Relating Preference Languages By Their Expressive Power

Michael Huelsman
Saint Anselm College
100 St Anselm Drive
Manchester, NH 03102

Miroslaw Truszczynski
University of Kentucky
Lexington, KY 40506

Abstract

There has been a great deal of research into methods for representing preferences, called preference representation languages. Often, research in this area deals with a limited number of similar languages, in isolation. This work establishes a new method of analyzing the similarity of different languages by considering the class of preference orders that each language is capable of expressing. Our method involves the definition of a relation called preference representation language subsumption, which allows us to relate various languages by their expressive power. We demonstrate several general proof techniques for showing that such a relation exists or does not exist. Additionally, we provide a small case study for several languages that express preferences over combinatorial domains and discuss several analytical uses for the proposed subsumption relation.

Introduction

Reasoning about an agent's preferences requires the use of a formal representation. Hence, methods and languages to represent preferences have been studied extensively (Kaci 2011; Fishburn 1974; Von Neumann and Morgenstern 1944; Booth et al. 2010; Boutilier et al. 2004). In most cases, these studies focused on a single representation language without any significant consideration given to how different languages are related. We propose a method of relating preference representation languages, hereafter languages, based on their expressive power.

This relation exists if a language can express any preference relation another language can express. Organizing preference languages according to this relation, which we call subsumption, we can determine their expressive power wrt other languages. Applying this to a set of languages sharing common elements can be used to determine which language elements increase expressive power and which do not.

This paper is divided into five sections. The next section provides a background on several preference representation languages as well as formal definitions for subsumption, and related concepts. Then, we introduce several techniques which are useful for proving subsumption. After that, as a case study, we apply subsumption to a set of 12 languages defined over combinatorial domains. Finally, we discuss the uses and drawbacks of subsumption before ending on some concluding thoughts and open questions.

Background

Consider an agent that can, given two alternatives α and β, say whether α is at least as preferred as β. The agent's *weak preference relation*, $\succeq$, is the collection of such preferences over some domain of alternatives Ω. We assume that $\succeq$ is a preorder, i.e. it is transitive and reflexive. Some related relations can be derived from this weak relation. First, we set $\alpha \succ \beta$, i.e. α is strictly preferred to β, if $\alpha \succeq \beta$ and $\beta \not\succeq \alpha$. Additionally, *preferential equivalence* or *indifference*, denoted $\approx$, is defined by setting $\alpha \approx \beta$ if $\alpha \succeq \beta$ and $\beta \succeq \alpha$. Finally, *incomparability*, denoted $\bowtie$, holds when $\alpha \not\succeq \beta$ and $\beta \not\succeq \alpha$.

A preorder also defines an *order* (a preorder with no equally preferred alternatives) over the equivalence classes of the associated relation $\approx$; For two different equivalence classes (sets of alternatives which are preferentially equivalent) $\boldsymbol{\alpha}, \boldsymbol{\beta} \subseteq \Omega$, we define the order $\succ$ by setting $\boldsymbol{\alpha} \succ \boldsymbol{\beta}$ if there exists $\alpha \in \boldsymbol{\alpha}$ and $\beta \in \boldsymbol{\beta}$ such that $\alpha \succeq \beta$. For example if $\Omega = \{a, b, c\}$ and we have $a \approx b$ and $b \succeq c$ (ignoring assumed reflexive comparisons) then the derived order is $\boldsymbol{\alpha} \succ \boldsymbol{\beta}$, where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the equivalence classes defined by $\approx$ with $\boldsymbol{\alpha} = \{a, b\}$ and $\boldsymbol{\beta} = \{c\}$.

There exist a few pertinent properties of preorders. First, we define one preorder $\succeq'$ as *consistent* with another preorder $\succeq$ if for all pairs $(\alpha, \beta) \in \Omega$, $\alpha \succeq \beta$ implies $\alpha \succeq' \beta$. Second, a preorder is a *total* preorder if for any two different alternatives α and β either $\alpha \succeq \beta$ or $\beta \succeq \alpha$.

Domains of alternatives are often *combinatorial domains*. Formally, a combinatorial domain is defined by a set of *attributes*, $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$, and their domains $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$. Each domain is assumed to be finite and contains more than one value. The shorthand $\mathcal{D}(v_i)$ denotes the domain of attribute v_i. An alternative α of a combinatorial domain is represented by a tuple such that: $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathcal{D}(v_1) \times \mathcal{D}(v_2) \times \ldots \times \mathcal{D}(v_n)$.

Clearly, combinatorial domains are exponential in size wrt the number of attributes. We present several languages commonly used for defining preferences over combinatorial domains. We assume that all preferences are expressed over a binary combinatorial domain, that is, a combinatorial domain whose attributes can have only one of two values

$\{0,1\}$. We refer to a binary combinatorial domain of a set of attributes $\mathcal{V}$ using the symbol $\mathcal{C}(\mathcal{V})$. By restricting to binary combinatorial domains we simplify many representations, but maintain generality, since any multi-valued combinatorial domain can be expressed as a binary one.

Later we will use binary combinatorial domains to construct a language of propositional logic $\mathcal{L}(\mathcal{V})$ where each alternative $\alpha \in \mathcal{C}(\mathcal{V})$ represents an interpretation such that if $\alpha[i] = 1$, for some attribute i, then the Boolean variable i is also true in the interpretation derived from α. This allows for the construction of logical formulas which describe properties of alternatives. In what follows we often take advantage of the following observation: For every alternative $\alpha \in \mathcal{C}(\mathcal{V})$, there is a formula $\phi_\alpha \in \mathcal{L}(\mathcal{V})$ such that ϕ_α is only satisfied by a given alternative, α.

Ranking preference formulas (RPFs) form a language defined over a binary combinatorial domain $\mathcal{C}(\mathcal{V})$. This language consists of vectors of propositional formulas $\boldsymbol{\varphi} = (\varphi_1, \varphi_2, \ldots, \varphi_k)$, where $\varphi_i \in \mathcal{L}(\mathcal{V})$, for $i = 1, \ldots, k$. Here $\mathcal{L}(\mathcal{V})$ denotes the language of propositional logic defined over the set $\mathcal{V}$ of attributes of the domain $\mathcal{C}(\mathcal{V})$. We define the *rank*, $r(\alpha)$, of an alternative $\alpha \in \mathcal{C}(\mathcal{V})$, by setting $r(\alpha) = \min\{i : \alpha \models \varphi_i\}$, where $\alpha \models \varphi_i$ means the truth assignment induced by α satisfies φ_i using the normal definition of logical satisfaction. Given two alternatives $\alpha, \beta \in \mathcal{C}(\mathcal{V})$, we define $\alpha \succeq_{\boldsymbol{\varphi}} \beta$ if $r(\alpha) \leq r(\beta)$. We note that if $\{i : \alpha \models \varphi_i\} = \emptyset$, $r(\alpha) = \infty$. Thus, alternatives which do not satisfy any formula in $\boldsymbol{\varphi}$ are less preferred than any other alternative. RPFs generate total preorders.

A *preference theory* (PT) P is a set of RPFs. For each RPF in P an alternative α is assigned a rank, or satisfaction value. These satisfaction values are collected into a *satisfaction vector* $S(\alpha)$ where each entry in the satisfaction vector is associated with a specific RPF (according to some a priori order of the RPFs in P), thus $S(\alpha)_1$ refers to the satisfaction value for the first RPF and so on. Given a PT P and two alternatives α, β we define the preference relation determined by P, written $\succeq_P$, by setting $\alpha \succeq_P \beta$, if for every RPF $r \in P$, $\text{rank}_r(\alpha) \leq \text{rank}_r(\beta)$. PTs are related to the ASO preference language (Brewka, Niemelä, and Truszczynski 2003). The original ASO formalism by Brewka, Niemelä, and Truszczynski allows for the specification of hard constraints on the domain, while PTs do not.

One may extend the PT language by assigning each RPF a Boolean formula over $\mathcal{L}(\mathcal{V})$ as a condition. Given preference theory P containing RPF r which is assigned condition c the satisfaction value for an alternative α of r is equal to $\text{rank}_r(\alpha)$ if $\alpha \models c$. Otherwise, the satisfaction value is ∞. Another extension is assigning ranks to the individual RPFs. Given a ranked preference theory (RPT) P, the preference relation between α and β is determined by comparing the alternatives in order of the formulas ranks (with lower ranks being more important.) Any result of this comparison other than equivalence is taken as the result of the comparison under the entire PT. If the alternatives are equivalent wrt all RPFs in P, they are equally preferred. Conditional and ranked RPFs are extensions to the PT language which represent common ways of thinking about preferences, i.e. some properties are more important than others and certain preferences only apply when considering certain alternatives.

An important concept underlying several preference representation formalisms is that of lexicographic ordering. A *lexicographic preference model* (LPM) represents preferences over a combinatorial domain by considering the attributes in order of importance with a preference order for each attribute domain (Fishburn 1974). In an LPM, an alternative α is preferred to another alternative β if α has a more preferred value on the most important attribute where it differs from β.

Lexicographic preference trees (LP-trees) are an extension of LPMs (Booth et al. 2010). An LP-tree uses a tree structure to represent the importance order of attributes. Each non-leaf vertex in the tree is labeled with a singular attribute v_i and has a number of descendants equal to the size of its domain, $|\mathcal{D}(v_i)|$. LP-trees have the additional condition that each path from the root to a leaf has exactly one occurrence of each attribute as a label. Like LPMs each vertex has an associated linear order over the domain of its attribute, different vertices with the same associated attribute may also have different linear orders. The process of determining dominance between two alternatives is similar to an LPM where one traverses the tree until the two alternatives differ. Variants of LP-trees are described as to whether or not they have conditional importance (CI, UI) or conditional preferences (CP, UP).

Finally, this work studies the following languages in addition to those above: penalty logic (Haddawy and Hanks 1992), CP-nets (Boutilier et al. 2004), CLPMs (Huelsman and Truszczyński 2020), utility functions (Von Neumann and Morgenstern 1944), weighted average models, and ranking functions. Due to space restrictions we do not formally define these languages here.

Preference languages consist of a syntax and a semantics. Each particular syntactically valid combination of preference statements defines a preference expression of that language. Interpreting a preference expression based on the language semantics gives us a preference ordering over a specific domain (i.e. the semantics provide a mapping from a preference expression M to a preference relation $\succeq_M$). We say that $M \models \alpha \succeq \beta$ if $\alpha \succeq_M \beta$. Since preference expressions directly imply a preference ordering we will use preference expressions as a shorthand to mean a preference ordering. Subsumption relies on equivalence between preference expressions, which we define below:

Definition 1 (Preference Expression Equivalence)**.** *Given two preference expressions M and M' selected from some two languages over the same domain of alternatives Ω, M is equivalent to M', $M' \equiv M$, if for all pairs $\alpha, \beta \in \Omega$, $M \models \alpha \succeq \beta$ if and only if $M' \models \alpha \succeq \beta$.*

Using this definition of preference expression equivalence we define our notion of subsumption as follows:

Definition 2 (Preference Language Subsumption)**.** *Given two preference languages L and L' and a domain Ω, L subsumes L' over Ω, $L' \sqsubseteq_\Omega L$, if for every preference expression $M' \in L'$ over Ω there exists a preference expression $M \in L$ over Ω such that $M' \equiv M$.*

We extend this definition to a more general case over a class of domains $\mathbf{\Omega}$ such that $L \sqsubseteq L'$ over $\mathbf{\Omega}$ if $L \sqsubseteq_\Omega L'$ for each $\Omega \in \mathbf{\Omega}$. This more general subsumption, which we will use going forward, means that subsumption exists when there exists a mapping from preference expressions of one language to preference expressions of another (for a given class of domains). If a polynomial time mapping can be constructed then we call the relation *efficient*.

Definition 3 (Efficient Preference Language Subsumption)**.** *Given two preference languages L and L' and a domain Ω, L efficiently subsumes L' over Ω, $L' \sqsubseteq_\Omega L$, if there exists a mapping function $f : L' \to L$ such that for any $M \in L'$ over Ω, $f(M) \equiv M$, and $f(M)$ can be computed in polynomial time.*

Efficient subsumption provides insight into the computational complexity of preference reasoning problems in different languages. For example, if L efficiently subsumes L' and dominance is NP-hard for L' then it is also NP-hard for L since we can construct a polynomial time mapping reduction from the dominance problem in L' to the dominance problem in L. In this work, all proofs of subsumption are not proofs of efficient subsumption, unless otherwise stated. Subsumption also defines a *language equivalence* relation, stated below.

Definition 4 (Preference Language Equivalence)**.** *Given two preference languages L and L' over a domain Ω, L is equivalent to L', $L \sim_\Omega L'$ if $L \sqsubseteq_\Omega L'$ and $L' \sqsubseteq_\Omega L$.*

These definitions provide a theoretical basis for the use of subsumption. Later we apply these definition to languages over the class of combinatorial domains.

Proof Techniques

The core strategy for proving the existence of a subsumption relation is to build a mapping from preference expressions of one language to equivalent preference expressions of another. Proving a lack of subsumption between two languages can be more difficult. The most direct method is to show that one of the languages can express a specific preference ordering the other cannot.

An easy way to demonstrate that one language can induce a preference ordering that another cannot is to show that a relation derived from a preference expression of the language's weak preference relation ($\approx$, $\succ$, or $\bowtie$) cannot be derived by any preference expression of another language. For example, if preference expressions of one language can have incomparable alternatives then that language cannot be subsumed by a language whose preference expressions cannot contain incomparable alternatives. We formally state this below in Extension 1.

Extension of Subsumption 1. *Given two languages $\mathcal{L}$ and $\mathcal{L}'$ and a preference relation $R \in \{\succ, \approx, \bowtie\}$ if there exists a preference expression $M \in \mathcal{L}$ over a domain Ω such that $\alpha R_M \beta$ for two alternatives $\alpha, \beta \in \Omega$ and there exists no preference expression $M' \in \mathcal{L}'$ such that $\alpha' R_{M'} \beta'$ for two alternatives $\alpha, \beta \in \Omega$ then $\mathcal{L} \not\sqsubseteq \mathcal{L}'$.*

Some properties of languages narrow down the class of preorders that can be induced by their preference expressions and can assist in establishing subsumption. A simple example of this is that every preorder defined by a utility or ranking function is a total preorder (Extension 2).

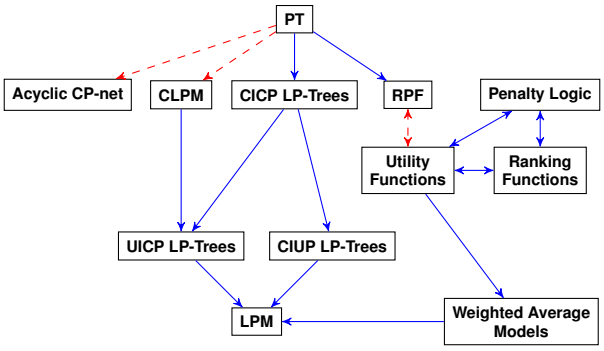

Figure 1: Each edge (u, v) means that preference language u subsumes v. Solid edges are known from the literature.

Extension of Subsumption 2. *If every preference expression M in a preference language defines its preorder by means of some utility function f_M and one of the relations $\geq$ or $\leq$ on reals (so that we have $M \models \alpha \succeq \beta$ if and only if $f_M(\alpha) \leq f_M(\beta)$, or in the other case $f_M(\alpha) \geq f_M(\beta)$), then every preference order defined by a preference expression from that language is a total preorder.*

While these results are trivial they simplify the process of establishing subsumption for some collections of languages. When combined with the transitivity of the subsumption relation these extensions show that it is not always necessary to compare each pair of languages in a set, but instead focus on a smaller subset of pairs.

Case Study: Combinatorial Domains

As a case study of subsumption we have created the graph in Figure 1. Each vertex in the graph represents a language with an edge between two vertices indicating subsumption, the origin vertex subsumes the destination vertex. If no edge is present between two vertices, and there is no path between them, the languages are incomparable (i.e. they are unrelated according to subsumption.)

Figure 1 represents both information from the literature as well as our own work. Our work consists of showing that subsumption exists between RPF and utility functions, PTs and CLPMs, and PTs and Acyclic CP-nets, all other relations can be found in the literature (Booth et al. 2010; Brewka, Niemelä, and Truszczynski 2003; Huelsman and Truszczyński 2020; Kohli and Jedidi 2007; Zhu 2016). Additionally, we proved that where one language is unreachable from another that no subsumption relation exists, thus the graph has as many non-loop edges as possible, ignoring those implied by transitivity.

Among those subsumption relations which are shown by the literature we note that in the case of CICP LP-trees subsuming CIUP LP-trees and UICP LP-trees, CIUP LP-trees and UICP LP-trees subsuming LPMs, and utility functions subsuming weighted averages these subsumption relations

are efficient due to being expressions of the same general structure with various different restrictions. The subsumption of LPMs by weighted averages is also efficient, but requires a non-trivial conversion (Fishburn 1974).

Towards proving the relations in Figure 1 we look at a cluster of languages (RPFs, ranking functions, penalty logic, and utility functions) which are equivalent to each other. This proof shows that a ranking function can replicate any given total preorder over a finite domain (Lemma 1.)

Lemma 1. *Given a total preorder $\succeq$ over a finite domain Ω there exists a ranking function r such that the preference relation induced by r, $\succeq_r$, is equivalent to $\succeq$.*

Proof. Sketch. Sort all alternatives according to $\succeq$ assign the most preferred alternatives rank 1, in the next most preferred rank 2, and so on. □

This result shows that ranking functions subsume RPFs, given that both produce total preorders. Moreover, one can also show that RPFs subsume ranking functions (Lemma 2) making them equivalent.

Lemma 2. *Ranking Function $\sqsubseteq$ Ranking Preference Formula*

Proof. Consider an arbitrary ranking function r over a domain $\mathcal{C}(\mathcal{V})$. For each rank $k \in r$ we define a Boolean formula Φ_k such that

$$\Phi_k = \vee_{\alpha \in \mathcal{C}(\mathcal{V}), r(\alpha)=k} \phi_\alpha.$$

We then define an RPF $\boldsymbol{\varphi} = (\Phi_1, \ldots, \Phi_n)$ where n is the highest rank according to r. This means for a particular alternative α its rank according to $\boldsymbol{\varphi}$ is the same as its rank according to r, thus they induce the same preference relation. Therefore, all preference expressions in the Ranking Function language have an equivalent preference expression in the RPF language. □

Using this proof, among other results, we show that the cluster containing ranking functions, utility functions, penalty logic, and RPFs are all equivalent languages, since they induce total preorders and also subsume ranking functions. This cluster represents a set of languages capable of expressing any total preorder over a combinatorial domain. Similar to how ranking functions include any total preorder, PTs include any preorder. This means that PTs subsume all the languages we consider here. We prove this result, stated later in the paper as Proposition 1, in several steps. First, we show that if there exists a set of satisfaction vectors associated with a set of alternatives then there exists a preference theory P which produces those satisfaction vectors for those alternatives, see Lemma 3.

Lemma 3. *Given a set of alternatives $A = \{\alpha_1, \ldots, \alpha_n\}$ from a combinatorial domain $\mathcal{C}(\mathcal{V})$ and a set of satisfaction vectors $S = \{s(\alpha_1), \ldots, s(\alpha_n)\}$ there exists a preference theory P which recreates S given A.*

Proof. Sketch. An RPF can replicate any ranking functions and a singular satisfaction value represent a ranking function of the alternatives in the domain. For each entry i in the satisfaction vector we create an RPF such that the RPF recreates the ranking function implied by the satisfaction vectors in S, thus recreating (in aggregate) S. □

Lemma 3 shows that constructing a set of satisfaction vectors is equivalent to showing that a PT preference expression exists. In order to construct such a set for any arbitrary preorder we first define a canonical ranking for that arbitrary preorder. The canonical ranking converts a preorder into a total preorder by adding relations between pairs of alternatives that are incomparable in the original preorder.

Definition 5 (Canonical Ranking Function)**.** *The canonical ranking function of a given preference relation $\succeq$, $r_{c,\succeq}$ is defined as follows:*

$$r_{c,\succeq}(\alpha) = \begin{cases} 1 & |Dom(\alpha)| = 0 \\ 1 + \max_{\beta \in Dom(\alpha)} r_{c,\succeq}(\beta) & |Dom(\alpha)| > 0, \end{cases}$$

where $Dom(\alpha)$ is the set of all alternatives in $\beta \in \Omega$ such that $\beta \succ \alpha$.

Given a canonical preorder (using the canonical ranking function) we can extract some useful information about the original preference relation between pairs of alternatives which will help when it comes to constructing satisfaction vectors, see Lemmas 4 and 5.

Lemma 4. *Given a partial preorder $\succeq$ if $r_{c,\succeq}(\alpha) = r_{c,\succeq}(\beta)$ then either $\alpha \approx \beta$ or $\alpha \bowtie \beta$.*

Proof. Suppose not. Since the cases are symmetrical we will handle the case where $\alpha \succ \beta$ and $r_{c,\succeq}(\alpha) = r_{c,\succeq}(\beta)$. In this case we know, by Definition 5, that either α and β are undominated, which cannot be the case since $\alpha \succ \beta$, or $r_{c,\succeq}(\beta) - 1 = \max_{\gamma \in Dom(\beta)} r_{c,\succeq}(\gamma)$. This cannot be the case because α is in $Dom(\beta)$, which means that $r_{c,\succeq}(\alpha) = \max_{\gamma \in Dom(\beta)} r_{c,\succeq}(\gamma)$. This is a contradiction. □

Lemma 5. *Given a partial preorder $\succeq$ if $\alpha \bowtie \beta$ and there exists a γ such that $r_{c,\succeq}(\alpha) > r_{c,\succeq}(\gamma) > r_{c,\succeq}(\beta)$ then $\alpha \bowtie \gamma$, $\beta \bowtie \gamma$, or both hold.*

Proof. If neither $\alpha \bowtie \gamma$ or $\beta \bowtie \gamma$ hold then a relation exists between α and γ and between β and γ. Since we know that $r_{c,\succeq}(\alpha) > r_{c,\succeq}(\gamma) > r_{c,\succeq}(\beta)$ these relations must be $\alpha \succ \gamma$ and $\gamma \succ \beta$. This means that, by transitivity, $\alpha \succ \beta$ which cannot be the case since $\alpha \bowtie \beta$, a contradiction. □

Using the canonical ranking function and Lemmas 3, 4, and 5 we are able to prove that for any given preorder there is a preference theory which induces it.

Proposition 1. *Given a preorder $\succeq$ over a combinatorial domain $\mathcal{C}(\mathcal{V})$ it is possible to construct a preference theory P such that for any pair of alternatives $\alpha, \beta \in \mathcal{C}(\mathcal{V})$, $\alpha \succeq_P \beta$ if and only if $\alpha \succeq \beta$.*

Proof. By Lemma 3 we know that if we can build a set of satisfaction vectors S, with one entry for each alternative in $\mathcal{C}(\mathcal{V})$, such that $s(\alpha) \geq_{\text{Pareto}} s(\beta)$ if $\alpha \succeq \beta$ then there exists a preference theory which induces $\succeq$ as its associated preorder. For our purposes we will consider the alternatives in $\mathcal{C}(\mathcal{V})$ to be split into clusters by the equivalence classes

defined by $\approx$ derived from $\succeq$. When we refer to alternatives below we really mean an equivalence class of alternatives.

We begin our construction by setting the first element of each satisfaction vector to the value of the canonical ranking function, $r_{c,\succeq}$, for that alternative. This means that, just considering the first element, all dominance relations in $\succeq$ are satisfied. This does not satisfy incomparable alternative relations, but as long as inserting incomparabilities does not remove these dominance satisfying properties we only need to worry about pairs of alternative (α, β) such that $\alpha \bowtie \beta$.

For each incomparable pair (α, β) induced by $\succeq$ we follow the process detailed below:

- Append $1, 2$ to the satisfaction vector for α.
- Append $2, 1$ to the satisfaction vector for β.
- Append $1, 1$ to the satisfaction vector for any alternative γ such that $r_{c,\succeq}(\gamma) < r_{c,\succeq}(\alpha)$
- Append $2, 2$ to the satisfaction vector for any alternative γ such that $r_{c,\succeq}(\gamma) > r_{c,\succeq}(\beta)$
- Append $1, 1$ to the satisfaction vector for any alternative γ such that $r_c c, \succeq(\alpha) < r_{c,\succeq}(\gamma) < r_{c,\succeq}(\beta)$ and $\alpha \not\bowtie \gamma$.
- Append $2, 2$ to the satisfaction vector for any alternative γ such that $r_c c, \succeq(\alpha) < r_{c,\succeq}(\gamma) < r_{c,\succeq}(\beta)$ and $\alpha \bowtie \gamma$.

For this construction we assume, without loss of generality, that $r_{c,\succeq}(\alpha) < r_{c,\succeq}(\beta)$.

Using this construction any two incomparable alternatives now have Pareto incomparable satisfaction vectors, since they each have at least one element strictly better than the other. Importantly, if there is a more preferred alternative than the pair each added element is at least as preferred as the ones added to the pair, thus preserving the dominance relation. Similar can be said for alternatives which are less preferred than the pair.

For alternatives which have a canonical rank in between those of the pair we know, by Lemma 5, they must be incomparable with at least one of the alternatives in the pair. The elements added to the satisfaction vectors of "in between" alternatives are such that dominance will still be satisfied after they are added.

Thus the constructed set of satisfaction vectors is such that $s(\alpha) \geq_{\text{Pareto}} s(\beta)$ if and only if $\alpha \succeq \beta$ holds. This means that for any arbitrary preorder $\succeq$ there is a preference theory P such that for any pair of alternatives $\alpha, \beta \in \mathcal{C}(\mathcal{V})$, $\alpha \succeq_P \beta$ if and only if $\alpha \succeq \beta$. □

The above proofs show the validity of edges shown in Figure 1, which are not covered in the pre-existing literature. It is important that we also prove that for each missing edge/path between a pair of languages there is no subsumption relation and if there is a one-way edge that the languages are not equivalent. We omit many of these proofs for space considerations, but give one such example below.

Proposition 2. *UICP LP-Tree $\not\sqsupseteq$ CIUP LP-Tree*

Proof. Consider the UICP LP-tree π in Figure 2. Also, consider building a CIUP LP-tree π' to replicate its preorder. First we must select the root node of π'. Suppose we select an attribute that is not A. Since there are only two attributes

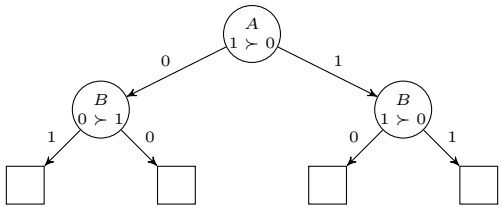

Figure 2: A UICP LP-tree whose preorder cannot be derived from a CIUP LP-tree.

if A is not the root then B must be the root. Suppose we select $0 \succ 1$ for the preference over values of B in π'. In this case (assuming alternatives are represented in the order AB) $01 \succ 10$ regardless of the value of A, which violates the preorder given by π. A similar argument can be made for $1 \succ 0$ being the preference over B.
This indicates that root node of π' cannot be B and so must be A, it is clear that the preference over values of A must be the same in π' as in π. Since there are only two attributes both children in π' must be B. Given that π' has unconditional preferences there are only two possible cases for preferences over B, either $0 \succ 1$ or $1 \succ 0$.
Case 1: $1 \succ 0$
In this case, π' will express the relation $01 \succ 00$ while π expresses the opposite, thus it cannot be the preference for B if π' is to derive the same preorder as π.
Case 2: $0 \succ 1$
In this case, π' will express the relation $10 \succ 11$ while π expresses the opposite, thus it cannot be the preference for B if π' is to derive the same preorder as π.
Since neither case expresses the same preorder as π it is clear that we cannot construct a π' such that it expresses the same preorder as π. Thus, UICP LP-Tree $\not\sqsupseteq$ CIUP LP-Tree. □

As mentioned above most of the subsumption relations we deal with are not efficient. Efficient subsumption exists between the various different types of LP-tree. This is due to their similar interpretation with differences being localized to limits placed on their structure. As another example of efficient subsumption we look at PTs with conditions and those without.

Proposition 3. *Given a PT $P = (\boldsymbol{R} = (r_1, r_2, \ldots, r_n), \boldsymbol{C} = (c_1, c_2, \ldots, c_n))$ where rule r_i has condition c_i, it is possible to create a PT $P' = (\boldsymbol{R}' = (r'_1, r'_2, \ldots, r'_n), \emptyset)$, in polynomial time wrt the size of P, such that for two alternatives α and β, $\alpha \succeq_P \beta$ if and only if $\alpha \succeq_{P'} \beta$.*

Proof. We begin by constructing P' from P. To do this we consider an arbitrary RPF from P,

$$r_i = (\varphi_1, \varphi_2, \ldots, \varphi_k),$$

and its condition c_i. Next, we create the RPF

$$r'_i = (\varphi_1 \wedge c_i, \varphi_2 \wedge c_i, \ldots, \varphi_k \wedge c_i)$$

for P'. It is easy to see that this creation process takes a polynomial amount of time and space wrt the size of P.

We now prove that $\alpha \succeq_P \beta$ if and only if $\alpha \succeq_{P'} \beta$, To this end, we observe that both P and P' follow the same method of deriving preference, i.e. satisfaction vectors. Thus if both produce the same satisfaction vectors then they will produce the same weak preference relation. Consider an arbitrary RPF and condition $r_i, c_i \in P$, its constructed equivalent r'_i, and an alternative α. If α does not satisfy c_i then it receives a satisfaction value of ∞ for r_i. Similarly in r'_i none of the ranks can be satisfied without satisfying c_i and so the satisfaction value for r'_i is also ∞ when α does not satisfy c_i. Consider the case where α does satisfy c_i. In this case the rules are identical because $\Phi \wedge T$ is logically equivalent to Φ. Thus, the satisfaction vectors produced by P and P' are identical meaning $\alpha \succeq_P \beta$ if and only if $\alpha \succeq_{P'} \beta$. □

Consider our proof for Proposition 1. In that proof, we show that a PT does not require the ability to rank or condition its RPFs in order to represent a given preorder. This means that a PT does not gain the ability to express additional preorders through the use of these language features making them expressively redundant. These features may allow for some preference expressions to be significantly more compact than those without rankings, but as Proposition 3 shows, this does not hold wrt adding conditions. Ranking, on the other hand, provides the ability to express meta-knowledge about preferences and alter the mechanism by which preference is decided. While we are currently unable to prove non-existence of efficient subsumption between unranked and ranked PTs we conjecture this is the case, simply due to the modification being external to the satisfaction values themselves. If this is the case it would show that while ranking does not improve expressive power it has the capability of making a PT more compact.

Discussion and Conclusions

Our case study shows a few interesting uses of subsumption. Firstly, subsumption creates a preorder over the set of preference representation languages. Ordering the languages using subsumption allows us to make broad comments about how expressive those languages are. For example, of the languages considered, PTs are the most expressive, while LPMs are one of the least expressive.

Another use of subsumption is the ability to compare various language features in terms of trade-offs. For example, we see that by limiting the type of utility function to a weighted average that the span of expressible orders is reduced. Similarly, we find that for LP-trees conditional importance and conditional preference cause an increase in expressive power over LPMs, and that conditional importance and conditional preference represent two separate expansions of expressive power, i.e. adding one is not the same as adding the other.

As a general trend we find that the graph in Figure 1 shows us that as languages become less expressive they tend to become more compact. As we move to less expressive preference languages, such as weighted averages and LPMs we find that preference expressions become relatively compact, that is polynomial in size wrt the number of attributes in the domain. This implies that the more expressive a language, the less compact its preference expressions may be.

While useful as a tool for analyzing languages, subsumption also has its limitations. First and foremost subsumption is a qualitative measure. This means that even if two languages are capable of deriving equivalent preference expressions except for a single case (one on each side), then the two languages will be rendered incomparable. We see this with CLPMs and CP-nets. There is a significant amount of overlap between CLPMs and CP-nets in terms of equivalent preference expressions (Huelsman and Truszczyński 2020). Subsumption cannot indicate such a relation.

This work introduces many open questions. Firstly, the definition of subsumption we provide here makes the assumption that the languages we deal with consist of static, deterministic expressions. The current definition of subsumption is inadequate for use in cases where expressions are probabilistic or dynamic and new definitions and perhaps even proof techniques will be needed. We leave this as an open question. Another open problem is the completion of Figure 1 through the addition of other languages. Finally, this analysis leaves open the question of a quantitative measure of language similarity.

References

Booth, R.; Chevaleyre, Y.; Lang, J.; Mengin, J.; and Sombattheera, C. 2010. Learning conditionally lexicographic preference relations. In *ECAI*, 269–274.

Boutilier, C.; Brafman, R. I.; Domshlak, C.; Hoos, H. H.; and Poole, D. 2004. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research* 21:135–191.

Brewka, G.; Niemelä, I.; and Truszczynski, M. 2003. Answer set optimization. In *IJCAI*, volume 3, 867–872.

Fishburn, P. C. 1974. Lexicographic orders, utilities and decision rules: A survey. *Management science* 20(11):1442–1471.

Haddawy, P., and Hanks, S. 1992. Representations for decision-theoretic planning: Utility functions for deadline goals. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, 71–82. Morgan Kaufmann Publishers Inc.

Huelsman, M., and Truszczyński, M. 2020. A lexicographic strategy for approximating dominance in CP-nets. In *The Thirty-Third International Flairs Conference*.

Kaci, S. 2011. *Working with Preferences: Less is More*. Springer.

Kohli, R., and Jedidi, K. 2007. Representation and inference of lexicographic preference models and their variants. *Marketing Science* 26(3):380–399.

Von Neumann, J., and Morgenstern, O. 1944. *Theory of games and economic behavior*. Princeton university press.

Zhu, Y. 2016. Preferences: Optimization, importance learning and strategic behaviors.