that may be quickly displayed to the user.

Acceptance of the expert system and the knowledge that drives it may be at least partially dependent on the goodwill generated by the OPERA-TIONS ANALYST Newsletter. The effectiveness of this knowledge gathering tool will determine to a large extent how well the knowledge base supports the needs of the user. Using the techniques described to focus attention on real-time problems, the OPERA Knowledge Base may be developed to a level of accuracy and robustness suitable for real-time analysis of complex system errors.

REFERENCES

Adler, R.M., Heard, A., Hosken, R.B. 1988. "OPERA- An Operations Analyst for a Distributed Computer System." In Proceedings of the 1st Florida Artificial Intelligence Research Symposium, Orlando, Florida.

Adler, R. M. and B. H. Cottman 1989. "A Development Framework for Distributed Artificial Intelligence." In Proceedings of 5th IEEE Conference on Artificial Intelligence Applications, Miami, Florida.

Boose, John H. 1984. "Personal Construct Theory and the Transfer of Human Expertise." In Proceedings of the National Conference on Artificial Intelligence, Austin, Texas.

Everett, P.A. and D. D. Dankel II, 1988. "A Study in Acquiring Knowledge from an Expert." In Proceedings of the 1st Florida Artificial Intelligence Research Symposium, Orlando, Florida.

Ford, Kenneth M. 1988. "An Approach to the Automated Acquisition of Expert System Rules." In Proceedings of the 1st Florida Artificial Intelligence Research Symposium, Orlando, Florida.

Hoffman, Robert R. 1987. "The Problem of Extracting Knowledge of Experts from the Perspective of Experimental Psychology." AI Magazine, Summer: 53-66.

Meister, David 1987. "Methods of Eliciting Information from Experts." Navy Personnel Research and Development Center, Report NPRDC TN 88-2.

REPRESENTING THE SOCIAL DIMENSION
Design and Methodology* for an Urban Planning System

Evelyn Stiller; Stig Løvstad; Wyllis Bandler;
Institute for Cognitive Sciences and
Department of Computer Science
The Florida State University
Tallahassee, Florida 32306-4019

Vasco Mancini
Architecture and Environments
11 The Avenue
Colchester CO3 3PA
United Kingdom

ABSTRACT

The paper introduces design and knowledge representation issues pertinent to a project in applying artificial intelligence methods to urbanistics. The ultimate goal is to construct a non-traditional model based on individual urban experiences, elicited through various psychological techniques. The model is diagnostically structured, and seeks to enhance quality of life (QOL), as socio-culturally defined. The paper is partitioned into two sections. The first section introduces the project as a whole, defines various knowledge representation structures relevant to the social dimension of the urban experience, and discusses preliminary empirical results. The second section conveys a detailed presentation of fuzzy relational techniques used to analyze portions of the information elicited.

MODEL DESIGN

Introduction

The intent of this paper to discuss an urban model that will express the psychological aspect of the human interaction with the physical urban environment through *personal constructs* or *constructs* as defined by Kelly [Kelly 1955] in his personal construct theory (PCT). These constructs will be elicited from images of specific urban objects, which will comprise the environment, or spatial dimension of the city. Thus, the constructs will act as the basis for the congruence relation between the aspatial structure, expressed as emotions and feelings of the human, with the spatial urban objects as suggested by Mancini and Bandler [Mancini and Bandler 1988]. The elicitation of these constructs from images is particularly appropriate due to the cognitive associations that the images have to emotions, meaning, values, and symbolism of the urban object.

The primary motivation for a highly non-traditional approach to urban modeling lies in the ineffectiveness and lack of credibility presently associated with existing models. A clear break from tradition appears to be called for. A contributing factor to this new approach lies in the "information glut" that we are now experiencing. Given the typical economic or statistical models, it is clear that lack of data is not the problem. A design overview has been conveyed in Stiller et al [Stiller et al 1988], in which the application of the various adopted theories is addressed. The purpose of this paper will be to concentrate on the knowledge representation issues appropriate to this model.

While the urban experience is comprised of many dimensions [Lynch 1981], it will be the social dimension that will be of interest here. The social aspect of the city, by incorporating socio-economic and cultural attributes, portrays a deep urban view, where a relevant economic picture may be presented in conjunction with the dynamic perspective of social interactions. With support of artificial intelligence techniques, a sophisticated decision support environment may be created to define the complex interrelationships that exist in an urban setting.

Knowledge Domains

The model contains four primary knowledge domains. The first domain involves a cultural, socio-economic description of each individual who contributes information regarding the *social network(s)* in which he/she participates. The second knowledge domain centers around the spatial and functional descriptions of the significant points or nodes comprising the individual social network.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

*This research has been supported by NSF grant IST-8604575.

A social network is defined as a set of points of social interaction, called *social nodes*, with socio-cultural characterization. The social network of a given individual is comprised of those places where social interaction takes place with a specific set of people. For example, the place one meets coworkers, such as restaurants, stores, bars, and so on, would comprise the "working social network". The third knowledge domain defines the general socio-economic and cultural information that describes the various sectors of the city. The final and most important domain of knowledge contains the urban residents' mental constructs through which the physical aspect of the city, especially the components of the social network, are experienced and interpreted. These constructs and their interrelationships define a *cognitive structure*, which, in turn, defines the values, symbols, emotions, and feelings critical to the decision support mechanism.

Knowledge Representation Technique

The social dimension is comprised of simple and complex *social elements* and their interrelationships. Two types of complex social elements are defined. Both, as their name implies, are comprised of simple social elements. The first complex social element is the social network. The social network is comprised of a human as well as spatial aspects. The human perspective is defined by the socio-cultural characterizations of the groups which take part in the social network. The spatial aspect is defined by social nodes or districts which comprise the points of social interaction for the urbanites. Therefore the simple social elements comprising the social network are the urban objects, specifically districts and social nodes, and socio-cultural properties of the urban residents. The second type of complex social element is the cognitive structure. This element, as previously described, expresses the mental constructs, representing the simple elements, through which the urbanite interprets the urban experience.

The simple social elements, which are building blocks of the complex social elements, are also independent actors. Three simple social elements have been defined, urbanites, urban objects, and mental constructs. The interrelationship between urbanites and urban objects is essential to the model. This interrelationship is defined by three link types, functional, emotional, and value. These link types are analogous to the *performance dimensions* defined by Kevin Lynch [Lynch 1981] in which the spatial form is evaluated in terms meeting the needs along the various dimensions. The functional and emotional relationships are derived directly from the constructs, since they are evoked from specific social nodes or districts. Value links are elicited through a technique developed by Hinckle [Banister and Fransella 1986], called *laddering* from the concepts. This technique allows an association from the social nodes to quality of life.

Social Policy

The fundamental goal of this model is to provide the foundation for urban decision support by surfacing dysfunctional relationships between the urban residents and the city. This dysfunctional relationship may be expressed not only in terms of problems between the urbanite and the objects that comprise the city, but may find expression as unhealthy associations between various groups of urban residents. In order to accomplish this, it is essential to define dysfunctional and therefore functional relationships between the social elements of the model. These concepts are what form the social policy, which is the basis for diagnosing the city, and the heart of the inference mechanism.

The social policy, is built hierarchically. It consists of several high level normative statements, supported by hierarchically subordinate statements, serving to substantiate the high level statement. As one would suspect, these subordinate statements may also be supported by increasingly subordinate statements. The policy is represented by English language statement conveying the meaning of the statement for the human interface to the system. Associated with the English language statement is a procedure for substantiating or refuting that portion of the policy. All statements must be weighted as to the necessity, sufficiency, or desirability of its substantiation.

Fuzzy Attributes

The use of fuzzy attributes in the definition of the simple social elements avoids data intensive, costly, and perishable attribute definitions. Verbal descriptors will be used to express quantification where possible. For example, a district may be classified as highly residential and minimally commercial, while the economic level of the residents may be described as lower middle class. The relationship of constructs to system primitives may also be described in fuzzy terms. For example, the urbanite may select the construct " repelling" in regard to some urban objects. The construct may be interpreted in term of system primitives as "minimal fear, and highly unattractive ".

Empirical Results

Preliminary testing has taken place involving Kelly's PCT. This discussion involves the analysis of constructs elicited from pictures of various house types. The analysis has provided insight into the establishment of the aspatial to spatial congruence relation, utilizing the cognitive structure. The analysis involves the application of relational techniques (to be addressed) upon Kelly's reper-

tory grid. Associations between the constructs evolved through the analysis in the form of *equivalence classes.* The associations indicate similarity in application to the objects (houses) from which the construct elicitation was performed. The equivalence classes maintained semantic validity. In other words, constructs which comprised a given equivalence class could be logically associated by meaning.

The most useful results are derived when classifying the constructs in terms of the following *construct types* 1) (physically) descriptive; 2) value; 3) emotion; 4) use; 5) socio-cultural. It is the equivalence classes comprised of a variety of construct types that are of most interest. Further, the relationship between specific construct types is of particular interest. For example, the associations between socio-cultural and value constructs, such as poor and unimportant, respectively, give insight into intercultural attitudes. Other associations, such as between descriptive and emotional constructs, such as decorative and happy, relate emotional states to physical attributes. Similarly, value constructs associated with descriptive constructs may convey socio-cultural preferences for various physical attributes. Somewhat less significant, but worth noting, are use and descriptive construct associations. These potentially indicate utilitarian physical features. Equivalence classes of a single construct type should not be entirely dismissed. They, however, tend to confirm semantic validity rather than provide deeper insight. For example, associations between the emotion constructs, happy, cheerful, and joyful are not of great use in the model.

It seems that the assessment of these equivalence classes will be instrumental in deepening the knowledge elicitation process, especially the laddering process. The highlighting of important construct interrelationships will allow the laddering session to be more focused. It appears that much complimentary information is derived from both of these processes.

TRISYS/TRIMOD: A Knowledge Engineering Tool

Making computers behave in ways we would consider intelligent in human beings is one of the concerns of Artificial Intelligence. One attempt at meeting this expectation is the research and development done in the field of expert systems where a computer acts as if it had the knowledge of a human expert in a certain domain. In this field, knowledge elicitation and organization is one of the major tasks facing the knowledge engineer. TRISYS/TRIMOD is used as an aid to solving the problems inherent in these tasks, and is based on the concepts of fuzzy relational theory.

Fuzzy set theory had its official beginnings in 1965 with Lotfi A. Zadeh's by now legendary paper [Zadeh, 1965]. Since then thousands of works dealing with this topic have been published, and hundreds of researchers are still investigating both the theory and its applications. Zadeh believes that the ultimate aim of fuzzy set theory is to represent how the human mind perceives and manipulates information [Dubois and Parade, 1979]: "Indeed the pervasiveness of fuzziness in human thought processes suggests that much of the logic behind human reasoning is not the traditional two-valued or even multivalued logic, but a logic with fuzzy truths, fuzzy connectives, fuzzy rules of inference."[Zadeh, 1973]

In response to the increasing difficulties in comprehending the consequences of changes caused by the complex interactions between man, nature, and man-made artificial systems, and in order to analyze these systems, understand their dynamics, and influence their behavior, Bandler and Kohout provide a framework of a *theory of action* [Bandler and Kohout, 1980a], based on the *activity structures* [Kohout, 1988] of Kohout, in which each participant must be seen as a unit possessing one or more of the features: cognitive ability, decision ability, action ability. The general methodological and computational framework is provided by the theory of fuzzy relational products.

This section of the paper deals specifically with the triangle subproduct between fuzzy relations, leading to order-like relations and their implementation into a working system for analyzing among others psychological data, whether clinical or not, of the sort derived from the Semantic Differential [Osgood et.al., 1957] or from Repertory Grids [Fransella and Bannister, 1977].

Relation Theory

In the triangle subproduct x stands in the relation $R \triangleleft S$ to z if and only if, for each y, xRy implies that yRz. For the matrices,

$$(R \triangleleft S)_{ik} = \min_j(R_{ij} \rightarrow S_{jk}).$$ [Bandler and Kohout, 1988a]

The usual way to study a fuzzy relation is through consideration of its α-cuts. The α-cut of the fuzzy set A, for any $\alpha \in]0,1]$, is the crisp set A_α of all those elements of X whose membership degrees in A are at least as great as α:

$$A_\alpha = \{x \in X | \mu_A(x) \geq \alpha\}.$$

In matrix terms we have,

$$(R_\alpha)_{ij} = \begin{cases} 1 & \text{if } R_{ij} \geq \alpha \\ 0 & \text{otherwise.} \end{cases}$$

[Bandler and Kohout, 1988b]

Using the triangle subproduct relation requires, in calculating, the use of a fuzzy implication operator [Bandler and Kohout, 1980b,c]. The system allows the user to choose between some or all of the set L1 to L8, including L4' and L5.5, as defined below. Our research presently makes extensive use only of operators L5 and L6, which are the Łukasiewicz and Kleene-Dienes operators respectively.

The table below indicates the program given identifier, followed by the literature's identifier, followed by its actual name and definition.

L1: S# Standard Sharp
$a \rightarrow_1 b = \{1$ iff $a \neq 1$ or b=1; 0 otherwise $\}$

L2: S Standard Strict
$a \rightarrow_2 b = \{1$ iff $a \leq b$; 0 otherwise $\}$

L3: S* Standard Star
$a \rightarrow_3 b = \{1$ iff $a \leq b$; 0 otherwise $\}$

L4: G43 Gaines 43
$a \rightarrow_4 b = \min(1, \frac{b}{a})$

L4': G43' Modified Gaines 43
$a \rightarrow_4 b = \min(1, \frac{b}{a}, \frac{1-a}{1-b})$

L5: Ł Łukasiewicz
$a \rightarrow_5 b = \min(1, 1\text{-}a+b)$

L5.5: KDL Kleee-Dienes and Łukasiewicz
$a \rightarrow_6 b \leq m1 \leq a \rightarrow_5 b$
$\max(1\text{-}a,b) \leq m1 \leq \min(1, 1\text{-}a+b)$

L6: KD Kleene-Dienes
$a \rightarrow_6 b = (1\text{-}a) \vee b$

L7: EZ Early Zadeh
$a \rightarrow_7 b = (a \wedge b) \vee (1\text{-}a)$
$= (a \rightarrow_6 b) \wedge {}_k a$

L8: W Willmott
$a \rightarrow_8 b = ((1\text{-}a) \vee b) \wedge (a \vee (1\text{-}b) \vee (b \wedge (1\text{-}a)))$
$= (a \rightarrow_7 b) \wedge {}_k b$
$= (a \rightarrow_6 b) \wedge {}_k a \wedge {}_k b$

The System

At its inception, the TRISYS/TRIMOD system introduced a new mathematical technique for analyzing grid matrices, allowing for the detection of dependencies, hierarchies, and partial orders in the concepts being studied.

In the system, construct to construct relations give the degree to which the use of a row-head construct implies the use of a column-head construct. The transitive closure of the original is taken, which provides an index of transitivity which in turn tells us how close the original is to this closure. Further analysis is then performed upon the closure, but tabs are continually kept on the index.

We study the fuzzy relation by taking α-cuts at its height, half-way down to its mean, at its mean, and half-way down from its plinth. Each cut is a crisp relation and if its index of transitivity falls below a specified criterion, the particular study ceases.

There is a tendency for all the operators to give, in cuts near their means, networks which are partial orders, or which, with the fusion of structurally equivalent elements, become partial orders. " An order is a relation R from a set of elements E, to itself, that has the properties of reflexivity, transitivity, and antisymmetry."[Lamm, 1989] These properties permit the partitioning of the elements of E into n exclusive subsets. The resulting subsets have a one-way hierarchical relation which can be shown pictorally by a Hasse diagram.

The main purpose of the TRISYS/TRIMOD sytem is to aid the knowledge engineer in gathering cognitive data. Previously it was used by Wilcox as a means of analyzing repertory grid data on social skills training. Wilcox believed that "the most significant contribution made by this analysis is a convenient way of deriving chains of implications of construct poles" [Wilcox, 1981]. In current research TRISYS/TRIMOD is being used extensively for analysis of personal and social constructs and their relationships to generic and specific locational nodes in the urban environment.

Input to TRISYS/TRIMOD is given in the form of matrices of attributes possessed by entities, where entity "e" possesses attribute "a" to a certain extent. This extent is a fuzzy degree, and is stored in element (e,a) of the matrix. Given two matrices R and S (where R = S is possible), the system calculates fuzzy dependencies of the form $R \lhd S^{-1}$ and $R^{-1} \lhd S$, where $\lhd$ is the triangle subproduct relation discussed earlier. The results of the calculations are presented in tabular form. The initial matrix with labels along with the various calculated matrices are presented, and for each calculation the various implications are shown along with the relation's properties.

The TRISYS/TRIMOD package has been implemented in both pascal and Modula2. A graphics package has been developed [Lamm, 1989] to allow for the viewing

and printing of the various Hasse diagrams which show, pictorally, the relationships among the data. Currently an extensive front-end is being developed for use with the Modula2 version.

REFERENCES

Bandler, W. and L.J. Kohout. 1980. *Fuzzy Relational Products as a Tool for Analysis and Synthesys of the Behavior of Complex Natural and Artificial Systems.* Fuzzy Sets: Theory and Applications to Policy and Information Systems, 341-367.

Bandler, W. and L.J. Kohout. 1980. *Semantics of Implication Operators and Fuzzy Relational Products.* Intl. J. Man-Machine Systems. 12. 89-116.

Bandler, W. and L.J. Kohout. 1980. *Fuzzy Power Sets and Fuzzy Implication Operators.* Fuzzy Sets and Systems. 4. 13-30.

Bandler, W. and L.J. Kohout. 1988. *Mathematical Relations.* International Encyclopedia of Systems and Control. Pergamon Press. New York.

Bandler, W. and L.J. Kohout. 1988. *Special Properties, Closures and Interiors of Crisp and Fuzzy Relations.* Fuzzy Sets and Systems. 26. 317-331.

Bannister, Don and Fay Fransella. 1986. *Inquiring Man: The Psychology of Personal Constructs, Third Edition.* Croom Helm, London.

Dubois, D. and H. Parade. 1979 *outline of Fuzzy Set Theory: An Introduction.* Advances In Fuzzy Set Theory And Applications, 27-48.

Fransella, F. and D. Bannister. 1977. *A Manual for Repertory Grid Technique.* Academic Press. London.

Kelly, George A. 1955. *The Psychology of Personal Constructs.* Norton, New York, N.Y.

Kohout, L.J. 1988. *Perspectives on Intelligent Systems: a Framework for Analysis and Design.* Kogan Page. London.

Lamm, C.J.W. 1989. *General Operation of the Program to Draw Hasse Diagrams. (Hasse Diagram Program).* Presented at Second Annual Florida Artificial Intelligence Research Symposium. Orlando. Florida. April 3-7.

Lynch, Kevin. 1981. *Good City Form.* M.I.T. Press, Cambridge MA.

Mancini, Vasco and Wyllis Bandler. 1988. *Congruence of Structures in Urban Knowledge Representation.* Proceedings of the IPMU. IPMU'88 (International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems. Urbino,Italy.

Osgood, C.E.; G.J. Suci; P.H. Tannenbaum. 1957 *The Measurement of Meaning.* University of Illinois Press. Urbana. Illinois.

Stiller, Evelyn, and Wyllis Bandler, and Vasco Mancini. 1988. *Expert System Design for an Advisory Urban Planning System.* Proceedings of the SRIC. SRIC'88 (International Conference on Systems Research, Informatics, and Cybernetics in Knowledge-Based Systems). Baden-Baden, W.Germany.

Wilcox, T. 1981. *The Usefulness of Fuzzy Set Analysis With Repertory Grids in Measures of Change in Social Competence.* Applied Systems Research and Cybernetics, 3020-3026.

Zadeh, L.A. 1965. *Fuzzy Sets.* Inf. and Cont. Vol. 8, 338-353.

Zadeh, L.A. 1973. *Outline of a new approach to the analysis of complex systems and decision processes.* I.E.E.E. Transactions. S.M.C. Vol. 3, 28-44.

APPLIED AUTOMATED KNOWLEDGE ACQUISITION FOR GREENHOUSE CONTROLS

Jeffrey S. Nelson and Douglas D. Dankel, II
Department of Computer and Information Sciences
CSE, University of Florida, Gainesville, FL 32611

Pierce H. Jones and Fedro S. Zazueta
Department of Agricultural Engineering
Rogers Hall, University of Florida, Gainesville, FL 32611

ABSTRACT

This paper describes the development of a knowledge acquisition tool designed to automate the process of developing expert systems for controlling greenhouse environmental conditions. The tool first elicits baseline information about current greenhouse operation practices. It then inquires about deviations from baseline conditions in order to build a set of rules that allow control of operations in commercial greenhouse production.

INTRODUCTION

Misting systems are commonly used in commercial greenhouse propagation to water plants and maintain a high relative humidity. The proper amount of misting is critical. Newly propagated plants are particularly vulnerable to water and temperature stress and require frequent mistings. Excessive misting can promote disease and leach nutrients from the potting media while inadequate misting can desiccate young plants. Both crop yield and production time are adversely affected by an inappropriate misting program.

Most commercial greenhouses use simple timers to control the frequency of the misting events and the duration of each misting. Because of competing managerial demands growers are often constrained in the amount of time they can devote to monitoring the misting program. In practice, most growers employ a relatively static misting program, generally checking settings once a day or less.

A misting program should respond to changing environmental factors. As an example, on a warm, sunny, dry morning, a grower may set a generous misting program. But if a heavy afternoon thunderstorm occurs, temperatures will decrease, light levels will decrease and relative humidity will increase. If the misting program is not adjusted to accommodate these changes, then over-watering occurs and the risk of water-related problems increases.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

There is a complex interplay among these ambient conditions that are important in determining an appropriate misting program. Light levels affect temperature, temperature affects relative humidity, heating events affect both temperature and relative humidity. In addition, the grower must consider the special misting requirements dictated by the plant variety and the age of the crop.

Algorithms for dealing with all of these variables do not exist. Most growers have a general "sense" of how these factors interact and how these factors bear on the determination of an appropriate misting program. They have, through trial and error, developed expertise and often express this expertise in the form of heuristics. Jacobson et al. (Jacobson et al. 1989) demonstrated that these heuristics could be extracted from an expert grower and employed, using expert system technology, in a system that dynamically controls misting events. In this study, a research horticulturalist experienced in plant propagation provided expertise for propagating Peperomia. From this, the knowledge engineer developed a knowledge base. Using commercially available hardware and software, an expert misting control system was developed and tested in a research greenhouse for 30 days.

As with any expert system, the knowledge acquisition phase (the development of the knowledge base) was critical in the Jacobson et al. study. It is likely that this process could be repeated for acquiring expertise in the propagation of another plant species.

In developing dynamic misting controls, the *technical feasibility* of extracting and employing an expert grower's knowledge has been demonstrated. It is now time to address the issue of *practicality*. Research into the development of expert systems has shown that historically the knowledge acquisition phase of expert system development consumes most of the time and effort expended (Prerau 1987; Hoffman 1987; Olson and Rueter 1987). Few growers have the resources (time, expert system expertise, and money) to devote to manually developing and employing a dynamic misting control system tailored to each of the crops they produce.

If the knowledge acquisition phase can be automated, and a "standardized" knowledge base produced, then the cost of developing dynamic misting control systems will be less prohibitive to growers.

OBJECTIVE

The primary objective of this study is to develop a system that will elicit misting knowledge from an experienced commercial grower, and, from this knowledge, synthesize a dynamic control program that will successfully employ the grower's misting strategy. A key feature of this system is transferrability, i.e. the system should be usable by ten different growers propagating ten different plant varieties. The system will be deemed successful if the misting control program synthesized by the system can produce a crop of quality comparable to (or better than) that which the expert grower can produce using existing timer controls.

MISTING

The MISTING expert system (Jacobson et al. 1989) is a dynamic and autonomous misting cycle controller designed to implement an expert grower's misting strategy for the propagation of Peperomia. In examining the knowledge base, five stages were identified between the time the parameters are "read" and the decisions are implemented.

In the first stage, the "apparent crop age" is determined. The "apparent crop age" is defined as an "adjusted chronological age" that is adjusted for the average cumulative greenhouse temperature during the propagation period and the actual crop age. Temperature has a significant impact on crop development. Cooler temperatures result in a longer development time. To adjust for the variability in temperature, the expert provided a table (table 1) from which "adjustment factors" are derived. The adjustment factor is multiplied by the actual crop age to compute the apparent crop age.

Table 1: Apparent Crop Age Adjustment

Avg Cum Temp (°C)	Adjustment Factor
< 10	42/80
10 < avgtemp < 21	[42+(temp-10)*3.45]/80
> 21	1

In the second stage, baseline frequency and duration settings are derived from table 2 using the apparent crop age computed from stage 1.

Table 2: Baseline Frequency and Duration

Apparent Crop Age	Frequency (minutes)	Duration (seconds)
1	5	6
2	10	6
3	15	6
3 < age < 42	7+2.7*age	4.2+.62*age
42 <= age	120	30

In the third stage, ambient temperature and relative humidity inputs are used to adjust the baseline frequency setting. The expert provided a table (table 3) from which "adjustment factors" are derived. The adjustment factor is multiplied by the baseline frequency to compute a new frequency setting.

Table 3: Frequency Adjustment Factors

Relative Humidity	Current Greenhouse Temperature (°C) < 13	13-16	16-18	18-21	21-27	27-31	> 31
> 95	.15	.30	.40	.50	.60	.70	.80
85-95	.25	.45	.50	.60	.70	.75	.85
75-85	.45	.60	.60	.65	.75	.80	.90
65-75	.60	.70	.70	.70	.80	.85	.92
50-65	.70	.80	.80	.80	.90	.90	.95
35-50	.75	.85	.90	.90	.95	.95	.97
> 35	.80	.90	1.00	1.00	1.00	1.00	1.00

In the fourth stage, the ambient light input is used to adjust both the frequency and duration settings. The expert provided a table (table 4) from which "adjustment factors" are derived. The adjustment factor is multiplied by the frequency and duration to compute new settings.

Table 4: Frequency and Duration Adjustment Factors

Light Level (W/m^2)	Adjustment Factor
0 - 7	.00
7 - 33	.50
33 - 67	.63
67 - 133	.75
133 - 200	.88
> 200	1.00

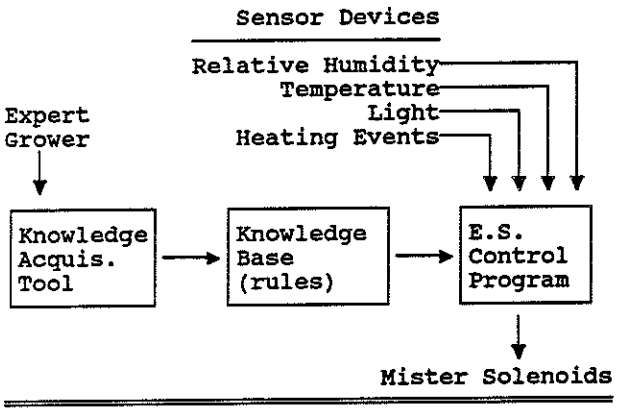
In the fifth stage, inputs are examined to detect the occurrence of a heating event. To counteract the drying effects of heating, an overriding frequency setting of 15 minutes and duration setting of 6 seconds is made.

These tables were translated into the rules which comprise the knowledge base for a backward-chaining inference engine. These tabular values are specific to the propagation of the Peperomia plant species and were provided by a research horticulturist expert in the propagation of Peperomia. The knowledge was acquired manually via interviews with the expert.

TOOL DESIGN CONSIDERATIONS

This knowledge acquisition tool will act as a stand-alone front-end to an expert system similar to MISTING. This tool will transfer and transform crop propagation expertise from commercial growers into a set of rules comprising a knowledge base. Figure 1 illustrates the role of the tool in relation to the other components of the misting control system.

Figure 1



The tool will "interview" the expert and elicit the required knowledge. Its desired characteristics have been defined. The tool should:

- be easily learned by growers who are not particularly computer- or expert system-literate
- adapt itself to a communication modality comfortable to most commercial growers
- actively engage the grower in a "dialogue" from which the knowledge will be extracted

APPROACH

In approaching this problem, the MISTING knowledge base and the research horticulturist's "view" of the process of propagating Peperomia was examined. The Peperomia expert had rather specific knowledge to be used in determining misting frequency and duration. But would commercial growers of other plant varieties "view" the propagation process in a similar manner? As a starting point, some base of commonality between the Peperomia expert's "view" and the "views" of other growers was explored. Interviews with other research horticulturists were conducted.

The Peperomia expert felt that a primary consideration in determining misting settings was the stage in development of the crop. In MISTING these stages are explicitly defined in table 2. In exploring this developmental stage approach, other experts confirmed that most growers do recognize stages in the development of the crop. If pressed to do so, it was felt that most growers could characterize crop development in discrete stages.

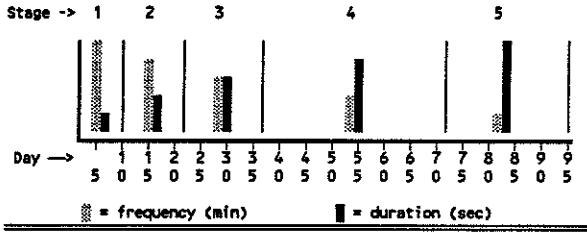
The tool, then, should prompt the grower to recognize and identify distinct stages in crop development along with baseline misting settings. To do this, the grower would be prompted for an optimal temperature (t_{opt}) for cultivating the crop. Using this optimal temperature, a sample scenario (figure 2) could be posed to provide the grower a framework from which to begin. Graphic representations are provided to help the grower visualize the scenario.

Figure 2: Sample Scenario

Many growers can identify several stages of growth in which different misting strategies are employed.

A grower, assuming an optimal temperature of 86°F with consistently bright, sunny, dry days, has identified the following stages in the development of his crop.

Stage 1: (day 0-9) - propagation begins, very fequent mistings of very short duration
Stage 2: (day 10-21) - root systems very early in development, frequent mistings of short duration
Stage 3: (day 22-38) - root systems progressing, mistings of moderate frequency and moderate duration
Stage 4: (day 37-71) - root systems well-developed, infrequent mistings of long duration
Stage 5: (day 72-95) - mature crop, preparation for market infrequent mistings of moderate duration

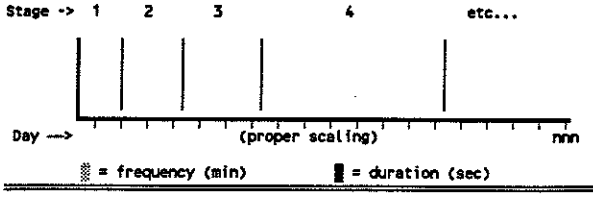


Then, through an interactive process, the grower is asked to construct a similar scenario specific to his crop assuming constant conditions (optimal temperature, low humidity and high light levels) (figure 3). Experts consulted felt that commercial growers most likely could answer the questions necessary to construct such a scenario. The number of stages is grower-defined.

Figure 3: User-Defined Scenario

Assuming a constant temperature of ___°F and consistently bright, sunny, dry days, identify important stages in the development of your crop with recommended misting frequency and duration settings.

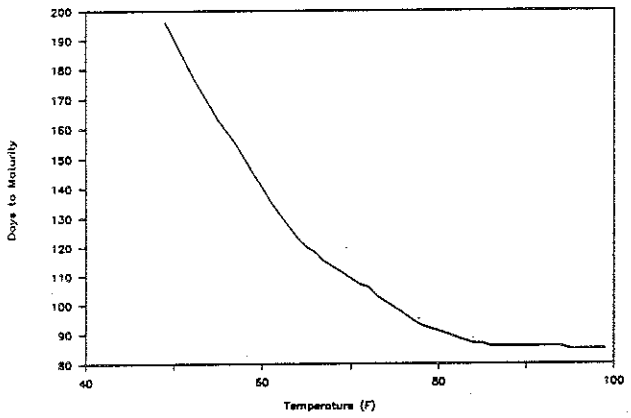
Stage 1: (day ___-___) - frequency: ___ min.
duration: ___ sec.
Stage 2: (day ___-___) - frequency: ___ min.
duration: ___ sec.
Stage 3: (day ___-___) - frequency: ___ min.
duration: ___ sec.
Stage 4: (day ___-___) - frequency: ___ min.
duration: ___ sec.
etc...



The Peperomia expert recognized the importance of temperature on crop development. If temperatures are sub-optimal, it will take longer for the crop to progress from stage 1 to stage 2, and from stage 2 to stage 3, etc. In MISTING, the expert prescribed an adjustment to the actual crop age based on the average cumulative temperture. The apparent crop age is then used to determine the appropriate developmental stage of the crop. Once the appropriate stage is determined, baseline frequency and duration settings from the user-defined scenario (figure 3) can be derived. Other experts agreed that this general approach was appropriate for this tool.

Research horticulturists understand that for each plant variety, there is a growth curve that characterizes the effect of temperature on growth rate. Most of these growth curves are similar in shape. A sample curve appears in figure 4.

Figure 4: Sample Growth Curve



To establish the apparent crop age, the MISTING expert prescribed a table of adjustment factors (table 1). A research horticulturist expert in growing Peperomia very likely has the analytic ability to recognize and the propensity to formally define the effect of temperature on crop development in this manner. The same abilities are not expected of a commercial grower. A primary task then is to extract what knowledge a commercial grower can provide, and then formalize and represent this knowledge in a format usable to an expert system.

If key points on the growth curve could be identified, then the curve could be approximated and values extrapolated that could subsequently be used in computing the apparent crop age.

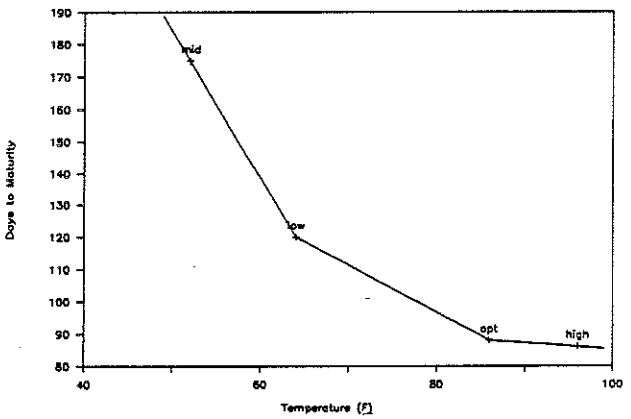
A series of questions were developed that could be answered by commercial growers and would provide some key points on the growth curve that should allow the apparent crop age to be computed.

Values for the variables defined below can be derived (directly or indirectly) from grower input.

- **topt** - the grower-defined temperature for optimal crop cultivation
- **dopt** - the grower-defined estimate for the length of time (in days) to crop maturity at topt
- **tlow** - the grower-defined lowest temperature for which good growth can be achieved
- **dlow** - the grower-defined estimate for the length of time (in days) to crop maturity at tlow
- **tmin** - the temperature below which the crop is damaged
- **tmid** - computed as [(tmin+tlow)/2]
- **dmid** - the grower-defined estimate for the length of time (in days) to crop maturity at tmid
- **thi** - computed as (topt+10)
- **dhi** - the grower-defined estimate for the length of time (in days) to crop maturity at thi

These values are used to plot four key points along the growth curve (figure 5). By extrapolating between the points, the growth curve for the plant variety in question can be estimated. These extrapolations can then yield adjustment factors for computing the apparent crop age.

Figure 5: Estimated Growth Curve



After computing the apparent crop age and establishing baseline frequency and duration settings, MISTING adjusts for current ambient temperature and relative humidity (table 3). Other experts agree that such adjustments must be made.

Growers typically do not have a formal, defined table of such adjustment factors. Relying on the structure from MISTING (table 3), the tool elicits specific knowledge that the grower is capable of providing, and extrapolates a usable table of values.

A similar approach for considering the effects of light-levels is employed.

TESTING

This knowledge acquisition tool is still in the design stage. Once the design is complete, several levels of testing will be performed. First, the commercial growers will be asked to use the tool under the observation of a member of the design team. Subsequent interviews will be conducted to determine if the grower felt comfortable using the system, if he understood the scenarios and questions posed by the system, and if he felt he had the ability to answer the questions. Refinements to the sytem at this point are expected.

The second level of testing will involve the Peperomia expert. The Peperomia expert will be asked to use the tool. The resulting knowledge base will be compared to MISTING's knowledge base to evaluate the level of conformity. Data collected from an actual "run" of MISTING will be applied to the tool's resulting knowledge base for evaluation.

A field test of the system in a commercial setting is expected to follow. This experiment will include both a test crop managed by the tool-derived knowledge base and a control crop managed by the grower using existing controls. These two crops will be grown simultaneously in the same greenhouse.

DISCUSSION

In developing this tool, the knowledge base of the MISTING expert system became a larger factor than was originally anticipated. It was eventually recognized that buried in the MISTING knowledge base was a "deeper" knowledge, knowledge that on the surface appeared simplistic, but after careful scrutiny provided a framework around which the knowledge acquisition tool could be built.

REFERENCES

Hayes-Roth, F.; D. Waterman and D. Lenat (eds.). 1983. *Building Expert Systems*. Addison-Wesley, Reading, MA.

Hoffman, R.R. 1987. "The problem of extracting the knowledge of experts from the perspective of experimental psychology." *AI Magazine*, 6, no. 2 (Summer 1987): 53-67.

Jacobson, B.K.; P.H. Jones, J.W. Jones; and J.A. Paramore. 1989. "Real-time greenhouse monitoring and control with an expert system." *Computers and Electronics in Agriculture*. (in press).

Olson, J.R. and H.H. Rueter. 1987. "Extracting expertise form experts: Methods for knowledge acquisition." *Expert Systems*, 4, no. 3 (August 1987): 152-168.

Preau, D.S. 1987. "Knowledge acquisition in the development of a large expert system." *AI Magazine*, 6, no. 2 (Summer 1987): 43-51.

Schweickert, R.; A.M. Burton; N.K. Taylor; E.N. Corlett; N.R. Shadbolt; and A.P. Hedgecock. 1987. "Comparing knowledge elicitation techniques: a case study." *Artificial Intelligence Review*. 1: 245-253.

ACKNOWLEDGEMENTS

The authors are grateful to Jimmy W. Jones, Professor of Agricultual Engineering, and Dewayne L. Ingram, Professor of Ornamental Horticulture at the University of Florida for their valued assistance in this project.

AN INVESTIGATION OF KNOWLEDGE BASED HELP FACILITIES

R.T.Plant

Department of Computer Information Systems
University of Miami
Coral Gables
Florida, 33124
U.S.A

Keywords: Natural Language Understanding, Expert systems, Shells, Functional programming, Parsing.

Introduction

In this paper we describe the research undertaken to create a knowledge based help facility for UNIX. The paper discusses and compares two approaches: i) An expert system shell approach and ii) a custom implementation of a "natural language" (NL) understanding expert system.

The initial section of the paper discusses how the problem was initially specified and how the specification was utilised in selecting an expert system shell. The creation of a rigorous specification was of vital importance for the project and the formal specification language "Z" was utilised [Sufrin. 84]. This language is mathematical in nature comprised of set theory and predicate calculus. The aim of using a formal specification was to enable the domain to be constrained and unambiguously described. The "Z" notation was used for its clear specification style and its previous successful record in specifying such systems as IBM's CICS system. The second section of the paper details the shell selection process and discusses the factors that influenced the choice. The third section of the paper provides details of the second line of research we followed, that of creating a customized expert system. The system that was ultimately created was based around three research issues: firstly, a methodology for the creation of the knowledge based system, secondly, the creation of a natural language front end to the system and thirdly, the implementation and integration of the system through the use of a pure functional language that utilised lazy evaluation.

Finally, the paper gives a comparative view of i) each of the implementations, ii) the applicability of a pure functional language to expert system construction, iii) the suitability of our parser design to the problem considered iv) the intelligence of each system and v) the usability of the systems.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

Problem Specification

An aim of this research project was to attempt to achieve a high degree of rigor in the system specification and subsequent development. It was therefore decided to utilize a formal language in order to initially specify the problem domain. Several specification languages were considered including VDM [Jones. 80], CSP [Hoare.83], PRO [Henner.84] and "Z" [Spivey. 87]. We found that of these "Z" was the most applicable: Firstly, the notation used is clear, concise and well defined with both a formal syntax and a denotational semantics. However, its primary advantages over the other specification languages are in two areas: It has a highly visual form of syntax that is extremely powerful in being able to abstract and combine specifications from smaller pieces which detail individual aspects of the specification--this is through the 'schema notation'. Secondly, the language itself is highly readable yet has the power if required to undergo the process of data refinement to take the specification towards an implementation. The clear syntax of "Z" is apparent when viewed against the heavy syntax of VDM and the heavy theoretical bias of CSP. An especially useful feature of "Z" is the ability it gives the specifier to combine within the specification both formal text and mathematics, the natural language describing the mathematics - thus the specification can be used both by a "Z" reader and a non "Z" reader, thus performing a dual role.

It should be noted that a specification is of the systems intention and not how to undertake or perform, in algorithmic terms these intentions.

The preparation of the specification serves several useful purposes: firstly, it acts as a definition of the problem in a clear and unambiguous manner, secondly, the specification of the domain is vital for the future maintenance of the knowledge base and thirdly, the specification assists in all aspects of project development - shell selection, verification and validation of system responses.

The Shell Based Approach

The process of producing a rigorous specification of the domain enables us to have a baseline document that could be used in the shell selection process.

The criteria for shell selection was based upon several critical areas, amongst which were the following: hardware, software factors, human factors and managerial factors . Of these factors we used the benchmarking procedures advocated by Press [Press.88] to consider their speed. The system and interface requirements were that it was not too heavy on memory/data storage utilization, and that the package could if necessary be interfaced to a high level programming language. The most important of our selection factors was that the shell had high quality graphical/text handling characteristics. This is due to the highly textual nature of the domain. Finally the system had to be economically viable if it were to ultimately act as a teaching aid.

This selection process yielded two shells of high suitability: 'Q&A' produced by SYMANTEC and 'ESP Advisor' from Expert Systems International, of these the later was chosen as it incorporates a process of 'text animation'. This process attempts to allow the rules, knowledge and advice present in a piece of text to be represented in a precise way through a knowledge representation language, thus creating a knowledge base. The knowledge base being 'animated'-- queried by the consultation shell to provide an interactive consultant for the text involved.

Evaluation of the Shell Approach

After having performed a though investigation of the current shells on the market and having chosen what appeared to be a very suitable shell that was specifically designed for the role of developing expert systems from textual sources, the final outcome of the shell approach was far from the desired ideal we had envisaged.

The shell initially promised to make the transfer of knowledge from text (we were using text books and system manuals as our sources) to a knowledge base a very straight forward process. In fact the shell became totally impossible to structure and it became necessary to write an intermediate specification language in which to model the conceptual structure of the domain. The queries allowed were also too rigid and the development terminated as un-productive.

A Natural Language Approach

The limitations of the shell approach necessitated the creation of a custom built program that would have a natural language (NL) front end and a knowledge base upon which the necessary inferences could be made in order to achieve useful and intelligent responses to a variety of user levels.

It was decided that the use of a functional language would be most appropriate for several reasons. The main advantage of functional languages over procedural languages is that it effectively reduced the amount of work the programmer has to perform. It does this in two ways: i) by handling the allocation of storage, ii) which is more important, the system takes all responsibility for the evaluation order of the functions, removing the problems associated with structuring the program in order to obtain the desired sequence of evaluation.

It was further decided to use a functional language that utilised a "lazy evaluation" strategy [Henderson.86] as this would greatly simplify the parsing process. In order to perform the task of parsing grammars, it is essential, for the system undertaking the task, to use a form of lazy evaluation in order to search for all possible solutions. Logic based languages such as PROLOG [Clocksin.81] use backtracking to perform this role, however this is not as easy to use as lazy evaluation. This implementation in a functional language will also save the programmer from having to explicitly write any backtracking code.

There are several pure functional languages available i.e, ORWELL [Wadler.85], KRC [Turner.81], LispKit [Henderson.83]. It was decided to use KRC.

KRC has the advantage of infinite data structures which are ideal for processing the input and output of parsing programs, which can be regarded as infinite streams of information. This

approach to Input/Output bypasses the problem of explicit 'reads' and having to decide upon the sequencing of all events prior to processing, also by programming in terms of infinite I/O streams this allows programming modules to be combined easily.

Pattern matching is a feature of KRC -- this is extremely useful in the domain of NLP. These techniques allow complex conditions to be expressed very simply, especially when the functions have many arguments, reducing the need to use guards.

The functional approach also allows the use of higher order functions and a recursive evaluation style of programming. These powerful features combined with the use of set expressions [Turner.81] allow for more readable, shorter programs to be developed.

The decision of which programming language to use in creating our system was made in consideration of the parsing techniques we were to use. The first step towards construction of the system was the specification of a grammar. Therefore a series of B.N.F definitions were created which could be parsed in a top down manner, such that all possible parses were generated. Having decided upon the B.N.F form of grammar and to use KRC as an implementation language the next step was to devise a representation for B.N.F in KRC.

There were four components making up the B.N.F i) terminal symbols, e.g., words like "dog", ii) Non terminal symbols i.e., <noun> these being the name of structural units and denoted by being enclosed within angular brackets, iii) The third and fourth components being disjunction and conjunction of the first two components.

e.g.i) <s>::=<noun>|<verb>
here 's' is a <noun> or a <verb>

ii) <s>::=<noun><verb>
here 's' is a <noun> followed by a <verb>

it was decided to represent these four components in terms of lists and by defining some terms which could be regarded by the parser as reserved words e.g., "or" and "seq" these allowing the conjunction and disjunction of symbols.

<noun>|<verb> being represented
in KRC as: ["or",["noun"],["verb"]]

<noun><verb> as ["seq",["noun"],["verb"]]

The complete B.N.F expression: <s>::<noun><verb> would be represented by the KRC equation:

```
bnf = ["s",["seq",["noun"],["verb"]]]
```

Having defined a representation in which to express the four components we were then able to write equations for any set of B.N.F definitions.

The Parser

Having devised the B.N.F and its KRC representation the next step was to develop a parser which would accept a sentence in English and check to see if it has an allowable construct according to the B.N.F definitions.

First step is to take the input and translate it into a list:

e.g., The sentence:

fruit flys like a banana, becomes
["fruit","flys","like","a","banana"]

The next and more important stage was to devise the functions to parse the list of words. The most important function upon which the recursive nature of the parser is based is 'match'.

Match is a function whose type ideally would be:

```
match: BNF X LIST(words) --> Parse_Tree
```

This takes two arguments: A fragment of B.N.F and a list of words which it then attempts to match together, in the ideal case the function having performed it's matching operation would return a single parse tree. This however can only occur when the grammar produces nothing but unambiguous parses.

Due to the problem of ambiguous grammars the function match has the type:

```
match: BNF X LIST(words) -->
          (Parse_Tree X LIST(words))
```

where the output from the match is a set of pairs representing valid parses. The first element of the pair being the parse tree and the second being the list of, as yet, unmatched words that remain after the function match has been applied to match the input list of words with the B.N.F fragments, if the match does not succeed then an empty set results. In the case of an unambiguous parse, match returns a set containing just one pair where the first element in the only valid parse and the second element is the remaining input. When the whole sentence is matched unambiguously again a set containing only one pair is returned this enabling the whole parse tree and the remaining input which should be empty as all of the sentence has been parsed.

Thus match is the basis of our parser on which further functions are added. The next stage was to make the system respond to the users queries with meaningful answers - this being done through the use of a knowledge base.

Knowledge Base Component

Several representations for the knowledge base were considered and it was decided that a semantic association list would be a suitable solution.

The aim of the association list was to condense the question such that the noise was removed within the query whilst amplifying the meaning.

In order to do this the B.N.F had to be extended with the addition of a third reserved word. The word "query" being used to distinguish a section of the B.N.F as significant. It does this by associating sections with tags that contain information on them. Thus the parser, in order to successfully process this new reserved word needed to be modified slightly. The next step was to further develop its interactive qualities of the system. This was done by modularising the knowledge base and utilizing scripts [Schank.77].

Conclusions

It is our conclusion that i) the area of natural language processing is still one where the shell approach is open for large scale improvement ii) The functional programming style of programming combined with the techniques associated with machine learning offer a way forward for significant potential increases in performance for natural language processing, and iii) that the utilization of formal specifications should be encouraged and developed in this area of artificial intelligence.

References:

Clocksin,W.F., and Mellish, C.S. 1981
"Programming in PROLOG".
Springer Verlag.

ESI. 1984
ESP/ADVISOR user guide and reference manual. Expert Systems International. Oxford. England.

Henderson,P. 1983
"The LISPKIT manual Vols I & II"
Oxford University Computer Laboratory
PRG Monograph 32.

Henner,E.C.R. 1984
"The Logic of Programming"
Englewood Cliffs, Prentice Hall.

Jones,C.B. 1980
"Software Development - A Rigourous Approach" Prentice Hall.

Press,L. 1988
"Eight Product Wrap Up: PC Shells"
AI Expert, September 1988.

Schank,R.C., Ableson,R.P. 1977
"Scripts, Plans, Goals and Understanding" Halstead Press.

Spivey,J.M. 1988
"The Z Notation"
Prentice Hall.

Sufrin, B.A. 1984
"Notes for a Z handbook, Part 1 - Mathematical language".
PRG, Oxford University.

Turner, D.A. 1981
"Recursion equations as a programming language". In, 'Functional Programming and its Applications" Eds, Darlington,J., Henderson,P., and Turner,D.A. Cambridge University Press.

Wadler, P. 1985
"An introduction to ORWELL"
Internal Publication, PRG, Oxford University.

Automated Inference of Flow Diagrams from Natural Language Functional Specifications

by

FRANK HADLOCK, *Tennessee Technological University*
MARK FISHMAN, *Eckerd College*
FRANK ANGER and RITA RODRIGUEZ, *Florida Institute of Technology*

§ PROBLEM DESCRIPTION §

Section 1

Software system development is a complex process beginning with the initial phase in which the behavior of the system is specified. The *functional specifications* or **SRS** (*software requirements specifications*) ideally describe *what* is to be done (as opposed to *how* to do it) and should serve as a basis for *design, testing* and *maintenance*. Typically there is a significant natural language component of functional specifications. Because of the inherent ambiguity of natural language there is the **potential for a differing interpretation** being incorporated into the *design* from that intended in the *functional specification*.

These problems motivate the definition of a model for system behavior which can be inferred from functional specifications and which can be used to check completeness of the specifications and to infer data flow between modules. The State-Event Calculus model (SeCalc model) defined in this paper is **based on** an examination of some of the existing models used in various life cycle phases as well as general models for *knowledge representation* such as *frames* and *semantic networks*. Its development was predicated on the following **basic assumptions and requirements :**

- *Functional Specifications will contain a significant natural language component.*
- *The model should serve as a basis for further stages of development and should be inferred from the functional specifications.*
- *The model should display an explicit correspondence to the conceptual components of natural language functional specifications.*

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

The specific problems addressed in this paper are that of inference of the SeCalc model from natural language functional specifications and of inferring data flow between modules from the SeCalc model. The purpose of this paper is to outline a possible approach to these problems, enumerating preliminary results and remaining problems.

§ BACKGROUND AND RELATED WORK §

Section 2

In this section, an overview is given of those concepts and of previous work relevant to the problems of interest.

Functional specifications describe what the software system is supposed to do. Some methods for functional specification are discussed in [Davis]. These include *Program Description Language* (PDL) and *Requirements Statement Language* (RSL). The State-Event Calculus defined here is closest in concept to the Entity-Relation-Attribute model.

The issue of automatic or interactive inference of the model from natural language functional specifications is the primary interest of this paper, while attributed context free grammars and case grammars are discussed briefly as a basis for *syntactic - semantic directed translation* from natural language to the model. Our approach employs frame structures as the basic building block of the State-Event Calculus. Thus, the concept of frames is introduced in this section.

Although *Natural language functional specifications* are not recommended, they are still used, because many existing systems which need to be maintained were specified by natural language. Furthermore, approaches such as PDL and RSL mix keywords and natural language. Consequently our objective is the development of mechanisms for deriving the system behavior model from natural language functional specifications. Appendix B gives a Track Management Function to illustrate SeCalc. The example was constructed after considering natural language functional specifications for an actual system, and is at least as difficult in terms of automatic interpretation as any of the language used in describing the actual system.

Entity-relation-attribute models are usually used in conjunction with representing information in databases.

However, the concepts of object sets (entities), value sets (attributes), and n-ary relations between them, can be applied to modeling system behavior. In fact, as pointed out in [Fullerton, et al.], SREM and RSL can be viewed in these terms. RSL is a relational language composed of **elements** (entities) classified as *data, data structures*, and *processing models*; **attributes** used to represent properties of elements; **relationships** used to represent relations between elements (e.g. INPUT is a relationship between a particular data entity and a particular processing entity); and **structures** which include R-nets.

Phrase structure grammars were introduced by Chomsky as an approach to defining natural languages. Context free grammars are useful in defining programming languages, since efficient algorithms do exist for analyzing character strings with respect to these grammars, thus forming the basis for syntax directed lexical analysis and compilation. Nonetheless, as far as modeling some aspects of natural language such as agreement between subject and verb (as to singular or plural), etc., or capturing semantic equivalence between active and passive constructions, they remain inadequate.

Case grammars, on the other hand, were introduced by Fillmore. These grammars are able to capture the semantic equivalence between active and passive cases by concentrating on the semantics aspects of a declarative sentence. The essential idea is that the meaning of a declarative sentence is composed of *a verb* along with some *noun phrases*, each noun phrase *related to the verb* by a **case** relationship. Here, *noun phrase* is an extended notion, including what are usually considered *prepositional phrases*. Fillmore proposed the following case relations: *agentive, instrumental, dative, factitive, locative,* and *objective.*

As can be seen, case names are somewhat descriptive of the relationship between noun phrase and verb. For example, *agent* is the relationship between the verb and the noun phrase which *performs the action*, *objective* is the relationship between the verb and the noun phrase which is *the object of the action.* We later develop our own case relation list appropriate to the language which would be used to describe system behavior.

Frames are similar to the *record construct* which many programming languages provide as a mechanism for structuring data items of differing base types. An example of their use in knowledge representation occurs in conjunction with Case Grammars where a Case Frame is associated with a verb and has a *slot* to be filled by a noun phrase for each role. The role slots present in the case frame will depend on the verb. To illustrate, consider the phrase from the Track Management Function description: *"Upon receipt of a new track from the Track Processor . . ."* The verb *receive* requires the roles of *agent, object,* and *From-Loc*(or source). The roles of *agent* and *To-Loc* (or destination) coincide and are played by the *Track Management Function*, the role of *object* by *new track*, and the role of *From-Loc* by *Track Processor.* Thus the meaning of the phrase can be represented by the verb and case frame:

Case frame for Verb: receive
Agent: *Track Management Function*
Object: *new track*
From-Loc :Track Processor

Parsing is the process of determining the structure of a sentence according to the rules of a grammar for the language. *Top-down parsing* is any approach in which the source string is constructed, beginning with the syntactic type ⟨*Sentence*⟩. No top down parsing algorithms lend themselves to processing *ungrammatical constructions* which are acceptable in natural language functional specifications. The reason is that, since top-down parsing begins with ⟨*sentence*⟩ and applies grammar rules to generate the sentence, there will be an immediate failure if, in the case of a fragment, the fragment does not begin as a valid sentence. Otherwise a failure occurs when there is no valid continuation. Thus top-down algorithms do not lend themselves to graceful degradation in the sense of *analyzing all valid constituents of the fragment or ungrammatical constructions.*

Bottom-up parsing is any approach which begins with the words of the sentence and constructs the derivation in reverse. Linear time algorithms exist for restricted classes of grammars. The Cocke-Kasami-Younger tabular algorithm, which is applicable to any context free language, runs in cubic time; however, grammars must first be converted to Chomsky normal form, an inconvenience. Younger's tabular parsing algorithm [Younger] has been extended, incorporating some of the notions of SLR(1) parsing [Hadlock], to be used in conjunction with arbitrary grammars.

§ State-Event Calculus §

Section 3

The State-Event Calculus may be regarded as an Entity-Relation-Attribute approach to representing system and domain knowledge. In this section we detail an approach to representing system and domain knowledge which will be the basis for representing system behavior. At the foundation are the *entities* in the *domain* and in the *system* which *initiate action* and which are *acted upon. States* have *duration* and involve *relations between* or *attributes of* these *entities. Events* are *instantaneous* and correspond to some *system activity.* A *type of system activity* is the *central component* of an event and events are classified according to standard *case relations* between the central system activity and the entities involved.

It will be seen that the components of States and Events correspond naturally to phrases and clauses within natural language functional specifications, or more formal specifications such as PDL and RSL. Consequently SeCalc does display an explicit correspondence to the conceptual components of a natural language functional specification.

An ISA hierarchy for software system entities is the basis of the system and domain knowledge employed by the SeCalc translator. The universe consists of all enti-

ties or objects belonging to the software system and to the physical domain in which the system is embedded. This universe is partitioned along conceptual lines into subtypes corresponding to the various roles that entities in a subtype play with respect to an event or activity. This partitioning is a key factor in our strategy to infer the model from natural language functional specifications, using a case grammar model for language. Those entities which play the role of the *object* of the action will be termed *Objective Entities*, of which data values, messages, tutorials are all examples. Possible subtypes are *transient* and *permanent* according to whether they are typically stored in secondary storage. Those entities which play the role of the *agent* of the action will be termed *Agentive Entities*. As a consequence, agentive entities will tend to be software processes. As examples, processes typically perform the actions of *send/receive, verify that, determine if, ..., etc.* and so are the *agents* of these actions. *Locative entities* are those which can have a *locative* role relative to an action constituent of system behavior. For example, if a system process *displays* a tutorial on a specific CRT, the CRT plays a *locative* role in the action of *display*.

A *State* corresponds to a *relation* or *attribute* associated with an entity which has *duration*. From the Track Management example, the clause ... *the buffer is full* ... would cause creation of the state:

State S_1

Object	:	*buffer*
Attribute	:	*remaining capacity*
Value	:	*zero*

Events have some system activity as central component and are considered to be instantaneous. The events or actions which characterize software system behavior can be grouped according to the following classes: *processing, data storage or retrieval, communication* and *control*. The classification method is motivated by the *types of entities* involved.

We distinguish between the following *types of involvement*. *Mandatory explicit*, *mandatory implicit*, or *optional*, which mean, respectively, that the entity must be explicitly referenced by the natural language description, that the entity is necessary but can be inferred, or that it is optional. For example, an *agentive entity* is *mandatory implicit* because, while an agent is necessary for an action, the default candidate is the process being defined. On the other hand, if not in an ethernet context, a *To-Loc* is *mandatory explicit* for an action of sending a message since it is vital to know which process it is to be sent to, and there is no reasonable way for inferring that information. In classifying events, we assume that, for all classes, an agent is mandatory implicit while an object is mandatory explicit. The *event classes* are as follows.

Processing events may require more than one *object* corresponding to the operands; additionally, a *result* will usually be required. Processing event subtypes correspond to the number of operands, as well as whether the result is implicit or explicit. For example, a natural language reference to a *comparison* (such as occurs in the Track Management Function specification) may *not* make explicit reference to a result while one to a *calculation* does.

As an example, the clause ... *the Closest-Point-of-Approach of the new track is compared with that of each track in the buffer* ... will cause the creation of a processing event, where no result has been inferred:

Event_Proc

Agent	:	Track Management Function
Action	:	Compare
$Object_1$	:	CPA(new track)
$Object_2$	:	CPA(track in buffer)
Result	:	?

Data storage and retrieval events also involve a software process as *agent*. A storage action requires a data value or file or program file as *object*, as well as a storage device as the *to-location* of the action.

A retrieval action is like a storage action, requiring an *agent*, *object*, and a *from-location* which is a *locative entity*.

Another example is the superordinate clause ... *the Track Management Function shall store the track in a buffer* ... will cause, (by using the reference of *the track* to *new track* in the subordinate clause), the creation of storage event:

Event_Str/Ret

Agent	:	Track Management Function
Action	:	Store
Object	:	new track
To-Loc	:	buffer

Communication events involve a software process as *agent*. They correspond to *send* and *receive* and differ from *store* and *retrieve* in that they require *agentive entities* in the *location slots* rather than *locative entities*.

To illustrate: the subordinate verbless clause ... *Upon receipt of a new track from Track Processor* ... will cause the creation of a communication event:

Event_Comm

Agent	:	Track Management Function
Action	:	Receive
Object	:	new track
From-Loc	:	Track Processor

Control events involve two entities, one of which plays the role of *agent*, initiating an action which alters the state of the other *object* entity. The object entity may be locative or agentive. Illustrations of these two cases occur when a process activates another process and when a process clears a buffer. Control events also involve a *new state* which may be inferred from the verb.

Such an event is given by the coordinate clause ... *activating the OverFlow Track Manager* ... with ellipsis of the subject (agent). It would cause the creation of the control event below:

Event_Ctrl

Agent	:	Track Management Function
Action	:	Activate
Object	:	OverFlow Track Manager
New State	:	S_2

Ellipsis and *reference* are some of the issues relating to the inference of the model from natural language. If a natural language reference to an event does not contain an explicit reference to an agent due to *ellipsis* (i.e. omission of subject), an agent must be inferred. Consequently, inference rules need to be developed. In the specification of the Track Management Function we might have said: ... *the new track shall be entered in the track buffer* .... It will be seen that the *Track Management Function as the location of the new track* can be inferred from this description and is the correct choice as the agent of the storage action.

Besides ellipsis, another major problem is *pronoun* and *noun reference*. For example, *the track* references *a new track* in the Track Management Function definition. Approaches to determining the correct reference include *semantic rules* based on *flow axioms*, analysis of *preceding text*, and establishing *location of similar entities*.

§ Model Inference §

Section 4

By *automatic inference* is meant that a program based on syntactic and semantic rules attempts to convert the natural language component of functional specifications to a SeCalc model. Particularly when adapting to a new domain (e.g., C^2 military applications or patient monitoring), the translator will encounter problems. The policy will be to translate what it can and flag the rest. The syntax and knowledge base of the translator will be extended by translating the rest with operator intervention. This mode will be referred to as semi-automatic, a natural part of the process of adapting to a new domain. A number of issues need to be addressed in designing and developing the translator; some of these are listed below:

Semantics of Collocations and Noun Phrases: Collocations such as *track buffer* and *Overflow Track Management Function* are commonly used for *objective, locative* and *agentive entities*. Rules need to be formulated for determining the *semantic roles* which can be played by an *entity* referred to by a collocation. Of course, these rules would make use of roles associated with the words in the collocation or phrase, this information being stored in the lexicon.

Lexicon format and generation: Each domain is apt to have words with special meaning. A *standard lexicon* for system software development would have standard terms such as *function, process, data, buffer,* etc. This would usually have to be augmented or modified for a specific domain. The lexicon should ultimately contain collocations or special phrases.

As to individual words, the lexicon needs to store, for each word sense, semantic role information. For **nouns**, *agentive* means the associated entity can perform action (e.g. *process* or *function*), *objective* means the associated entity can be acted on (e.g. *data* or *tutorial*), etc. For **adjectives**, the role information would refer to the roles which could be played by nouns modified by the adjective. For **verbs**, the case information would be a **verb class** with an associated *case frame*. For **prepositions**, various prepositions typically introduce a prepositional phrase which plays a particular semantic role.

Case relations: Standard case relations would include *Agent, Object, Instrument, To-Loc* and *From-Loc* as well as *At-Loc*, $Object_i$, $i = 1 \ldots k$, and *Result*. Verb classes need to be identified with a *case frame* for each class. Within a case frame would be a list of case relations appropriate to this class. For each relation one of the alternatives *Mandatory implicit, Mandatory explicit* or *optional*, as described earlier, would be stored. If a case relation is omitted, it is *inappropriate* (e.g. *result* relation for verb *enter*).

System and domain knowledge: Because translation will depend heavily on semantics for word sense, referential, and case relation disambiguation, both system and domain knowledge need to be stored in a dynamic knowledge base on which expert rules will be used to guide the translation. These expert rules themselves would be stored in the knowledge base; hence, guidelines for storing and making use of such information need to be formulated.

Lexical analysis: Lexical analysis refers to the process of looking up text items in the lexicon and returning references to the lexicon. Since word sense ambiguity poses a problem in natural language processing, context information is needed for word sense disambiguation. If case information is also needed, there would be reason to perform only rudimentary operations in lexical analysis, and to do it concurrently with parsing and translation, one word at a time. On the other hand, there is a need to gain as much information as possible about the whole phrase, in order to determine the verb class and corresponding case frame.

A major task of this investigation is the design and implementation of a SeCalc model generator, consisting of a lexical analyzer, parser, and semantic processor, cooperating to generate a formal model through the use of a semantic and syntactic rule base.

Parsing and translation : Parsing and translation are accomplished concurrently, with semantic information being used to aid in the parsing. The parsing and translation approach taken is somewhat patterned after *Parsifal*, the approach developed by Marcus. The criteria for developing a parsing and translation approach for this application are that: *ambiguity is provided for, ungrammatical constructions can be processed,* and when failure occurs, the parser/translator *translates what it can, presenting alternative interpretations for ambiguous clauses,* and *providing diagnostic information for failure situations*. The general approach employed in the SeCalc translator is as follows:

For each sentence, perform the following:
For each clause, perform the following :
- *perform lexical analysis,*
- *identify verb phrase*
  else invoke verbless clause rules
- *retrieve case frame associated with verb class*
- *parse noun/prepositional phrases,*
  using prepositional case information to fill slots

- *At the sentence level (for compound sentences) :*
  - *Use interclausal relations to infer fillers*
    for missing slots due to ellipsis
  - *Infer pronoun and nondefinite noun references*
  - *Infer temporal and causal relations*
    from subordinators, coordinators.

General Tabular Parsing [Hadlock] : To identify verb, noun and prepositional phrases, the general tabular parsing (GTP) algorithm is employed which has the advantages that it handles ambiguity, degrades gracefully in ungrammatical situations, and lends itself to the concurrent use of semantic rules to resolve ambiguity, and to schedule its efforts (e.g. find verb phrase first).

The GTP algorithm employs a triangular table $T_{i,j}$ as does Younger's algorithm. While Younger's algorithm enters nonterminals in the table ($A \in T_{ij}$ means $A \Rightarrow^* x_j \dots x_{j+i-1}$), the GTP algorithm employs *primitive* and *phrase items.*

A *primitive item* is of the form $X\bullet$ where $X \in \Sigma \cup N$ is either a terminal or nonterminal character. A *phrase item* is of the form $\alpha \bullet \beta$ where $A \rightarrow \alpha\beta \in P$. For a primitive item, the meaning of $X\bullet \in T_{i,j}$ is that $X \Rightarrow^* x_j x_{j+1} \dots x_{j+i-1}$ while for a phrase item, the meaning of $\alpha \bullet \beta \in T_{i,j}$ is that $\alpha \Rightarrow^* x_j x_{j+1} \dots x_{j+i-1}$.

The description of the General Tabular Parsing algorithm, operating independently of any semantic control, appears in [Hadlock]. It is employed here in conjunction with the concept oriented grammar given in Appendix A, which generates loosely constrained clauses and phrases, occurring within even more loosely constrained simple and compound sentences. The associated semantic rules are used to map noun and prepositional phrases onto slots in the verb case frame.

§ Inference of Data Flow §

Section 5

Once the SeCalc model is obtained from the natural language functional specifications, data flow between modules is easily derived from the *communication events* occurring in module descriptions. A description of the data flowing from one module to another is obtained from the *object* slot of the *communication event case frame* while the originating and terminating modules correspond to *agent* and *To-Loc* slots respectively if the *action* slot is *send* or to the *From-Loc* and *agent* slots if the *action* slot is *received.*

§ Conclusions §

We have outlined an approach to the automatic inference of data flow between modules from natural language specifications. The approach introduces a case frame formal model of software system behavior, the SeCalc model. The grammar presented in Appendix A, along with the mixed syntactic-semantic analysis, employing the general tabular parsing algorithm and case frame analysis, represent preliminary steps in the automatic inference of the model. In practice, we envision an interactive system which, for a given application area, begins with a seed grammar and lexicon. As functional specifications are being processed in the learning phase, the system will attempt to process sentences which are ungrammatical (with respect to the seed grammar), or will be unable to fill in the case frame slots. Through user interaction, additional syntactic and semantic rules will be inferred. Realistically, we do not expect the system to be ever fully automatic.

References

ALLEN, J.,**Natural Language Understanding,** *Benjamin Cummings, 1987*

CULLINGHAM, R.,**Natural Language Processing,** *Rowman & Littlefield, 1986*

DAVIS, A., **A Comparison of Techniques for the Specification of External System Behavior,** *Comm. ACM, Sept. 88, pp 1098-1115*

FULLERTON, J., ET AL., **Specifying Distributed System Requirements Using SREM** , Davis, C. G., et al., (eds.), *Entity-Relationship Approach to Software Engineering* , North Holland, 1983

HADLOCK, F., **General Tabular Parsing,** *submitted*, 1989

HADLOCK, F., SHOKOOH, A., ANGER, F., RODRIGUEZ, R., FISHMAN, M., **The State-Event Calculus as a Basis for Completeness Checking,** *Proceedings of 27th Conference SE Region ACM*, 1989

HARRIS, M.,**Introduction to Natural Language Processing,** *Reston, 1985*

MELLISH, C.,**Computer Interpretation of Natural Language Descriptions,** *Halsted, 1985*

WINSTON, P.,**Artificial Intelligence,** *2nd Ed., Addison-Wesley, 1984*

WINSTON, T., TAYLOR, M., LEEDS, R.,**Natural Language Query Processing,** *AI Expert, Feb. 1989*

YOURDAN, E.,**Modern Structured Analysis,** *Prentice-Hall, 1989*

§ Appendix A §
A Grammar for Natural Language Functional Specification

⟨*Sentence*⟩ → ⟨*Antecedent_Phrase*⟩$_1$ ⟨*Consequent_Phrase*⟩$_2$
Corresponding Semantic Event Form:
(IF 1 THEN 2)
⟨*Antecedent_Phrase*⟩ → ⟨*Antecedent_Prep*⟩$_1$ ⟨*Action_Nominalization*⟩$_2$
[⟨*Object_Prep*⟩$_3$ ⟨*Noun_Phrase*⟩$_4$]
[⟨*Agent_Prep*⟩$_5$ ⟨*Noun_Phrase*⟩$_6$]
[⟨*Source_Prep*⟩$_7$ ⟨*Noun_Phrase*⟩$_8$]
[⟨*Destination_Prep*⟩$_9$ ⟨*Noun_Phrase*⟩$_{10}$]
(Corresponding Semantic Event Form:
(Verb-Form(2) Object 4 Agent 6 Source 8 Destination 10)
⟨*Antecedent_Phrase*⟩ → ⟨*Antecedent_Prep*⟩$_1$ ⟨*Action_Nominalization*⟩$_2$
[⟨*Agent_Prep*⟩$_3$ ⟨*Noun_Phrase*⟩$_4$]
[⟨*Object_Prep*⟩$_5$ ⟨*Noun_Phrase*⟩$_6$]
(Corresponding Semantic Event Form:
(Verb_Form(2) Object 6 Agent 4)
⟨*Antecedent_Phrase*⟩ → ⟨*Antecedent_Int*⟩$_1$ ⟨*Sentence*⟩$_2$
⟨*Antecedent_Int*⟩ → IF, WHEN
⟨*Antecedent_Prep*⟩ → ON, UPON, AT
⟨*Sentence*⟩ → ⟨*Noun_Phrase*⟩$_1$ ⟨*BE*⟩$_2$ ⟨*Adjective_Phrase*⟩$_3$
(Corresponding Semantic Event Form:
(HAVE_PROPERTY THEME 1 ATTR 3)
⟨*Adjective_Phrase*⟩ → ⟨*Adjective*⟩
⟨*Noun_Phrase*⟩ → [⟨*Det*⟩$_1$] [⟨*ADJ*⟩$_2$] N_3 [⟨*Prepositional_Phrase*⟩$_4$]*
[⟨*Complementizer*⟩$_5$ ⟨*Sentence*⟩$_6$]
(Corresponding Semantic Event Form:
(3 Det Determinacy(1) Modifier_Case_Name_Selector(2) 2 _4_ SUCH_THAT 6)
⟨*Prepositional_Phrase*⟩ → ⟨*Preposition*⟩$_1$ ⟨*Noun_Phrase*⟩$_2$
(Corresponding Semantic Event Form:
(Case_Name_Selector(1,HEAD_Noun(2)) 2)
⟨*Consequent_Phrase*⟩ → ⟨*Consequent_Int*⟩ ⟨*Sentence*⟩
⟨*Consequent_Int*⟩ → THEN, ","
⟨*Sentence*⟩ → ⟨*Noun_Phrase*⟩$_1$ ⟨*Aux*⟩$_2$ ⟨*V*⟩$_3$ [⟨*Noun_Phrase*⟩$_4$] [⟨*Prepositional_Phrase*⟩$_5$]*
[⟨*CONDITION_Int*⟩$_6$ ⟨*Sentence*⟩$_7$]
(Corresponding Semantic Event Form:
(IF (NEGATION?(6) 7) THEN (3 Agent 1 MODALITY 2 Object 4 5))
(NOTE: "NEGATION?" IS A Function WHICH RETURNS "NOT"
IF ⟨*Condition_Int*⟩ = UNLESS; "" IF ⟨*Condition_Int*⟩="IF")

⟨*Condition_Int*⟩ → IF, UNLESS
⟨*Sentence*⟩ → ⟨*Conditional_Situation_Referent_Phrase*⟩ ⟨*Sentence*⟩
(Corresponding Semantic Event Form:
(IF Selection_Function(1,Previous_Sentence())
THEN 2))
⟨*Conditional_Situation_Referent_Phrase*⟩ → IN ⟨*Selector*⟩ CASE
— WHEN ⟨*Selector_Pronoun*⟩ OCCURS
⟨*Selector*⟩ → THIS, THAT, SUCH A , THE ⟨*Ordinal_Adjective*⟩
⟨*Ordinal_Adjective*⟩ → FIRST, SECOND, THIRD...
⟨*Sentence*⟩ → ⟨*Passive_Sentence*⟩
⟨*Passive_Sentence*⟩ → ⟨*Noun_Phrase*⟩ ⟨*BE*⟩ ⟨*V*⟩_ ⟨_*ED*⟩ _[⟨*Clitic*⟩] ⟨*Noun_Phrase*⟩ _
[⟨*Conditional_Situation_Referent_Phrase*⟩]
⟨*Sentence*⟩ → ⟨*Test_Sentence*⟩
⟨*Sentence*⟩ → ⟨*Passive_Test_Sentence*⟩
⟨*Passive_Test_Sentence*⟩ → ⟨*Noun_Phrase*⟩$_1$ ⟨*BE*⟩$_2$ ⟨*Test_V*⟩$_3$ ⟨_*ED*⟩$_4$
_⟨*Clitic*⟩$_5$ ⟨*Noun_Phrase*⟩$_6$ ⟨*CONJ*⟩$_7$ ⟨*Sentence*⟩$_8$
(Corresponding Semantic Event Form:
(IF (3 1 2) THEN 7)))
⟨*Sentence*⟩ → ⟨*Sentence*⟩ ⟨*Time_Adverbial*⟩
(⟨*Gerundive*⟩ — ⟨*Action_Nominalization*⟩)
⟨*Gerundive*⟩ → ⟨*Gerundive*⟩ ⟨*Conjunction*⟩ ⟨*Gerundive*⟩
⟨*Conjunction*⟩ → AND, BUT
⟨*Time_Adverbial*⟩ → AFTER, BEFORE, WHEN
⟨*Gerundive*⟩ → ⟨*V*⟩ _ING [⟨*Noun_Phrase*⟩] [⟨*Prepositional_Phrase*⟩]*

§ Appendix B. §
Example of SeCalc Model Inference

Track Management Function: Upon receipt of a new track from the Track Processor, the Track Management Function shall store the track in a buffer unless the buffer is full. In this case, the Closest_Point_of_Approach of the new track is compared with that of each track in the buffer, and the track with maximum CPA is discarded after sending a Track Discarded notification to CRT_4 and activating the OverFlow Track Manager. When a Track Display operator request is received, all tracks in the buffer are displayed.

The State_Event Calculus model for the Track Management Function (TMF) is inferred, using the parsing and translation approach outlined in section 4. The natural language specifications are **decomposed into phrases** and the associated components *states, events and causal relations* of the SeCalc model given with each phrase:

Phrase	**Classification**	**Interpretation**
Track Management Function:	Process ID	Default Agent ← $TrkMgtFnc$ Default To_Loc ← $TrkMgtFnc$
Upon	$\langle Sub\_Conj \rangle_{Antecedent}$	Begin definition event E_1 with $E_1 \rightarrow E_2$
receipt	$\langle Nom\_Vrb \rangle_{Comm}$	$Event\_Type \leftarrow Comm$ $Action \leftarrow Receive$
of a new track	$\langle Noun\_Phr \rangle_{Obj}$	$Object \leftarrow new\ track$
from the Track Processor,	$\langle Noun\_Phr \rangle_{FrmLoc}$	$From\_Loc \leftarrow Track\ Processor$ E_1 definition complete
the Track Management Function	$\langle Noun\_Phr \rangle_{Agent}$	Begin E_2 definition $Agent \leftarrow TrkMgtFnc$
shall store	$\langle Vrb\_Phr \rangle_{Str/Ret}$	$Event\_Type \leftarrow Str/Ret$ of $Action \leftarrow Store$
the track	$\langle Noun\_Phr \rangle_{Obj}$	$Object \leftarrow track$
in a buffer	$\langle Noun\_Phr \rangle_{ToLoc}$	$ToLoc \leftarrow buffer$
unless	$\langle Conjunction \rangle_{PostCond}$	E_2 definition complete Expect definition of state S_1 with new antecedent of $E_2 : E_1 \wedge \neg S_1$
the buffer	$\langle Noun\_Phr \rangle_{Loc}$	$Object \leftarrow buffer$
is	$\langle Relr \rangle_{Equals}$	$StateRel \leftarrow Equals$
full.	$\langle PredAdj \rangle_{full}$	$StateAtt \leftarrow full$ S_1 definition complete
In this case,	$\langle Conj. \rangle_{Post\_Cond}$	Begin definition of event E_3 with antecedent $S_1 \wedge E_1$
the Closest_Point_of_Approach of the new track	$\langle Noun\_Phr \rangle_{Object}$	$Object_1 \leftarrow CPA(new\_track)$
is compared with	$\langle be\ Vrb\ \_ed\ Part \rangle_{Proc}$	$Event\_Type \leftarrow Proc$ $Action \leftarrow Compare$
that of each track in the buffer,	$\langle Noun\_Phr \rangle_{Object}$	$Object_2 \leftarrow CPA(each\_track\_in\_buffer)$
and	$\langle Conj \rangle$	Definition of E_3 complete
the track with maximum CPA	$\langle Noun\_Phr \rangle_{Object}$	Begin definition E_4 $Object \leftarrow maximum\ CPA\ track$
is discarded	$\langle be\ Vrb\ \_ed \rangle$	$Event\_Type \leftarrow Str/Ret$ $Action \leftarrow Discard$
after	$\langle TempRel \rangle$	E_4 definition complete Begin definition of E_5 Insert E_5 between E_3 and E_4
sending	$\langle Vrb\_Phr \rangle_{comm}$	$Event\_Type \leftarrow Comm$ $Agent \leftarrow TrkMgtFnc$ $Action \leftarrow Send$
a Track Discarded notification	$\langle Noun\_Phr \rangle_{Object}$	$Object \leftarrow TrkDis\_notification$
to CRT_4	$\langle Noun\_Phr \rangle_{To\_Loc}$	$To\_Loc \leftarrow CRT\_4$
and	$\langle Conj \rangle_{Coord}$	

activating	$\langle Vrb\_Phr \rangle_{Cntrl}$	
the OverFlow		
Track Manager.	$\langle Noun\_Phr \rangle_{Agent}$	
When	$\langle Subord \rangle_{Antecedent}$	
a Track Display operator		
request	$\langle Noun\_Phr \rangle_{Object}$	$Object \leftarrow TrkDispReq$
is received,	$\langle Vrb\_Phr \rangle_{Comm}$	$Event\_Type \leftarrow Comm$
		$Action \leftarrow Receive$
		E_7 definition complete
		Begin E_8 definition
all tracks in the buffer	$\langle Noun\_Phr \rangle_{Object}$	$Object \leftarrow all\_tracks$
are displayed	$\langle Vrb\_Phr \rangle_{Comm}$	$Event\_Type \leftarrow Comm$
		$Action \leftarrow Display$

As the analysis is being performed, the SeCalc model of the process is being constructed as shown below:

↓

$Comm_1$		Str/Ret_2	
Agent :	TMF	Agent :	TMF
Action :	Receive	Action :	Store
From_Loc :	TP	To_Loc :	buffer
$Proc_3$		Str/Ret_4	
Agent :	TMF	Agent :	TMF
Action :	Compare	Action :	Discard
$Object_1$:	CPA(new track)	Object :	max CPA track
$Object_2$:	CPA(track in buffer)		
Result :	?		
$Comm_5$		$Ctrl_6$	
Agent :	TMF	Agent :	TMF
Action :	Send	Action :	Activate
Object :	TrkDscrd Notification	$Object_1$:	OvFlw TrkMgr
$Comm_7$		$Ctrl_8$	
Agent :	TMF	Agent :	TMF
Action :	Receive	Action :	Display
Object :	TrkDispReq	$Object_1$:	All tracks

The temporal/causal event relations are given by:

$\neg S_1 \wedge Comm_1 \rightarrow Str/Ret_2$;
$S_1 \wedge Comm_1 \rightarrow Proc_3 \rightarrow Comm_5 \rightarrow Ctrl_6 \rightarrow Str/Ret_4$;
$Comm_7 \rightarrow Comm_8$

A NEW APPROACH TO REPRESENTING IMPRECISE LINGUISTIC INFERENCES*

Daniel G. Schwartz
Department of Computer Science and the
SUS Center for Artificial Intelligence
Florida State University
Tallahassee, Florida 32306-4019
Arpanet: schwartz@nu.cs.fsu.edu

Abstract

A key problem in the field of expert systems is how to simulate the manner in which humans reason with imprecise linguistic information. For example, one would like to model inferences such as "*Most* professional basketball players are *very tall*; Bill is a professional basketball player; therefore it is *very likely* that Bill is *at least tall.*

Reference [1] gives one approach to this problem, and [2] gives a related, although substantially different, one. The present paper is a report on current work towards synthesizing these two ideas into a unified system. At present, this deals only with inferences involving linguistic attributes, e.g., *very tall* and *at least tall.* It reserves until later the task of dealing with linguistic quantifiers (e.g., *most*) and linguistic likelihood.

This work also goes beyond [1 and 2] by extending from the case of simple decision rules, characterized as single-variable Horn clauses, to general Horn clauses. For example, the revised system can handle the three-variable inference "If X is *very much preferred* to Y, and Y is *preferred* to Z, then X is *very much preferred* to Z.

Introduction

The reasoning system to be described is based on an adaptation of L.A. Zadeh's concept of a linguistic variable [3]. Informally, a linguistic variable is a variable which takes as its values specific terms in a natural or artificial language. Formally, a *linguistic variable* Λ is here represented as a triple (T, U, M), where T is a set of *linguistic terms*, U is a *universe* of *individuals*, and M is a *meaning assigment.* In general, a meaning assignment is a mapping which gives interpetations for the members of T in terms of U. To illustrate, if Λ = Age, a possible choice of T might be the set {VERY YOUNG, YOUNG, MIDDLE-AGED, OLD, VERY OLD}, U might be the set of ages from 0 to 150 years, and M might be an assignment of terms in T to subintervals of U, e.g., M(VERY YOUNG) = [0, 12]. Alternatively, for terms $\tau \in T$, $M(\tau)$ might be a fuzzy subset of U (a possibility distribution in the sense of Zadeh [4]) or a probability distribution over U. We shall also allow that U be empty, in which case M is undefined; such would be appropriate, for example, in the case Λ = KINDNESS, for which there is no naturally occurring mesurement scale.

The universe U, when there is one, is used primarily for determining what term in T should apply to an individual A whose measurement along U is known. For example, if we again use the inverval [0, 12] as an interpretation of VERY YOUNG, and it is known that Francine's age is 7, then we conclude that this is the term in the term set for Age which should apply to A. On the other hand, such interpretations do not play a role in the inferencing scemes.

Inferences in the system are represented as expressions of the form

$$H_1, \ldots, H_n \Rightarrow C$$

where the *hypotheses* $H_1, \ldots, H_n$ and *conclusion* C are expressed via linguistic terms for some linguistic variables $\Lambda_1, \ldots, \Lambda_n, \Lambda$—e.g., H_i would be an expression of the form $\tau_i(x, y, z)$, where τ_i is from the term set for Λ_i. In such expressions, the variables x, y, z are assumed to range over individuals from the appropriate universe.

To simplify the present exposition, it will be assumed that all term sets consist of exactly seven terms: to wit, if Λ is a linguistic variable, with *primary term* λ (e.g., the primary term for Height might be TALL), then T_Λ will consist of the terms

VERY λ
λ
MODERATELY λ
med(λ)
MODERATELY ant(λ)
ant(λ)
VERY ant(λ).

Wherever appropriate, ant(λ) may be replaced with an antonym of λ, and med(λ) may be replaced with some intermediate term. We furthermore assume that the terms

* This work was supported by the Office of Naval Research, Grant Number N00014-87-G-0219.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

in T_Λ are linearly ordered as shown by a relation $\leq$, with VERY λ being greatest and VERY ant(λ) being least.

Inference Scheme A

In the first scheme to be discussed, the H_i and C each use a single linguistic term from the term sets for the corresponding Λ's, and the inference itself is interpreted as a procedure in which, given some terms $\tau'_1, \ldots, \tau'_n$ from $\Lambda_1, \ldots, \Lambda_n$, describing specific individuals $a_1, \ldots, a_m$, the appropriate conclusion τ' for those individuals (or subset thereof), in the termset for Λ, is computed in terms of a *distance* measure δ defined for linguistic terms: τ' is taken to be the term for which the distance $\delta(\tau, \tau')$ is closest to the sum

$$\sum_{i=1}^{n} \delta(\tau_i, \tau'_i).$$

In the simplest case for δ, the distance between successive terms in the upward direction is +1. To illustrate, let K=KIND, C=CONSIDERATE, and G=GENEROUS, suppose the inference is

$$\mathrm{K(X)}, \mathrm{C(X)} \Rightarrow \mathrm{G(X)},$$

and suppose individual A is determined as being KIND and MODERATELY CONSIDERATE. Let MOD abbreviate MODERATELY. We have

$$\delta(\mathrm{K}, \mathrm{K}) + \delta(\mathrm{C}, \mathrm{MOD\ C}) = 0 + (-1) = -1.$$

Thus, since $\delta(\mathrm{G}, \mathrm{MOD\ G}) = -1$, we conclude that the latter term applies to A. As another example, if A is VERY CONSIDERATE and VERY KIND, yielding a sum of +2, then the rule dictates that A is VERY GENEROUS. In this way, such an inference may be taken as expressing a "rule of thumb."

The above summation algorithm suggests several variations. For example, instead of the simple sum, one might weight the hypotheses in order to specify that some are more important than others. Such modifications to the present approach will be considered in a future work.

Inference Scheme B

In the second form of inference, the H_i and C are expressed as subsets of the term sets for the corresponding Λ_i and Λ, an individual A is characterized along each Λ_i by a similar set of terms for that linguistic variable, a hypothesis H_i is *satisfied* for an individual A if the set of terms ascribed to A is a subset of the set of terms given by H_i, and the conclusion set, i.e., the set of terms given by C, is ascribed to A just in case all the hypotheses are satisfied. To illustrate, again let MOD abbreviate MODERATELY, let AL abbreviate AT LEAST, and suppose that

AL K stands for the set {K, VERY K},

AL MOD K stands for {MOD K, VERY K},

that AL G and AL MOD G are defined similarly, and that the inference is

$$\mathrm{AL\ MOD\ K(X)} \Rightarrow \mathrm{AL\ MOD\ G(X)}.$$

Then, for an application of this inference, suppose that A is determined as being AL KIND. Since

$$\mathrm{AL\ K} \subset \mathrm{AL\ MOD\ K},$$

the hypothesis of the inference is satisfied, and we may conclude that A is AT LEAST MODERATELY GENEROUS. Note that in this examle there has been some loss of *specificity* in the knowledge about A: the set of Kindness terms ascribed to A contains only two terms, whereas the derived set of Generosity terms for A contains three.

It will be required that the subsets of terms, such as used above for defining AT LEAST, are continuous sequences from their repespective term sets, i.e., they do not contain gaps.

Evidence Combination

In general, the problem of designing a reasoning system based on either of the inferencing schemes, taken seperately, is that of defining an appropriate method of evidence combination. More exactly, suppose one has a collection of inferences

$$\begin{aligned} \mathrm{H}_{1,1}, \ldots, \mathrm{H}_{1,n_1} &\Rightarrow \mathrm{C}_1 \\ &\cdots \\ \mathrm{H}_{m,1}, \ldots, \mathrm{H}_{m,n_m} &\Rightarrow \mathrm{C}_m \end{aligned}$$

where the C_i are all composed of terms from the same linguistic variable, and suppose that more than one of these inferences apply. Then the problem amounts to crafting a rule for combining the corresponding C_i into a single C.

Such a rule for use in conjunction with Scheme A is as follows. Let $\tau_1, \ldots, \tau_k$ be the distinct terms appearing among the C_i. For each application of the above inference system (in which some or all the inferences might be used), let κ_i be the number of times that a conclusion involving τ_i is inferred. Each such κ will be referred to as the *affirmation count* for the corresponding τ, reflecting the view that to infer τ is equivalent with affirming τ for the individuals in concern. The rule of combination then uses the affirmation count as a form of weighting. Suppose that in some application $\kappa'_1, \ldots, \kappa'_k$ are all maximal. Then (i) if $k = 1$, let C be composed of the τ corresponding to κ'_1, (ii) if $k > 1$ is odd, let C be composed of the τ which is intermediate among the corresponding τ with respect to the ordering $\leq$, and (iii) if $k > 1$ is even, let C be composed of the largest of the two intermediate terms among the corresponding τ. It can be argued that this rule has an inherent intuitive appeal.

For Scheme B, a variation on the above may be used. Here, since conclusions are represented as sets of terms, the affirmation counts may be defined for such sets, and the ordering $\leq$ may be extended to a partial ordering of the same sets in the natural way ($S \leq S'$ iff $S \subseteq S'$). Then the above rule may be applied in situations where the sets with maximal affirmation counts are all mutually related by the extended $\leq$. In case some of these sets are not related by $\leq$, another algorithm must be devised, e.g., one favoring sets which are least or most specific among

those with maximal κ. Once again, this should yield an intuitively appealing rule of evidence combination.

Synthesis of the Two Methods

Extending the above rules of combination to systems which simultaneously involve both types of inference is straightforward. One simply treats the τ in the conclusions of inferences of type A as singleton sets and applies the combination rule devised above for inferences of type B.

Of course, a complete system must also have forward-chaining and back-chaining techniques. Forward chaining for each of the above inference schemes, taken seperately, is provided for by the schemes themselves. The task of forward chaining when the first inference is of type A and the second is of type B requires treating the τ in the conclusion of the first rule as a singleton set, so that it then has the proper form for consideration as a hypothesis for the second. When the first rule is of type B, and the second is of type A, a suitable chaining method can be obtained by treating the τ in the hypotheses of the second rule as singleton sets and extending the distance measure δ to a measure δ_s defined on sets of terms. One such extension might be: where $T_1, T_2 \subset T$, let

$$\delta_s(T_1, T_2) = \min_{\tau_1 \in T_1, \tau_2 \in T_2} \delta(\tau_1, \tau_2).$$

Corresponding backward-chaining techniques are currently under development. These are somewhat more complex than for systems based on classical logic but, at this writing, seem at least within the realm of being computationally tractable.

References

[1] Lee, M.-J. and Schwartz, D.G., A scheme for logical inference with linguistic attributes. *Proceedings of the Florida Artificial Intelligence Research Symposium, FLAIRS-88*, Orlando, Florida, May 4–6, 1988, pp. 208–212.

[2] Schwartz, D.G., A computationally simple interpretation of linguistic decision rules. *Proceedings of the North American Fuzzy Information Processing Society, NAFIPS-88*, San Francisco, California, June 8–10, 1988, pp. 216–220.

[3] Zadeh, L.A., The concept of a linguistic variable and its application to approximate reasoning, Part I: *Inf. Sci.* 8 (1975) 199–249; Part II: *Inf. Sci.* 8 (1975) 301–357; Part III: *Inf. Sci.* 9 (1975) 43–80.

[4] Zadeh, L.A., PRUF—a meaning representation language for natural languages, International Journal of Man-Machine Studies 10 (1987) 395–460.

Toward a Pictorial Knowledge-Based Approach to Facial Expression of Emotion *

A. Abd-Aziz, Edward T. Lee and M. Gruenberg
Department of Electrical & Computer Engineering
University of Miami
P.O.Box 248294
Coral Gables, FL 33124

This paper describes the compilation of a knowledge base of facial expressions by applying the concept of machine vision. The goal of the knowledge base is to be able to predict generalized human emotions from the descriptions of an arbitrary facial expression. Actual implementation of the knowledge base in an expert system is also presented. In the development of the knowledge base, graphical representations of facial elements in the form of picture primitives are interpreted using the Picture Description Language (PDL) and encoded in the software. To clarify the PDL interpretation, graphical approximations of the picture primitives are also provided.

Human emotions are described by feelings, which are sensations of joy, sorrow, hate, etc. These sensations are normally determined by the inner-state of a person and/or his surrounding environments. It is quite difficult to predict correctly the feelings of a person in a situation; however, from the facial expressions, distinctive feelings can be deduced. This, however, may not necessarily be the true feeling since a person is capable of hiding his emotion. For simplicity, we assume that facial expression relates directly to generalized feelings.

The concept of machine vision have been applied throughout the process of compilation of the knowledge base. The basic components of machine vision system and the scope of research covers in this paper are shown in Figure 1. Hardware implementation will be presented in subsequent papers. Horn suggested three elements of machine visions: image processing, pattern classification and scene analysis. Correlations of the above mentioned elements with the current work are as follows:

Image processing - The complex features of the face are first approximated by simple geometric shapes. Then, facial elements which do not contribute significantly to the determination of emotions are excluded. The face features are further reduced using PDL to a set of chain codes.

Pattern classifications - A set of nine photographs of facial expression describing fundamental human emotions is approximated using the above method. Common attributes from the reduced elements are compiled and a knowledge tree is constructed. By observation of an arbitrary face, an observer, using heuristic search, will match the description of the input image with the stored patterns.

Scene analysis - The goal of the knowledge base is that after a successful match is encountered, the machine will predict a generalized human emotion.

The results of such analysis can be further applied to research in robotics, human-computer interface, or neurolinguistic programming to aid counselors.

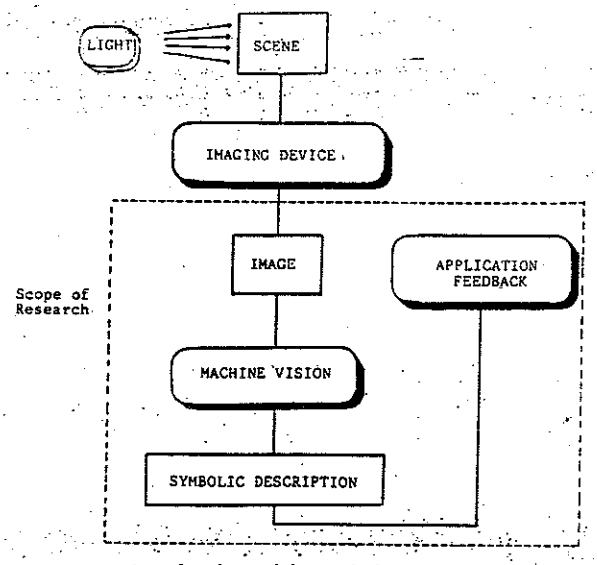


Fig. 1. A machine vision system

* This research was supported by a grant from Florida High Technology and Industry Council.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

OPSEM: A TOOL TO ALLOW SEMANTIC NETS IN OPS5

Sriram Adhyapak Erich Hentschel Jai Navlakha
School of Computer Science
Florida International University
University Park, Miami, Florida 33199

1 Introduction

OPSEM is tool that allows the user to represent knowledge in a semantic network based scheme. Building the semantic net involves defining classes with attributes and specifying relations between the classes. The essence of OPSEM is embedded in two menus. Menu options are the actions that OPSEM can perform.

2 Why OPSEM ?

Since OPS5 does not handle relations between objects it is very difficult to represent knowledge that is based on relations. OPSEM enhances the Knowledge Representation scheme of OPS5 by complementing it with the Semantic Network Knowledge Representation scheme. A user may build a schema by using OPSEM, then the user can include OPSEM literalizations and the schema created above into any OPS5 application program.

3 Functionality of OPSEM

OPSEM operations are implemented as menu options which are described below:

1::**CLASSIFY** This option allows the user to create new OPS5 classes. A unique name is associated with each class. Attributes and their default values may be given to a class at the time of its creation.

2::**CLASS–RELATE** Once one or more classes have been created relations between them can be established by specifying the Domain–Class name Relation name and then Range–Class name. The **ISA** relation defines the Domain–Class as a subclass of the Range–Class. Attributes and relations of superclasses are inherited by subclasses. A relation type is by default many:many and the user may specify the relation type to be 1:many, many:1 or 1:1.

3::**INSTANTIATE** An instance is a realization of a class that has been created. The class name of which an instance is to be created must be given, then all attribute values can be specified. The default values will be assumed for unspecified attribute values.

4::**RELATE** This option is used to establish relations between instances, where the relation between the corresponding classes already exists. It is nescessary to supply information about both instances involved in order to establish the relation. To identify an instance the user must supply the class name to which it belongs, one attribute name and the value of the attribute.

5::**DELETE–CLASS–RELATION** This option allows the user to remove a relation between classes. The existing relation links between the instances are also removed.

6::**DELETE–INSTANCE–RELATION** Relations between instances are undone by removing a link that was established during the RELATE operation.

7::**DELETE–INSTANCE** This option allows the user to remove existing instances. Instance relations in which this instance participates are also removed.

8::**DELETE–CLASS** Classes may also be deleted but it is a restricted operation since an entire subnet of the network could be lost this way.

9::**INVOKE QUERY–MENU** Its purpose is to supply information about the knowledge that the semantic network holds. Typical options include LIST ALL CLASSES WITH THEIR ATTRIBUTES, SHOW THE CLASS HIERARCHY, SPECIFY TWO INSTANCES AND FIND IF THEY ARE RELATED, etc.

4 Principles of Implementation

The classes in OPSEM are implemented as OPS5 OAV triples, as also their attributes. The attributes are linked to the class by a unique identification number. Relations are implemented by domain and range links which contain the ID numbers of two classes (or instantiations). The ISA relation is treated as a special relation.

5 Conclusion

OPSEM provides a means of representing knowledge as a Semantic Network, within OPS5. The user can either include the tool as a whole or include only an image of the working memory along with the literalizations, in an OPS5 application program.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

KNOWLEDGE-BASED PLANNING FOR PRELIMINARY MECHANICAL DESIGN

Jeffrey S. Bohren and Gale E. Nevill, Jr.
Department of Aerospace Engineering,
Mechanics and Engineering Science
University of Florida
Gainesville, Florida 32611

ABSTRACT

Design is a vital but poorly understood, engineering activity for which automation appears to have a high potential payoff. In both computational and theoretical models of design, the process is commonly divided into different phases including preliminary design and detail design. These are commonly further divided into several activities, such as problem abstraction, problem decomposition, solution planning, and solution implementation. The focus of the work reported here is a planning and decomposition framework, for use in preliminary design, to satisfy multiple goals.

The paper presents an approach taken in the MOSAIC project for automating the preliminary design of mechanical structures. Design tasks involve the generation of promising structural configurations to stabilize objects and point forces using rod, column and beam components. Typical goals include requirements on weight, cost, stiffness, natural frequency, and design techniques.

A top down refinement design process model using multiple levels of abstraction, iteration and decomposition has been chosen. The process begins with the creation of a top level abstract problem description. Abstract features and abstract operators are used to represent the design task. Problem solving knowledge is attached to separate goal objects associated with each performance goal and constraint of the problem. A basic iterative strategy is followed in which the current problem state is evaluated, candidate design moves are generated, the implications of the alternative candidate steps are predicted and the step judged best is chosen for commitment.

At this high level of abstraction, problem complexity is reduced so that an abstract solution to the design can be generated with a minimum of look-ahead or back-tracking. At this abstract level, the differences between various classifications of goals are minimized, allowing for goals to be treated in a uniform fashion. The uniform treatment of different classes of goals enhances the ability to treat multiple goals.

The goals present are classified into different types, such as performance goals, constraints, and design methodologies. Each type of goal is treated the same by the control structure. Knowledge about where in the planning process a goal is applicable is attached to the goal object.

Candidate design steps are generated and selected only when they explicitly support achievement of one or more design goals. Two types of design steps are utilized. The first involves decomposition and identifies subproblems in terms of abstract features and abstract goals. The second involves selection of abstract operators representing subproblem solutions at the current level of abstraction. The output of the planner is an ordered set of subproblems, which, when refined further, generate a complete preliminary design.

The principal contributions of this work are the development of a flexible framework for the achievement of multiple goals in an abstract planning/problem solving process, and demonstration of the successful use of abstract level planning to guide automated mechanical design.

Acknowledgement This work was supported by the National Science Foundation under Grant DMC-8813808.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

CONSTRAINT GUIDED SUBPROBLEM REFINEMENT IN KNOWLEDGE-BASED DESIGN

Jeffrey P. Brown, James H. Clinton and Gale E. Nevill, Jr.
Department of Aerospace Engineering,
Mechanics and Engineering Science
University of Florida
Gainesville, Florida 32611

ABSTRACT

This paper describes work in the MOSAIC project on automating preliminary mechanical structure design. This involves the generation and evaluation of promising, viable configurations which can then be "fleshed out" in a detail design phase. Preliminary design is particularly important in the overall design process since it establishes the basic nature of the final design, and thus strongly influences its quality.

In this work, preliminary design is modelled as an iterative, top down refinement process. Abstraction and decomposition are used to manage complexity. Decomposition divides a problem into sets of smaller subproblems which are (hopefully) easier to solve. This process is repeated until each subproblem becomes readily solved. However, subproblems often interact. These interactions must be managed to avoid conflicts and to take advantage of possible synergisms.

The MOSAIC domain is two-dimensional mechanical structures. Design tasks involve the generation of promising structural configurations to stabilize objects and point forces using rod, column and beam components. Typical goals include requirements on weight, cost, stiffness and natural frequency. The automated design capability described is implemented using InterLisp-D and LOOPS running on Xerox 1108 Lisp machines.

This paper focuses on generating good refinements, not on the control issues of choosing the next subproblem for refinement. The paper describes a scheme for incrementally generating subproblems and managing potential interactions. This scheme consists of four phases: evaluation of the subproblem state, suggestion of promising operators, prediction of each operators's implications for providing a suitable solution, and decision on which operator should be selected (committed to). A partial solution is incrementally developed by this process until complete. Since the goal is a promising preliminary design, backtracking currently is not allowed, and a commitment is made for each design step selected. To avoid interfering interactions and to coordinate facilitating ones, MOSAIC has a constraint anticipation and satisfaction capability.

MOSAIC has available a collection of operators which define the set of possible design steps at each point. After the current state of a subproblem is evaluated, a rule-based system determines which of the available operators are applicable by checking against current constraints for compatibility. If none are applicable (because of conflicts or inappropriateness), MOSAIC currently reports failure.

When more than one operator is applicable, some form of look-ahead to predict the implications of each choice is desirable. Inadequate look-ahead leads to poor choices while extensive look-ahead is computationally very expensive. Our approach is to carry out a one level deep simulation of the action of each refinement operator, using constraint knowledge to predict its effect(s). These predictions are then used as the basis for choosing the operator and thus the design step taken.

The approach described works well for a broad range of problems in this domain and appears to have considerable generality.

Acknowledgement This work was supported by the National Science Foundation under Grant DMC-8813808.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

COUPLING A MICROCOMPUTER EXPERT SYSTEM WITH ANIMATED GRAPHICS FOR COMPUTER SECURITY AWARENESS TRAINING

Dr. Bonnie Jo Buck
Advanced Technology, Inc.
1545 Orange Avenue
Winter Park, FL 32789

ABSTRACT

This Microcomputer Expert System is designed to run on IBM compatible computers throughout the Government to provide computer security awareness training. Animated graphic simulations of computer security threats and policies are coupled with an intelligent tutoring module. This is accomplished using Level Five (a low cost shell) and PC Paintbrush (a graphics program). The knowledge base contains the knowledge of a security agent. Through rule-based knowledge representation, a game and simulation environment is created. The student is involved in a video presentation and/or exercise. This creates a mastery learning environment where transfer of learning accommodates individual learning styles in an enjoyable and positive way. Furthermore, the new Expert System technology is low cost!

INTRODUCTION

The purpose of this project was to develop an instructional media to increase the awareness of computer security needs; motivate agency personnel to develop agency specific training and quality assurance programs; and to provide training which meets the individualized differences of all participants in the Federal community.

The expert system is an alternative instructional media which queries the user in a non-threatening manner to determine the student's knowledge and understanding of computer security. The student receives immediate positive feedback and reinforcement after each response in order to ensure and maintain a high level of student motivation and interest in completing the lesson. The expert system, based upon the student's responses during the dialogue/ conversation, processes the responses, draws conclusions, and provides recommendations to the student. The system also monitors and records the student's progress via a knowledge tree. Test and measurement strategies are conducted through a conversation/dialogue interaction with the student. The instructional strategy is based upon a series of cognitive learning experiences on computer security with skill practices presented in a simulation format combining text and animated graphics. The Computer Based Instruction (CBI) begins with an introduction that presents an overview of the lesson and the terminal learning objectives. The expert system then proceeds to instruct the student on the threats and vulnerabilities of computers through a dialogue that consists of information presentation and questioning techniques. During this dialogue, the expert system simultaneously evaluates and tracks the student's learning. For example, the system will define a computer threat and present several graphic illustrations in which the student has to recognize and identify the types of threats contained in the illustrations. The last segment of the CBI instructs and demonstrates how to apply the acquired knowledge and skills to specific needs and areas of concern. Several sample scenarios with hypothetical organizations/agencies and computer security problems have been developed. The student is required to identify a goal or objective (e.g., to maintain a database), state the problem (e.g., threat to data), and formulate a solution (e.g., develop a countermeasure against a computer threat). The estimated length of training will vary with each individual, but should average 30 minutes.

SUMMARY

The research described above has been demonstrated to government agencies at numerous conferences in the Washington, D.C. area. The responses have been very supportive of this new technology which couples a microcomputer expert system with animated graphics. The techniques used in this project can easily be adapted to many application areas through knowledge representation.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

Ada and Artificial Intelligence : An Overview of Current Research

Maria A. Cianci, BS
Darrell G. Linton, PhD, PE
Computer Engineering Department (CEBA I - 207)
University of Central Florida
Orlando, Fla. 32816
407 - 275-2236

The Department of Defense's (DoD's) requirement (directives 3405.1 and 3405.2) that Ada must be used in the development of military software has begun to impact research in the field of Artificial Intelligence (AI). In this paper, three areas of research in Ada and AI will be addressed :
1. ADALOG, a new programming language that integrates software engineering principles and AI inference features,
2. frameworks for the development of Expert Systems (ESs) and
3. inference engines developed in Ada.
Each of these areas will be detailed with respect to the objectives of the research, what has been achieved and the inadequacies (if any) of Ada as perceived by these researchers.

In the area of new programming languages a typical example is ADALOG, a language designed at the University of Nice (France). The primary goal of this research was to provide a programming environment capable of integrating procedural schemes and deductive reasoning capabilities. The result of this research is a language that supports separate programming in standard Ada and Prolog-like code.

Since the incorporation of Ada in AI models, substantial progress has been made in the development of frameworks for expert systems. This paper addresses two different frameworks : Embedded Rule-Based System (ERS) and Ada Blackboard Environment (ABLE).

ERS was originally written in C. The latest version of ERS is an Ada ES shell designed to satisfy U.S. DoD directives 3405.1 and 3405.2. It is a Rule-Consultation system which incorporates probabilistic inference schemes similar to those in Prospector and EMYCIN.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

A system similar to ERS, called ABLE, is being developed at Boeing Military Airplanes. The ABLE system combines the design of BFS (Boeing Frames System, implemented in LISP) and architectural characteristics of Erasmus. At present, the incorporation of all the features of Erasmus into ABLE is still not complete.

The mechanisms to be used in the implementation of inference engines in Ada are still under research. Two models are discussed in this paper : the Ford Ada Inference Engine (FAIE) and an unnamed generic inference engine developed at Planning Research Corporation.

The FAIE prototype started as a feasibility study for using Ada to develop rule-based expert systems with real-time performance require- ments. The finished prototype is an Ada inference engine for general purpose expert systems whose rule-firing rate is over 500 rules/second running on a single processor.

The objective of the unnamed generic tool designed by Marc Adkins at Planning Research Corporation was to construct a generic inference engine that would make use of the multi-tasking capabilities of Ada; however, the outcome is a rudimentary generic inference engine.

The primary goal of this paper is to provide an overview of AI software currently implemented in Ada. Recent research in Ada and AI is discussed with respect to what has been achieved to date and the problems encountered during the implementation of these systems. The influence of Ada and software engineering methodology on the development of future AI systems is also detailed.

ACKNOWLEDGEMENT

This research was supported by the State of Florida, Intelligent Simulation Training System, University of Central Florida, Orlando, FL 32816; Dr. John E. Biegel, Project Director.

USING CONSTRAINTS TO CONTROL SEARCH IN KNOWLEDGE-BASED DESIGN

James H. Clinton, Jeffrey P. Brown and Gale E. Nevill, Jr.
Department of Aerospace Engineering,
Mechanics and Engineering Science
University of Florida
Gainesville, Florida 32611

ABSTRACT

This paper is concerned with automating the search process in preliminary mechanical design. This involves the generation of promising, feasible concepts and configurations which are then "fleshed out" in a detail design phase. Preliminary design is particularly important in that it establishes the fundamental design concept and thus an upper bound on product performance. The paper presents an approach taken in the MOSAIC project for automating the preliminary design of mechanical structures. Design tasks involve the generation of promising structural configurations to stabilize objects and point forces using rod, column and beam components. Typical goals include requirements on weight, cost, stiffness and natural frequency. The automated design capability described uses InterLisp-D and LOOPS running on Xerox 1108 Lisp machines.

A top down refinement model using multiple levels of abstraction, iteration and decomposition has been chosen. The effectiveness of this strategy is strongly dependent on the management of subproblem interactions so as to avoid conflicts. This paper describes an approach based on the use of both anticipated and consequent constraints associated with subproblems. Anticipated constraints are constraints on subproblems which are predicted but not yet instantiated. They commonly involve information which is either essential or very desirable for subproblem refinement. It is undesirable to refine a subproblem until the information associated with anticipated constraints becomes known. Consequent constraints are constraints on other subproblems arising as a consequence of commitment to a particular subproblem. Consequent constraints thus impose requirements on the selection and refinement of subproblems and provide information needed by anticipated constraints.

A critical decision in iterative design processes is whether to further refine the subproblem of current focus or to change focus and refine some other subproblem. MOSAIC makes these decisions using an opportunistic commitment strategy in which subproblems are refined as soon as they are "well constrained" (when they have no anticipated constraints, either because none are present initially or all have been transformed into local constraints as a result of additional information becoming available). Thus well constrained subproblems are considered to have available all of the information required so that they may be refined effectively, with minimum danger of conflicts.

However, if anticipated constraints are present, a subproblem is called under constrained, the subproblem is suspended and focus is returned to its parent. The controller seeks incompletely refined subproblems labeled well constrained (i.e. having sufficient interaction knowledge) for refinement. When only suspended subproblems remain to be refined, there is a deadlock. When deadlock occurs, the suspended subproblem considered best constrained (i.e. having the most interaction information), is chosen for refinement focus and its anticipated constraints are converted into local constraints using heuristic knowledge.

This approach allows MOSAIC to combine many of the advantages of both a pure opportunistic search strategy and a least commitment strategy and has substantial generality.

Acknowledgement This work was supported by the National Science Foundation under Grant DMC-8813808.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

DESIGN OF AN INTELLIGENT USER INTERFACE FOR TUTORING SYSTEMS

Thomas W. Diefenbach, Lois W. Hawkes, Sharon Derry
Computer Science & Psychology Departments
Florida State University
Tallahassee, Florida, 32306-4019

This research has been partially funded by the Florida High Technology and Industry Council.

The quality of the user interface (IF) very often is a significant factor in determining the acceptability to the user of the associated system.

One of the major reasons that man/machine IFs are not remarkably more user friendly and sophisticated is because they are designed in a static fashion. Once the subject domain of human/machine interaction is specified and the I/O hardware chosen, the user IF is typically designed to perform every essential task in one unique way. This implies that the communication is unable to adapt to the various situations that can arise during the interaction between a user and the machine.

A more dynamic behavior in the IF could improve not only the effectiveness of the interaction but also the overall performance of the whole system. In order to achieve such dynamic IFs, it seems appropriate to include AI techniques in their design. Such an approach is being tested in the design of a user IF for an intelligent tutoring system, the domain of which is arithmetic word problems primarily for 4th to 7th graders. It is not sufficient for a tutor to be knowledgeable of the subject domain. The tutor must also be able to transfer this knowledge effectively to the student. Therefore a good human teacher must be able to adjust his way of communicating with the student depending on the student's behavior. This principle also holds for machine tutoring. A sophisticated tutoring system can only be effective when the communication between the tutor and the student is equally sophisticated.

This project involves studying the effect of incorporating a knowledge base into the user IF, making it capable of 'smart' communication skills. This can be applied to the human machine interaction in addition to the information the tutor needs to channel through the IF to the student. Based on the age of the student, for example, the IF might change the level of vocabulary used for its conversation or might decide to use cartoon figures to present the messages. Additionally the IF might try to apply several of its embodied communication strategies to the user until it establishes a stable suitable structure of communication with that user.

Other capabilities would involve analyzing whether the user is working seriously or whether he is just playing with the keyboard or clicking messages away immediately, indicating that he could not possibly have read the text.

Often when a user starts working with new software products he repeatedly tries erroneously to invoke particular tasks, based on experiences with other products. Usually the user then has to adapt to the way the IF requires him to perform the task. On the other hand, the intelligent IF would be able to detect in many cases the specific way the user chooses to perform the task. After verification with the user the IF can adapt to the user by allowing him to work with the product in the way which is intuitive to him or her.

All the individual peculiarities of the users will be kept in a database that is exclusively used and maintained by the intelligent IF.

With all these capabilities the man/machine interaction will adapt incrementally to every user in order to make the attached product as user friendly as possible. In the extremist view it might look as if every user has his own private customized IF to a specific product. But obviously there are also some trade offs to consider:

One of the major design criteria for a user IF is that its use should be consistent. But this requires a static approach of communication. Therefore, the extent of the flexibility accepted by the IF should be explored.

Another trade off addresses the overhead added by such an intelligent IF. In many systems the additional processing time would be unacceptable for the user. This may explain the lack of research in this area. On the other hand, distributed systems are becoming increasingly available. Using such an environment for a sophisticated IF, the added 'intelligence' can execute in parallel to the actions of the IF that are directly initiated by the attached program. Therefore this overhead would not be of concern in such a system.

In summary, the goal of this research is to produce a model for future man/machine IFs, using AI techniques, that is user friendly through being 'usersensitive'.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

The Application of AI and Fuzzy Logic Techniques to Software Fault-tolerance

Lois Wright Hawkes, Sooyong Park
Computer Science Dept & SUS Center for Artificial Intelligence
Florida State University
Tallahassee, Florida

AI techniques in general and Expert Systems in particular have been applied successfully to a wide variety of problems. This has prompted the consideration of such an approach for important technical problems with less than optimal algorithmic solutions. The area of Software Fault-tolerance (SWFT) is such a problem domain. Reliable software is becoming more and more crucial as computers are used in an increasing number of critical settings. A significant degree of hardware fault-tolerance (FT) can now be achieved. The next step in obtaining highly reliable computing will come with Software Fault-tolerance. Of the two existing approaches, N-version programming, and recovery block, we are concentrating on the latter. This technique associates with each 'module' of the program, an acceptance test, which must be passed to continue program execution. There are signifcant problems with the existing acceptance test approach which we are addressing by designing a "Knowledge-based Acceptance Test" (KBAT). Expert system techniques and fuzzy variables will be used by the KBAT to make inferences concerning errors in the software. The KBAT will incorporate information from the programmer and from experts in the area, as well as build and use a history of faults. This new KBAT approach is being investigated using both the standard von Neumann architecture and parallel architecture.

The objective of the proposed KBAT is to detect, for each module, a software fault using the following three components: a) the calculated result, b) knowledge of the expected result and its purposes, and c) the history of previous faults.

If the knowledge-based acceptance test detects a fault then the faulty module is replaced with an alternative and the alternative is then executed. The process involves three interdependent activities: a) receiving an input and result from each module, b) determining if the input and result are acceptable or not, and c) maintaining the knowledge base of the history of faults.

To decide if a module is performing acceptably or not, there are two different reasoning techniques: a) given an input to a module, determine if it will cause a faulty result, b) given an input to, and output from, a module, determine if the output is acceptable or not. Besides these two, we can add another technique to improve software safety: c) when either of the above indicate a fault, determine if we can produce a safe output.

Fuzzy set theory is very useful in making computer decisions more human like. Zadeh(1965) introduced the concept of fuzzy sets to handle inexactness. He suggested grouping nonprecise objects into classes that do not have sharply defined boundaries. Each object then has a membership value in the class of ideal objects rather than simply a statement of belonging or not. Based on this concept, there have been many theories and applications developed. In building a KBAT, often the criteria used by the programmer or experiences of the expert can be abstract or subjective, and hence imprecise. By incorporating this type of fuzzy knowledge, the KBAT is able to make decisions which are more human like. Thus, a result similar to an expert supervising the execution of the software can be attained.

This domain presents a situation where the addition of "intelligence" should produce a significant improvement over the strictly algorithmic approaches used to date.

Zadeh, L. 1965. "Fuzzy Sets", *Information and Control*, Vol. 8, p. 338-353.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

REAL-TIME EXPERT SYSTEMS' ARCHITECTURE

Robert Hodson and Abraham Kandel
Department of Computer Science
Florida State University
Tallahassee, Florida 32306

The expert system concept which divides knowledge and inferencing into separate entities has proven successful in implementing solutions to complex problems. The knowledge-base is unique to a particular domain while the inference mechanism or inference engine may be common to several domains. Expert systems have traditionally implemented nonreal-time applications in part due to an awkward mapping of the expert systems' actions and knowledge representations to conventional computer architectures. This mapping causes bottlenecks in the system resulting in poor response time. New computer architectures tailored to the expert system concept could reduce response times allowing expert systems to be used in dynamic domains.

Real-time domains have characteristics which impose new requirements on expert systems. Foremost the system should have an awareness of its operating environment and be responsive to changing events. Awareness requires the system to detect events and responsiveness requires effective, timely processing of events. Time constraints may vary, but the most demanding environments will require hardware implementation of time consuming operations and exploitation of system parallelism. Progressive reasoning, also required to meet system timing, provides a means of producing the *best* solution in the allotted time.

Other system requirements imposed by real-time environments are robustness and temporal reasoning. Robustness is the ability of a system to respond effectively with incomplete or uncertain information, like noisy sensor data for instance. Temporal reasoning provides the ability to reason about time and ordered events. An additional derived requirement is the ability to integrate declarative and procedural knowledge. This requirement allows system users to take advantage of a declarative design approach while still having the flexibility to use procedural knowledge inherent in many real-time control applications.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

An architecture for real-time expert systems that meets the proposed requirements is currently being developed. A rule-based expert system was chosen as a foundation for the architecture because of its adaptability to real-time requirements. Event based processing supports awareness. Special purpose hardware incorporating several level of parallelism and dynamic priorities supports responsiveness. Goal driven inferencing, as a basis for hierarchical implementations, supports progressive reasoning. Fuzzy reasoning provides a means for handling uncertainty and is also an effective mechanism for reducing search space. Event times and primative timing operations support temporal reasoning.

The system supports high level inferencing in a parallel processing environment. A reconfigurable topology is used to interconnect the processors so the system's configuration can reflect the a application's domain. The network behaves as group of cooperating expert systems sending and receiving goals to each other. Additionally the processors have an environment interface for control applications.

The design of the individual processors reflect a rule-based inference engine with special hardware to support efficient processing of rules. Rules are processed through the system in a priority ordered dataflow fashion. Priorities are dynamic to focus processing on critical goals. When rules are fired in the system, a general procedure may be invoked. This integrates procedural knowledge with a rule-based triggering mechanism. Procedures can modify working memory, communicate with other processors or interact with the environment.

Acknowledgements

This research is supported in part by NSF grant ISI 8405953 and by Florida High Technology Council grant UPN 85100316.

A STUDENT KNOWLEDGE MODEL - FUZZY EXPERT SYSTEM FOR AN INTELLIGENT TUTORING SYSTEM

Dawn J. Holmes*, Lois W. Hawkes*, Sharon J. Derry**

*Department of Computer Science
**Department of Psychology
Florida State University
Tallahassee, Florida 32306

This paper discusses the design of a fuzzy expert system which will serve as the student knowledge model for an intelligent tutoring system (ITS). This system offers a novel approach in the way that student knowledge is represented and maintained within an ITS. Intelligence is being incorporated into the student model itself not just in the tutoring system, and the system's knowledge of the student will be stored using a unique representation, a Fuzzy Temporal Relational Database (FTRDB).

STUDENT KNOWLEDGE

Most ITSs provide a general knowledge model when representing the student's knowledge. However, by giving the tutoring system a more complete view of what is known by each student, the ITS is able to provide individualized or adaptive instruction. We have chosen a fuzzy temporal relational database (FTRDB) as the method of representing the student's knowledge, or student record for the following reasons:

First, use of a relational database provides relational operators for ease of data manipulation and user-friendly query languages. Relational databases also have the advantage of relatively speedy response times even when large amounts of data is stored.

Another advantage to the FTRDB approach is that fuzzy data may be kept in the student knowledge model. The existence of fuzzy data allows the system to provide imprecise, incomplete or vague information to the ITS. Thus the tutoring system can make decisions and judgements using the same type of information that a human tutor may use. An example of the fuzzy information that might be stored is "student X performs above average on moderately difficult problems most of the time".

Finally, the FTRDB will allow for the use of historical information. Temporal data may reduce the adverse affects of noise and inconsistency in diagnosing a student's performance level by allowing the student's development to be evaluated over a sequence of tutorial sessions.

FUZZY EXPERT SYSTEM

Data manipulations on the student record will be performed by a fuzzy expert system, the student record manager (SR manager), which acts as an interface between the student record and the tutoring system. As previously mentioned, the incorporation of intelligence within the student model is a new approach in ITSs. By building intelligence into the student model we are better able to emulate the actual tutoring process of human tutors.

The student record represents the system's knowledge of what the student knows. However, in reality, the human tutor is not only aware of the student's skills and abilities, but is also constantly forming new impressions of the student through analyses of the student's work performance and inferences drawn from his or her knowledge about the student.

It is this knowledge that is captured in the SR manager. The system uses data from the tutor as well as patterns observed from protocols of the student's performance to make inferences about the student's performance levels. This performance data as well as the system's knowledge about the student is then used to determine what, if any, modifications will be made to the knowledge in the student record.

Research partially funded by:
National Science Foundation grant IST-8510894
Florida High Technology and Industry Council

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

A Rule-Based Expert System for Network Configuration Management

Michael Humes - Julie LeBlanc
AT&T - IMS
Network Business and Management Systems
2301 Maitland Center Parkway
Maitland, Fl. 32751

Many large organizations today are involved with communication networks either as users of public networks or as operators of their own private networks. The management of these networks has become an important issue in the communications industry; there is great interest in and considerable pressure for development of techniques and strategies to effectively manage complex networks.

As network users have become more sophisticated in using the technology, their demands have become more complex. Large X.25 networks with over 150,000 terminals require a large staff to engineer and implement the 100 plus change requests received monthly. Users want rapid response to their requests for service modification. These modifications may be adding a new device to the network or possibly accessing the network from different locations; it may be requesting access to the new applications or to the latest communications technology as soon as it becomes available.

In addition to the users' increased and more sophisticated demands, there is a corresponding increase in the complexity of the networks. AT&T has combined the X.25 and SNA networks to create an expanded network referred to as XNA -- Extended Network Architecture. The multiplicity of hardware and software vendors adds yet another layer of complexity to the problem. Associated with network changes are many tasks ranging from calculating network capacity and circuit costs to updating the network database.

The purpose of this paper is to describe a development effort which treats one important aspect of the network design problem--that of engineering changes or modifications in a network subsequent to the initial design. This paper describes the expert system NESTOR -- Network Engineering System for Topology and Requirements. NESTOR was designed to aid engineers in implementing changes to AT&T's X.25 internal computer network.

The paper describes the manual process of engineering a network change; the hierarchial network configuration is also explained. The rationale for using an expert system approach rather than a conventional programming method is discussed. The paper then relates how the manual process has been incorporated into NESTOR with a rules-based system which accesses and updates an online global database used by the entire Network Management group. The software and hardware environments are addressed. The method by which NESTOR processes a network change is outlined and includes the following:

1. User interface
2. Execution of rules and procedures
3. Techniques used in accessing the database
4. Development of an Engineering Document
5. Techniques used in updating the database

A typical design problem is presented to illustrate the system's capability. To conclude, a brief discussion of benefits and limitations of NESTOR is offered.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

THE ROLE OF KNOWLEDGE REPRESENTATION IN KNOWLEDGE-BASED-SYSTEM DESIGN

Leslie D. Interrante, MSE and John E. Biegel, PhD
Intelligent Simulation Training System Project
Department of Industrial Engineering and Management Systems
University of Central Florida
Orlando, Florida 32816

Knowledge-based systems (KBSs) are computer programs which embody knowledge about a particular domain (or domains). It is important to spend some time considering the design of a KBS before building and implementing its components.

KBS design should focus particularly on the types of reasoning required in the application domain. How can these types of reasoning be achieved? These reasoning capabilities must be carefully designed into the system components. It is difficult, if not impossible, to modify or redesign a KBS in order to achieve more complex reasoning tasks. A mechanism must be provided at the outset to symbolically represent the information needed for drawing inferences in the application domain.

The extent and amount of rework in KBS development can be greatly reduced by including an explicit design phase in the KBS development process. Of the available knowledge representation paradigms, inferencing mechanisms, and description languages, those elements which most closely match the representation and reasoning needs for the application domain should be chosen.

The goal is to incorporate the required capabilities in the original design, rather than "evolve" them. The benefits of this method are as follows:

1. less time spent in the total KBS development effort,
2. ease of expertise elicitation (reduction of human/computer representational mismatch), and
3. better reasoning capability in the finished KBS.

The choice of application domain(s) (with associated reasoning requirements) and various programming considerations should drive the design process. Particular care should be given to design of the knowledge representation paradigm, as this paradigm affects all other elements of KBS design. This decision depends on the type of knowledge to be stored, which in turn depends on the application domain(s) for which the KBS is intended.

Related design elements include: knowledge sparsity/richness, the description language, bias, and use of meta-knowledge. How much structure or modeling (richness) is needed to capture domain concepts? How can this structure be represented/incorporated in KBS reasoning?

The description language expresses domain concepts and primitives. It is a language built up from the implementation language (e.g. LISP). The description language must provide a way to express such concepts as time, causality, and uncertainty.

Bias is the set of all factors which determine hypothesis selection, thereby affecting the decisions made by the KBS. Bias may be inherent in the knowledge representation paradigm or it may be explicitly stated in preference criteria and/or an evaluation function.

Meta-knowledge is knowledge which describes a KBS's internal world of representations. Uses of meta-knowledge include: explanation facility enhancement, provision of context-sensitive problem focus, automatic knowledge acquisition enhancement, dynamic control, knowledge-base debugging, and generation of rule-calling strategies.

In order to develop a KBS, one must possess more than the ability to implement various KBS elements in an "artificial intelligence programming language". It is necessary to identify pertinent features of the application domain(s) and to determine what kind of constraints and requirements these features place on the KBS to be developed. Appropriate KBS elements must be matched to these requirements. In other words, the KBS must be engineered. It must be carefully designed if it is to perform as a valid problem solver which aids (or replaces) human decision processes.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

AN ADVISORY EXPERT SYSTEM FOR SUPERCOMPUTER JOB CONTROL LANGUAGE

Karen Lee Johnson
Department of Computer Science
Florida State University
Tallahassee, Florida 32306

1. INTRODUCTION

A demonstration expert system has been developed to assist supercomputer users in making optimal use of complex and expensive supercomputer resources. The CLARA expert system, described here, advises users of the CYBER 205 supercomputer in constructing job control language statements that will execute their applications in an optimal fashion. Such an expert system is needed since there are too few human experts available to assist users.

Users can consult CLARA in a variety of ways, depending on their needs and interests. Available consultation modes include interpreting and debugging job control language (JCL) error messages, building control statements tailored to the user's supercomupter application, and a quick online reference mode and glossary.

2. SYSTEM DESIGN

2.1 Knowledge Base

The CLARA knowledge base contains all of the system's knowledge about CYBER 205 JCL. This includes general knowledge, such as legal statement formats, and knowledge about individual statements. The knowledge base also contains information about specific error messages, such as the statements that may trigger them and rules for debugging them.

A multiple knowledge representation scheme is required to fully express the system's domain knowledge. Frames provide the basic structure of the knowledge base. Production rules and procedural knowledge appear within some frame slots to completely represent the control language knowledge. Separate frames represent each control statement, each parameter of each control statement, and each error message.

2.2 Control

Inference control mechanisms are organized into frames similar to those which represent the knowledge base (Aikins, 1983). The basic tasks needed to control inferencing in each consultation mode are stored on a frame representing that mode. Overall control is provided through a central control frame. This frame-based control scheme is very helpful during system development, debugging, and modification.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

Tasks stored within control frames are executed by placing them on an agenda or list of tasks. Tasks are removed from the agenda for execution in last-in-first-out order. Tasks can be added to the agenda at various stages of a consultation from control frames or from other executing tasks. Execution continues until the agenda is empty.

2.3 System-User Interaction

In recent computer science literature there has been a great deal of discussion of the need for improved system-user interfaces in advisory expert systems. An expert system is worthless to a user if he or she cannot easily communicate the nature of the problem to be solved and understand and apply the solutions generated by the system. Accordingly, particular attention was paid to developing a workable system-user interface for the CLARA system.

The user has a great deal of control over the nature of the dialog between the system and user. By selecting a consultation mode (e.g. statement building or quick reference) the user determines the degree of system control over the consultation. At any point during a consultation the user may ask why a question is being asked of him or consult a glossary of supercomputer-related terms and phrases. Once a problem solution is reached, the user may ask how the solution was arrived at and why it is applicable.

User input errors are avoided to as great an extent as possible by requiring entry of the fewest possible keystrokes. Where errors are unavoidable, the expert system provides clear, concise, and constructive error messages to the user.

Screen design has been carefully considered to provide maximum readability for the user. This is especially critical in the online reference consultation mode. Information is clearly labelled and organized for easy identification.

3. SUMMARY

The CLARA system applies advice-giving expert systems technology to a new domain, that of supercomputer control language. This system represents a first step toward developing much needed intelligent software tools to help supercomputer users fully utilize expensive and scarce supercomputer resources.

REFERENCES

Aikins, J. S. 1983. Prototypical Knowledge for Expert Systems. *Artificial Intellignce* 20: 193-210.

Johnson, K. L. 1988. *An Advisory Expert System for Supercomputer Job Control Language.* Masters Thesis, Florida State University.

AN INTERVENTION MODULE FOR AN INTELLIGENT TUTORING SYSTEM[1]

Harald W. Kegelmann*, Lois W. Hawkes*, Sharon J. Derry**

Department of Computer Science*
Department of Psychology**
Florida State University
Tallahassee, Florida 32306

[1]Research partially funded by National Science Foundation grant IST-8510894 and Florida High Technology and Industry Council

ABSTRACT

This paper describes research in progress on an Intervention Module for an Intelligent Tutoring System, represented as a Fuzzy Expert System (Hawkes et al. 1988, (in press)).

INTRODUCTION

The Intelligent Tutoring System currently under development is designed to train 4th to 7th grade students to solve arithmetic word problems. At the heart of the system is the Intervention Module (IM). This module performs several complex tasks: it must follow the student's solution path, identify lower- and higher-order errors and decide on the most appropriate tutoring strategy.

To accomplish these tasks, the IM communicates with three other modules: a highly graphic *Interface*, incorporating menus, windows and icons, the *Expert Solutions Module* (ESM), which generates a tree representation of the solution or solutions, and the *Student Record* (SR), which stores and evaluates knowledge about each individual student.

In operation, the IM has to monitor the student by comparing his responses to the tree of acceptable solution paths in the ESM. Intervention of the student's performance is determined by two levels of error: On a lower-level, the module monitors such behavior as how frequently the student alters his response, as well as correctness in the selection of operators and operands. Current research also attempts to identify higher-order error patterns that occur across sessions and problems.

REPRESENTATION OF THE IM

Next to these domain specific constraints, a proper representation of the IM should also satisfy the following design principles: a *modular* and *well structured* design, to make the complexity of the system manageable, *great flexibility*, since the research is highly exploratory, and a *black box approach*, where each module is viewed as an independent unit, communicating with other modules by a uniform protocol.

These guidelines resulted in the selection of an object-oriented representation for the IM that also incorporates features to implement fuzzy production rules (Beckstein et al. 1987), (Kegelmann 1988). Lower-level objects represent the students' solution path, more complex objects are: *Tutor*, the central data structure and supervisor of the session, *Problem*, storing information about the problem to solve, and *Student*, which contains data about previous performances of the student.

Methods associated with objects perform minor tasks like updating data and displaying error and tutoring messages, whereas complex tutoring strategies are performed by fuzzy production rules. A typical such rule for a lower-level error response would be (in English):

IF student 'often' selects wrong_operator
and is 'rather_weak' on 3-schema_problems
and student has not been interrupted 'recently'
THEN suggest [closer_look_at_schema]

CONCLUSION

The proposed Intervention Module provides a novel application for Fuzzy Expert Systems, and a prototype system is currently implemented in CommonLISP. We feel that the significant degree of uncertainty involved in modelling the human mind makes the fuzzy approach not only appropriate but indispensable, and is a novel application of this AI technique for tutoring systems.

Bibliography

Beckstein, C.; G. Görz; M. Tielemann. "FORK: A System for Object- and Rule-Oriented Programming." In: *Proc. European Conference on Object-Oriented Programming*, June 1987, pp. 303-314.

Hawkes, L.W.; S.J. Derry; U. Ziegler; T. Diefenbach. "An Intelligent Tutoring System for Arithmetic Word Problems." In: *Proc. First Florida Artificial Intelligence Research Symposium*, 1988, pp. 245-249.

Hawkes, L. W.; S.J. Derry; A. Kandel. (in press) "Fuzzy Expert Systems for an Intelligent Computer-based Tutor." In: *Expert Systems in the Fuzzy Age*, A. Kandel (ed.), Addison-Wesley.

Kegelmann, H.; *FORK für Arbeitsplatzrechner*, Studienarbeit am Institut für Mathematische Maschinen und Datenverarbeitung der Friedrich-Alexander Universität Erlangen-Nürnberg, 1988.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

KNOWLEDGE ACQUISITION AND REPRESENTATION FOR THE AIR TRAFFIC CONTROL EXPERT SYSTEM

Andrew Kornecki, Richard Phinney, Stephen Letter, Ross Dukeshier, David Hanzlik

Department of Computer Science
Embry-Riddle Aeronautical University
Daytona Beach, Florida 32019
(904) 239-6681

ATEC - Air Traffic Expert Controller - is an expert system for Air Traffic Control (ATC). This research is part of a larger project on Intelligent Simulation Training Systems that has chosen ATC as its first application (Gonzalez et al. 1988).

The knowledge of an Air Traffic Controller can be divided into two distinct categories. The first of these is the knowledge that is derived from the standards and procedures manual. The other category consists of the expert controller's own experience. In an attempt to obtain the two types of ATC knowledge stated above, we first tried to go through the manual and write the rules, supplementing this knowledge with information gained by interviewing experienced controllers. This proved not to be a practical approach. A different approach was therefore needed.

There are many ways to represent knowledge in an expert system. The rule of thumb for ATEC was to represent the knowledge in a manner that best fits the ATC view of the real system. As the causal viewpoint is the most popular, a majority of the existing expert systems are rule based. The ATEC system also falls into this category.

In developing ATEC, it was found convenient to divide the operations of the expert system into different categories. These categories were developed through consideration of the various duties that ATC entails during the progression of an aircraft from departure through arrival. It was decided that the major areas an Air Traffic Controller will have to deal with are: initial clearance before takeoff, initial radar contact (after departure), pilot requests and emergencies, separation, handoffs, weather, traffic advisories, and arrivals. The ATEC system must be capable of handling situations from each of these categories.

Because of the multitude and complexity of the situations that a controller might face in each of these categories, we do not intend to cover all possible courses of action for all situations in the beginning. For the purpose of this project it has been decided that we shall attempt to handle all situations that occur 80% of the controller's time. The remaining 20% is left for future system expansion. It will be possible to supplement the knowledge base in the future as methodology for ATC changes, or the need is seen to extend the knowledge base.

The process of building ATEC consists of four stages repeated in an iterative cycle. First a representation of the ATC knowledge was agreed upon. The second stage is where knowledge is acquired through the utilization of the standards and procedures manual, training materials, and interviews with actual Air Traffic Controllers. Interactions with experts may have the form of structured or unstructured interviews, protocol and/or scenario analysis. The development process needs to be carefully documented (Everett and Dankel 1988). Facts and heuristics are extracted from the documentation and expert interviews. The third stage is where knowledge is designed into a program based on the knowledge representation scheme. Finally the testing and evaluation stage is performed, which may result in some deficiencies that have to be corrected by returning to the early stages of the knowledge acquisition process.

During the program design the knowledge was first expressed in a more computer oriented manner. Action facts were developed first which describe the possible actions a controller may carry out. Next, situation facts that would be present in order for the controller to perform the action were identified. Finally, the reasoning process or rule was developed that would lead to the appropriate action fact as a result of certain situation facts being present. The current version is composed of over 100 rules.

The ATEC base development machine is a Symbolics Lisp machine. We have been developing our system under the expert system shell available form Inference Corp., known as ART. We are now considering changing to another expert system design tool, Joshua, available from Symbolics. At present most of the data is represented by Lisp structures because a great deal of the information is coming from a simulation which is written in Lisp.

Now that the initial questions of which representation scheme to use and how to acquire and structure the knowledge have been answered, the task of testing and evaluating this knowledge has begun. Undoubtedly continued corrections will need to be made as already changes and additions have occurred. This is part of the repetitive development process. However, now that the initial design phase is nearing completion, steps are being taken to start the integration of the separate portions of the knowledge base into the complete ATEC expert system.

REFERENCES

Everett, P. A. & D. D. Dankel II. 1988. "A Study in Acquiring Knowledge From an Expert." In Proceedings of the 1988 Florida AI Research Symposium (Orlando, FL, Mar. 4-6). 81-85.

Gonzalez, A.; A. Kornecki; A. Ransom; P. Bauert; & R. Phinney. 1988. "A Simulation Based Expert System for Training Air Traffic Controllers." In Proceedings of the 1988 Florida AI Research Symposium (Orlando, FL, Mar. 4-6). 231-235.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

GENERAL OPERATION OF THE PROGRAM TO DRAW HASSE DIAGRAMS (HASSE DIAGRAM PROGRAM)

Christie J. W. Lamm
Department of Computer Science
Florida State University
Tallahassee, Florida 32306
(904) 644-2296

ABSTRACT

The Hasse Diagram Program takes an input file of coded representations of a series of Hasse diagrams, such as that generated by the program in (Løvstad and Bandler 1989), and outputs a graphical representation of each diagram. Each representation may be interactively manipulated. The program uses the SunCore (Trade Mark) graphics package (Sun Microsystems 1986), and is written in C. A detailed set of instructions for its use can be found in (Lamm 1988).

WHAT IS A HASSE DIAGRAM

A Hasse diagram is a pictorial method of representing the relationships among elements of an order in an organized manner. An order is a relation R from a set of elements, S , to itself that has the properties of reflexivity, transitivity, and anti-symmetry. In such a relation, it is always possible, because of the anti-symmetric property, to partition the elements of S into n exclusive subsets (numbered S_0 through S_{n-1}) such that, for each S_i, the only elements related by R to elements of S_i (aside from elements related to themselves) are those in $S_0, \ldots, S_{i-1}$. In a Hasse diagram, each of these subsets comprises a stratum, with S_0 corresponding to stratum [0], S_1 corresponding to stratum [1], etc. The elements of each stratum are drawn as circles (nodes) in horizontal rows. No particular order is assigned to nodes within a stratum. The nodes are numbered for identification. The strata are arranged from lowest to highest, with stratum [0] drawn at the bottom. Relationships between elements are drawn as lines (arcs) connecting the nodes. The Hasse diagram shows all the relationships of the original order with the following exceptions: By the transitivity property, if N1, N2,and N3 are three nodes such that N1 R N2 and N2 R N3, and therefore N1 R N3, then the line connecting N1 and N3 is considered redundant and is not drawn. By the reflexivity property, every node is related by R to itself. This relationship is assumed and not drawn. Details concerning the representation of orders with Hasse diagrams can be found in (Bandler and Kohout 1982), (Bandler and Kohout 1988).

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

USING THE PROGRAM

Details on the specific format of the input data file can be found in (Lamm 1988). The fewer the lines and nodes, the clearer the representation will be. When all the nodes have been read from the data file and stored internally, assessments are made for drawing purposes. The size of the grid is directly based on the size and the portion of the window used by SunCore. The size of the window can be adjusted at any time through mouse controls. The menu of action choices is shown at the bottom of the window, and the labels of the strata are shown at the left side of the window. The diagram is drawn within a grid, which is a rectangular box with horizontal lines separating each stratum. After the grid is drawn, the circles (representing nodes) and the arcs (representing relations) are added. The size of the circles is inversely proportional to the number of nodes in the largest stratum. The edge of each circle is a series of points; each point is shown with the * (asterisk) character. Circles are added symmetrically around the central vertical axis of the grid, and are centered in the horizontal space allotted for each stratum. After the diagram, strata labels and menu choices are drawn, the user may choose to move a node, name or number a node, refresh the screen, enlarge the diagram to its fullest size, view the next diagram, or exit the program. The mouse is used to choose an action. The user is not allowed to create or delete nodes or arcs, or to move a node into another stratum, since this would alter the meaning of the original Hasse matrix.

REFERENCES

Løvstad, S. and W. Bandler. 1989. "TRISYS/TRIMOD: A Knowledge Engineering Tool Using Fuzzy Relational Products". In *Proceedings of the Second Annual Florida Artificial Intelligence Research Symposium* (Orlando, Florida)

Lamm, C.J.W. 1988. *Manual for the Hasse Diagram Program* Unpublished, available from the author.

Bandler, W. and L. Kohout. 1982. "Fast Fuzzy Relational Algorithms". *Proc. Second Int. Conf. on Mathematics at the Service of Man (Las Palmas, Spain)*, Ballester, A.; Cardus, D. and Trillas, E.,eds., Universidad Politecnica de las Palmas. 123-131.

Bandler, W. and L. Kohout. 1988. "Special Properties, Closures and Interiors of Crisp and Fuzzy Relations". *Fuzzy Sets and Systems* no.26: 317-331.

Sun Microsystems. 1986. *Suncore Reference Manual.* 2550 Garcia Ave., Mountain View, CA 94043.

A STRATEGY FOR REPRESENTING SUBJECTIVE RULES

James E. Lamm and Daniel G. Schwartz
Center for Artificial Intelligence
Department of Computer Science
Florida State University
Tallahassee, FL. 32306 - 4019
904-644-2296
This work was supported by the Office of Naval Research,
Grant No. N00014-87-0219.

ABSTRACT

This paper outlines a strategy presented in (Lamm 89) for representing subjective rules such as: "tall people are good at basketball", "uncoordinated people are bad at basketball", etc., so that they may be used to make decisions.

1. PRELIMINARY CONCEPTS

This section defines several preliminary concepts, which are used in section 2 to represent subjective rules.

Characteristic A *characteristic* is a general property which can be ascribed to the various elements of some class of objects (where objects may include persons). Examples are Height (of people), Age (of civilizations), etc.

Attribute An *attribute* is a characteristic of a particular object or person. An example is the height of John.

Subjective Term A *subjective term* is a word which denotes a range of values that may be interpreted differently by different people. For example, "tall" is a subjective term.

Term Set For a Characteristic A *term set for a characteristic* C is a set of subjective terms which represent possible values of C. A characteristic may be simultaneously associated with different term sets. For example, if C is the height of people, then a possible term set for C is {short, medium, tall}. Another term set for C is {short, tall}.

Term Set For an Attribute Let C be a characteristic for a set of objects O. Let T be a term set for C. Let A be the attribute (C of o) where o is an object in O. T is a *term set for attribute* A (as well as for C).

Random Sample Let A be an attribute whose value has been measured precisely. Let T be a term set for A. A *random sample for A and T* is a set of responses obtained by picking one or more people (at random) and asking them:

"Which term(s) in T best describe the value of A"

Here, a person may respond by choosing a set of one or more terms.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

Tabulation Function Let A be an attribute and T be a term set for A. Let R be a random sample for A and T. The *Tabulation Function for A and T* (described in (Lamm 89)) maps R to an estimate of how reasonable each term in T is for describing the value of A.

Subjective Rule Let O be a class of objects. Let C_1 and C_2 be two characteristics of O. Let T_1 and T_2 be two term sets for C_1 and C_2 respectively. Finally, let S_1 and S_2 be two proper subsets of T_1 and T_2 respectively. A subjective rule is a statement of the form:

For any object o in O, the knowledge that S_1 contains the most reasonable terms in T_1 for describing the value of (C_1 of o) (as determined by the tabulation function) is a reason to believe that S_2 contains the most reasonable terms in T_2 for describing the value of (C_2 of o).

For example, the phrase:

"For any object p in the set of all persons, the knowledge that tall is the most reasonable term in {short, average, tall} for describing p's height, is a reason to believe that good is the most reasonable term in {bad, good} for describing p's basketball skill"

is a subjective rule.

2. BASIC STRATEGY

The basic stategy is to model how subjective rules influence decision making by comparing their effect to that of receiving certain responses when forming a random sample. For example, let A_1 be the attribute John's height and A_2 be the attribute John's basketball skill. Let T be the term set {bad, good}.

Suppose one wished to determine John's basketball skill and knew that "tall people are good at basketball", and that "tall" is the most reasonable term in {short, tall} for describing John's height. This would increase one's belief that good is the most reasonable term in {bad, good} for describing John's basketball skill. Now suppose one knew that a person, after watching John play, chose "good" as the best term in {bad, good} for describing John's basketball skill. This would also increase one's belief that good is the most reasonable term in {bad, good} for describing John's basketball skill. If the increases in belief due to each of the above cases are considered equal, a strength of 1 could be assigned to the rule "tall people are good at basketball".

REFERENCES

1) Lamm J.E. and D.G. Schwartz. 1989. *A Stategy for Representing Subjective Rules.* Available at poster session or send mail to "lamm@nu.cs.fsu.edu".

A NEURAL NETWORK IMPLEMENTATION OF AN INTELLIGENT CACHING ALGORITHM

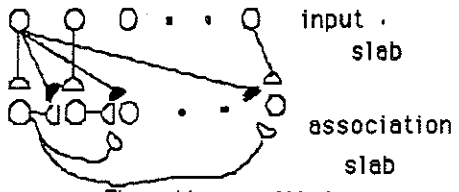
David Lawson
Department of Mathematics/Computer Science
Stetson University
DeLand, Fl 32720

Abstract. We implement a caching algorithm using an associative memory. We use a neural network implementation of list learning to predict the next page to be used. We then demonstrate a method of implementing the neural network prediction mechanism in conjunction with the LRU algorithm.

Introduction. Caching is an important method of improving computer performance. Caching is able to improve performance to the extent that the algorithm used is able to predict future behavior. Neural networks are able to learn. We propose that one use a neural network to learn the sequence of pages used in the past, and then use this sequence to predict behavior in the future. We also exhibit a method of combining the neural network predictor with the LRU algorithm, enhancing the performance of both.

Architecture. Each program will have its own neural network. This network will be able to learn different sequences of pages used by that program during execution.

Algorithm. A simple associative memory will can serve as a good predictor. The present page is associated with the next page to be used. A program may take different paths depending on the data. In such a situation a neural network can learn several lists and then pick the appropriate one as the program begins execution.

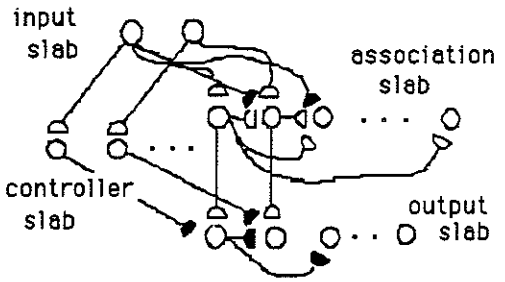


The architecture of List-learner.
Darkened synaptic knobs are inhibitory.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

Implementation. *List-learner, a model that can learn several lists.* This is a simple associative memory enhanced with inhibitory connections. This architecture is able to learn several lists.

Methods of predicting the next element in a list. If the association slab, has learned a list 1, 2, 3, . . , n, then one can discover that neuron k is followed by k+1 by picking the maximum of the connections k-to-1, k-to-2, . . , k-to-(k-1), k-to-(k+1), . . , k-to-n.



Winner-Take-All output slab with Controller. The predictor neuron will fire on the input and association slabs. The predicted neuron fire on the association and the output slab.

Combining the LRU with neural page prediction. The LRU algorithm can be implemented using a FIFO. As a page is used the number of the page is placed on top of the stack. If that page number is already on the stack, then it is removed from its former position. The least-recently-used page is then the page whose number is at the bottom of the stack. We combine the two methods to provide a method of implementing neural page prediction. First, load a LIFO with the pages predicted by the neural network. After n pages have been predicted load these n pages into a copy of the FIFO, reversing the order as you do, so that the next page predicted is then on top of the FIFO, followed by the page after that, etc. Duplicate pages, pages that are already in the FIFO are removed, as described above. Pages in memory that are not predicted are treated as they normally would be by the LRU algorithm, with the least-recently-used page (which has not been predicted) at the bottom of the FIFO.

Parallel Parsing Using Logic Programming: Natural Language Processing

Todd Paul Lehman
Florida Institute of Technology
150 W. University Blvd. #5531
Melbourne, Florida 32901

1. Abstract

The origins of this paper lie in the interests of natural language processing and logic programming. Syntactic parsing, a phase of natural language processing, is concerned with how words can be put together to form sentences and how those words relate to each other. Syntactic parsing in logic programming has traditionally been a sequential process incorporating the logic programming language Prolog. However, recent research is being done in the area of parallel logic programming languages, namely PARLOG, which embodies a control strategy that is designed for parallel evaluation.

2. Syntactic Parsing

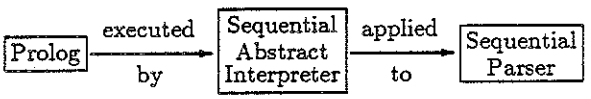
The syntactic structure of a natural language is analyzed by a parser. To examine how the syntactic structure of a sentence can be computed, two things must be considered: the **grammar**, which is the formal specification of the structures allowable in the language, and the **parsing technique**, which is the method of analyzing a sentence to determine its structure according to the grammar.

Definite clause grammars (DCG), which are natural extensions of context-free grammars, provides not only a description of a language, but also an effective means for analyzing strings of a language. Actually, DCG are executable programs of the programming language Prolog. DCG overcome some of the inadequacies encountered with context-free grammars in three ways. Firstly, DCG provide for context-dependency in a grammar so that the permissible forms of a phrase may depend on the context in which that phrase occurs in the string. Secondly, DCG allow arbitrary tree structures to be built in the course of parsing. Thirdly, DCG allow extra conditions to be included in the grammar rules. In addition, a DCG extends a context-free grammar by augmenting non-terminals with arguments.

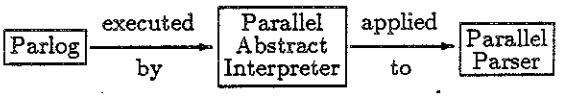
Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

3. Parallel Logic Programming

DCG fall into a class of non-deterministic sequential logic programs. In running these compiled programs under Prolog, the non-determinism is simulated by a depth-first backtracking control strategy. Recently, there has been research in the area of defining logic programming languages which take advantage of concurrency. The execution models representing sequential and parallel processing in logic programming are shown below.



Sequential Model



Parallel Model

A sequential abstract interpreter consists of a unification algorithm and depth-first searching with backward chaining. A parallel abstract interpreter embodies all the mechanisms for parallelism – concurrency, communication, synchronization, and indeterminacy.

PARLOG is characterized by guarded clauses and committed choice non-determinism as a control strategy rather than full AND/OR parallelism. It can be distinguished primarily by the way in which communication is controlled. In PARLOG, each procedure has a fixed mode of use: mode declarations state whether each argument is input or output. Processes suspend on input arguments which are not sufficiently instantiated.

4. Conclusion

This paper is a brief outline of my ongoing research intended to investigate the uses and advantages of parallelism over sequential strategies in the syntactic parsing phase of natural language processing.

Simulation of Multi-Layer Neural Network on ETA-10

Holmes S. Liao, R. C. Lacher
Department of Computer Science
Florida State University
Tallahassee, FL 32306

Keywords: Neural Network, Multi-layer Network, Back Propagation, Supercomputing

1. Introduction

In a particular back propagation learning model (Rumelhart et al. 1986), there are four layers in multi-layer neural network with every node in the same layer running the same formula. We plan to pipeline four different learning formulae and distribute the four layers in the network onto the four processors of ETA-10. We believe that by employing a multi-processor vector machine, such as ETA-10, the speed to convergence can be greatly improved because of the dominant summation computations in the learning algorithms and the widely use of matrixes in storing temporary results.

2. Proposed Simulation on ETA-10

There are two approaches to simulate the multi-layer neural network on ETA-10. We call the first one SIMD approach and the second MIMD. (The classification is borrowed from M.J. Flynn's scheme.)

In the first approach, SIMD, all the four processors of ETA-10 run the same layer of the multi-layer neural network at any given instant. Because every unit in each single layer of the multi-layer neural network runs the same formula, this implies that the four processors can possibly execute the same instruction stream at a particular instant; this is the reason why we call it SIMD. In this approach we plan to employ static load balancing scheme among processors, that is, each of the processors has a designated number of processes during initialization stage. Little but a fixed pattern of process migration or none is required if static load balancing scheme is adopted. To store wights propagating through the network a fixed amount of buffer is required. Consequently a synchronization scheme, possibly barrel synchronization, will be used to control the four processors to advance to the next layer of the neural network and to execute the next formula. The use of the synchronization scheme can prevent overflow of the buffer from happening. The execution pattern of the four processors on ETA-10 will be as follows: If there are 1,000 units in a particular layer, each processor will be assigned to 250 units. When a processor is executing a process (or say, simulating an unit), it receives messages from other units in other layers and generates another message for other units in other layers. As mentioned above, socket or communication buffer will be used to simulate the weight propagation in the neural network.

In contrast to the SIMD the second approach, MIMD, a CPU is allocated by a single layer in principle. (If 8 CPUs are installed on ETA-10, we can assign 2 CPUs for each neural layer which will make the convergence time even faster.) Since every neural layer runs different algorithm and the number of units in different layers are almost always different, the time for a CPU to finish its share of neurons will certainly be different from the other CPUs'. We plan to employ dynamic load balancing strategy to fully utilize ETA-10's CPU resources. We also plan to develop a robust buffer manager to coordinate the message passing among neurons and to control the growth and the decline of the communication buffer. We speculate that if the buffer manager is powerful enough the synchronization for different layers in the neural network can be carried out by the manager. Similar to a vector machine MIMD approach also needs a start-up time for all ETA-10's CPUs to fully activate in that the weight propagation activity is done in a serial back-and-forth fashion. This start-up time is not necessary in SIMD approach.

3. Conclusion

It is not clear whether SIMD or MIMD is better in terms of the system performance. One of our objectives is to evaluate the two approaches and compared with previous experiments implemented on serial machines. Computation speedup of massively parallel architecture is expected to be in constant folding (Fahlman and Hinton 1987). Though we don't have unlimited hardware on ETA-10, it is believed that the constant-folding speedup can be achieved to a certain extent if we increases the number of processes (units) in the neural network. Another objective of this project is to find the curve of computational speedup in this particular simulation. All these results shall be reported once they are known.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

A STUDY ON THE APPLICATION OF A NEW FUZZY REASONING METHOD

Alex H. Meng, Zhiqiang Cao and Abraham Kandel

SUS Center for Artificial Intelligence and
The Department of Computer Science
Florida State University
Tallahassee, Florida 32306-4019, U.S.A.

Fuzzy reasoning, proposed and developed by Zadeh[6] from his fuzzy sets theory[5], and its application of modelling imprecise systems have been studied by Zadeh himself and many other workers[2,3]. However, an alternative approach to fuzzy reasoning is suggested by Cao, Kandel and Li[1] recently. The new method deals with the same form of fuzzy reasoning as in Zadeh[6], but it differs in the following aspects:

(1) A new fuzzy concept is employed to replace the precise concept of a variable having a value. This new fuzzy concept is described by the set of grades of membership of the value to the fuzzy subsets that are used to cover all the possible values of that variable.

(2) The fuzzy relation matrix is formed directly from the verbal description, and it represents all the possible fuzzy relationships between the two sets of fuzzy subsets in the universes of the input and output vasriables of a system.

(3) In the inference process, the new method uses the fuzzy relation matrix which acts like a linear transformation and converts the new fuzzy concept of the input variable having an input value into the new fuzzy concept of the output variable having the inferred value.

In this work, we apply the new fuzzy reasoning method to construct a fuzzy model of a d.c. (direct current) series motor. That motor is originally used by Kiszka, Kochanska and Sliwinska[4]. Using Zadeh's fuzzy reasoning scheme[6], they build a fuzzy model of the series motor to investigate the effects of implication operators. We choose to use their motor because we, then, are able to compare the two reasoning methods by the construction procedures and the results of the two models under the same conditions. In addition, we aim to obtain a better understanding about the new reasoning method and provide a better use of it by analyzing the model from a practical prospect. Thus, a simple heuristic algorithm which can modify the fuzzy relation matrix based on the feedback from the system is developed in the model. Using the modified fuzzy relation matrix, our model can produce more accurate results with respect to the feedback.

Our studies suggest that the new fuzzy reasoning method carries following qualities:

(1) Comparing the two model results, we assert it is at least of the same effectiveness.

(2) It is more proper to apply to a system whose domain is a continuum because the universe of the domain need not be quantized.

(3) It excludes the implication and union operations that are needed in Zadeh's reasoning scheme[6] to form the fuzzy relation matrix from the verbal description.

(4) A fairly effective heuristic algorithm which modifies the fuzzy relation matrix based on the feedback of the system to improve the model performance is easy to develop.

(5) The problem of finding a good inference operator becomes trivial because we can always alter the fuzzy relation matrix by either using the heuristic algorithm in the model or interpreting the fuzzy value differently to accommodate different operators. However, when another inference operator is used, that algorithm must be redesigned. Luckily, it is easy to be developed.

(6) The size of fuzzy relation matrix in the new reasoning method is smaller. Therefore, the computaion is both less space and time consuming in our model.

(7) It doesn't need a custom-made definition for the fuzzy subsets to produce a good results for a particular system. In other word, it provides the generality. Our model have shown it can be used to model different systems with only the verbal description and the range of the domain of a system being updated.

References

[1] Cao, Z., Kandel, A. and Li, L., A new model of fuzzy reasoning, submitted to *fuzzy sets and systems*.

[2] Kandel, A., Fuzzy Techniques in Pattern Recognition, John Wiley and Sons, N.Y., 1982.

[3] Kandel, A., Fuzzy Methematical Techniques with Applications, Addison-Wesley, MA., 1986.

[4] Kiszka, J.b., Kochanska, M.E. and Sliwinska, D.S., The influence of some fuzzy implication operators on the accuracy of a fuzzy model - Part I, *Fuzzy Sets and Systems* 15, pp. 111-128, 1985.

[5] Zadeh, L.A., From circuit theory to system theory, *Proc. Institute of Radio Engineers* 50, pp. 856-865, 1962.

[6] Zadeh, L.A., Outline of a new approach to the analysis of complex systems and decision processes, *IEEE Trans. Systems Man Cybernet* 3, pp. 28-44, 1973.

Acknowledgements

This work is supported in part by NSF grant ISI 8405953 and by the Florida High Technology and Industry Council grant UPN 85100316.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

INTERFACING AND USING AI EXPERT SYSTEM SHELLS WITH LASER-OPTICAL TECHNOLOGY

Dr. James M. Ragusa
Associate Professor
University of Central Florida
Orlando, Florida, (407) 275-2915

Ann E. Barron
Martin Marietta Electronics & Missiles Group

Michael Vine
Harris Government Information Systems Division

The purpose of this paper is to describe preliminary results of a ten-month project whose primary objective was to expand knowledge and applications potential made possible by the synergistic interfacing and use of two relatively new technologies -- artificial intelligence (AI) expert system shells and laser-optical devices. Presently these technologies exist separately with little, if any, effort being made by either researchers or commercial firms to couple them for application purposes. Independently these technologies are just starting to be used to meet a variety of beneficial needs, and implementation of these methods in real-time operational hardware is just beginning.

Potential productivity enhancement of human performance through the use of these coupled technologies for a broad range of technical and commercial applications should be obvious. A number of applications are identified in the paper. In addition, a demonstration expert system interfaced with a Laserdisc is described. Present study results indicate that there are no significant restrictions which prevent the full integration of these technologies.

This paper has been primarily scoped to the integration of PC-based expert systems (with two exceptions) and laser-optical technology currently available. Ten commercial expert system shells were identified for in-depth analysis during the study. They are: Personal Consultant Easy and Personal Consultant Plus (Texas Instruments), NEXPERT Object (Neuron Data), EXSYS (EXSYS Inc.), 1st-Class (Programs in Motion), GURU (Micro Data Base Systems Inc.), VP-Expert (Paparback Software Inc.), GoldWorks (Gold Hill Computers), TIMM (General Research), and LEVEL5 (Information Builders Inc.).

In addition, two large hybrid tools were included for completeness. They were ART (Inference Corp.) and KEE (IntelliCorp). These tools were considered because they are widely recognized as signifcant mini/workstation development systems. With their inclusion the full range of expert system capabilities (frames, semantic nets, object oriented programming, blackboard architectures, dynamic inheritance, etc.) will have been studied. Laser-optical systems which have been investigated for interface compatibility are: Compact Disc-Read Only Memory (CD-ROMs), Write-Once-Read-Many (WORM), Laserdiscs, Lasercards, and Laserfilm.

The Laserdisc player selected for demonstration purposes is the Pioneer LD-V4200 LaserVision system used with a "repurposed" laserdisc which had been developed earlier for another application. Texas Instruments' expert systems shell software product, Personal Consultant Plus, was used in the project for demonstration system development because of an existing university site license. Other database, graphics, and spreadsheet computer software and language programs already available were used as required for expanded compatibility purposes.

To date, an ever expanding segment of users is beginning to recognize the advantages of laser optical technology for rapid retrieval, low-cost archival storage, and displays of high quality color images with sound. This technology has moved from one with sales of less than $500,000 per year to one with multi-million dollar potential. A similar expansion is presently developing with expert systems shell technology. By integrating these technologies, it will be possible to provide intelligent and user-friendly interfaces to text, graphics, high resolution color stills, motion sequences, and high quality sound -- either separately or in combination, as required by the application.

Identified applications interest ranges widely for both technologies. Included are such diverse fields as aerospace and defense, business (including a large base of financial services), education, training and simulation, arts and sciences, and entertainment. Specifically, the marriage of these technologies is presently of interest to DoD organizations for simulation, training, and logistics purposes. In addition, NASA has expressed strong interest in applications for current and future space programs. Discussions have already begun with NASA-Kennedy Space Center and NASA-contractors (Lockheed Space Operations, Grumman Technical Services, and McDonnell Douglas Astronautics) and several defense corporations (Martin Marietta Electronics & Missiles Group and Harris Government Systems). Each of these organizations has recognized the promise and advantages of linking these technologies. Possibilities for an even wider range of applications offered by this coupling are limited only by the imagination of those in need. This technology transfer-based research has been made possible by a grant (IST 20-21-059) from the University of Central Florida's Institute for Simulation and Training. Project personnel consist of the Principal Investigator and graduate students from the Colleges of Business and Education. Fortunately, these students have advanced technical skills needed to make this project feasible.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

Problem-Solving Paradigms and Requirements Analysis

S. Smith, East Tennessee State Univ
S. Stoecklin, Florida A & M Univ
A. Kandel, Florida State Univ

The relationship between artificial intelligence and software engineering is a concern among computer scientists. This article deals with software engineering's contribution to artificial intelligence. The premise presented here is that software engineering with the heuristics of the requirements analysis can aid in the application of problem-solving paradigms to AI systems.

Requirements analysis applied to an AI system describes the following:

1. information domain,
2. relationships between information,
3. control domain,
4. decision-making properties, and
5. functions.

This knowledge gathered by the application of requirements analysis can be used in developing AI systems. First of all, this knowledge acts as a guide in the determination of which problem-solving paradigm to use in the AI system. With some AI problems, the paradigm to use is quite evident; for example, diagnosis systems use a rule-based paradigm. However, there are systems for which the problem-solving paradigm is not evident.

Secondly, this knowledge provides the information which is needed to implement the problem-solving paradigm. The data in the knowledge base for the AI system, the "artificial" intelligence that the system requires, and an abstract view of those functions required to implement the system are part of the information delineated in this knowledge.

The benefits of the knowledge acquired from requirements analysis can be seen in three problem-solving paradigms. Requirements analysis delineates in the definition of the information and decision-making domains the equivalence classes for test cases to be generated for the generate and test problem-solving paradigm. In the rule-based paradigm, the identification of problem partitioning early in development can yield a more effective method of developing rules and a more efficient execution rate of these rules. The use of requirements analysis with the object-oriented paradigm provides the means of concisely defining the interfaces necessary for communication between objects.

The use of software engineering methodologies has shown significant improvement in the reliability, deliverability, and maintenance of software systems. In the future, when artificial intelligence systems will be larger and more complex, these benefits will then be seen in the development of artificial intelligence systems. The use of requirements analysis in developing AI systems is a first step in providing a way for software engineering to contribute to artificial intelligence.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

A HEURISTIC APPROACH SIMULATION TOOL FOR BEST PATH GENERATION FOR ROBOTIC APPLICATIONS

R. SURESWARAN, A. KATBAB

R. RAVINDRAN & R. ALONSO.

UNIVERSITY OF MIAMI

ELECTRICAL & COMPUTER ENGINEERING DEPARTMENT

P.O. BOX 248294

CORAL GABLES

FL 33124

KEY WORDS : Heuristic, Simulation, Robotics, Best Path Generation.

ABSTRACT

Automated robotic movement is a major field of research in robotics. Automated robotic movement itself consists not only of moving the entire robotic module from one place to another or moving individual parts of the robot from point to point, but also finding the most efficient path of movement.

This paper discusses the simulation of a robot arm with three degrees of freedom. The aim is not just to move the tip of the arm from one point to another, but to also find the most efficient path of movement. The shorter the time it takes to reach its destination, the more efficient that particular path is. The path chosen should also take into consideration any obstructive restriction resulting from the structure of the arm and environment. This most efficient path is calculated with the help of A.I. programming using Heuristics.

The simulation displays a 3D graphic image of the robot arm and the path it moves from its current position to its destination. It displays the actual movement of each link in accurate detail of relative size and speed with respect to one another. The input may be of any form; that is, either in cartesian coordinates relative to the robot arm or by angles of the joint. The coordinates or joint angles of the robot arm during the course of motion are displayed together with the total relative time. The user has the option of whether to continue the movement from any point that he had already reached, or to restart the movement from home position.

The A.I. algorithm also takes into consideration all joint restrictions imposed upon the robot arm. Any point which is beyond the reach of the arm or is impossible to reach due to structural and environment restrictions will be clearly displayed by the simulation.

The simulation software, together with a few additions and some communication hardware, can be used to control the robot arm directly once the user is satisfied with the simulation results.

The above described simulation software tool has been successfully implemented and tested. It is compatible to simulate any robot with two links and a rotary base. The length of the links and the speed of the joints are all constants which can easily be assigned new values. It can also be expanded to handle robots with more freedoms of movement as the algorithm used is very modular and the same heuristics can be applied.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

Plan Generation with Time Constraints

Bharadwaj S. Tirumala and Lawrence O. Hall
Department of Computer Science and Engineering
University of South Florida
Tampa, Fl. 33620

Temporal reasoning, which is a way of pursuing goals and drawing inferences based on events occurring over time, plays an important role in automated planning systems and in general in common sense reasoning. This work is an attempt at exploring the problems involved in reasoning over time which typically involve updating a plan structure with changing world patterns. This involves developing the appropriate knowledge representation in addition to a plan generation system.

A deductive retrieval mechanism, which has been tailored to the needs of temporal retrievals, has been implemented. Uncertainty due to incomplete information and indecision is resolved using fuzzy values and a dynamic resolution over a temporal data base. Imprecise temporal information is captured in fuzzy intervals. These intervals are made up of a beginning hour and ending hour. The system can find the tightest possible bounds on a possible event or step in a plan. The system user provides the constraint information for plan development. This is combined with basic domain information in the knowledge base. A plan or set of steps through some temporal constraints will be presented based upon the constraints and domain information. A fuzzy belief in the chance of the plans' success is associated with the information provided by the system.

The deductive mechanism is made flexible with a combination of the techniques of forward and backward chaining. Current goals cause information to be searched for in a backward chained manner. As information is added to the systems working memory it may trigger data-driven rules (or forward chaining rules) which will derive further information and add it to the working memory.

The truth maintenance system which incorporates a data dependency network is also capable of default reasoning in the temporal context. When information that was believed becomes no longer tractable, the nodes which depend upon it being true will be examined and their belief toggled appropriately. This process will continue until no more information needs to be changed.

The system is capable of generating simple, alternative plans for a set of user specified goals with appropriate belief values and options associated with each feasible plan. Upon encountering a choice point the system will ask the user for advice in pursuing alternatives. This interaction can be used to prevent the combinatorial explosion that will occur with many feasible choices in a plan. It is also possible for the user to specify in advance which possibilities should be pursued upon coming to the point of having divergent possibilities.

This work has been an effort at providing a usable basis for planning systems which deal with time. A consistent strategy has been used in dealing with the incomplete and potentially defeasible temporal information that is available in realistic situations. The system developed is capable of planning, as a pattern of temporally organized intended action descriptions. It can be described as a combination of the linear and the hierarchical planning methodologies. It is capable of *refinement*, making choices in the course of a plan formulation and, *validation*, in the sense of determining if the commitments made would lead to a solution.

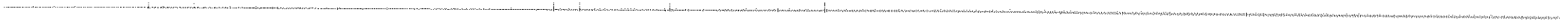
The system is being developed to generate plans for the set up of multiple actor game scenarios, which may be played out. It is being used to generate plans for airplanes must negotiate hostile air space. In this domain the system is quite robust in the relatively simple examples that have been tried. The system has in general provided a strong basis for the development of both applications.

Acknowledgement: This research was partially supported by the Florida High Technology and Industry Council's Simulation and training division.

Proceedings of the 2nd Florida Artificial Intelligence Research Symposium, 1989, by The Florida AI Research Society.

List of Authors

INDEX




flairs
florida artificial intelligence research symposium