An incomplete knowledge compilation map for conditional preference statements-based languages

Jérôme Mengin

An incomplete knowledge compilation map for conditional preference statements-based languages

Jérôme Mengin

IRIT - Université de Toulouse 3 - France

Jerome.Mengin@irit.fr

Résumé

Les assertions de préférences conditionnelles (CP-statements) permettent de représenter de manière compacte les préférences sur des domaines combinatoires. Elles sont au cœur des CP-nets et de leurs généralisations, et des arbres de préférences lexicographiques. Plusieurs travaux ont abordé la complexité de certaines requêtes liées à ces formalismes (optimisation, dominance en particulier). Cet article étend certains de ces résultats, et s'intéresse à d'autres requêtes (comme l'équivalence), contribuant ainsi à une carte de compilation pour les langages basés sur les assertions de préférences conditionnelles.

Abstract

Conditional preference statements have been used to compactly represent preferences over combinatorial domains. They are at the core of CP-nets and their generalizations, and lexicographic preference trees. Several works have addressed the complexity of some queries (optimization, dominance in particular). We extend in this paper some of these results, and study other queries which have not been addressed so far, like equivalence, thereby contributing to a knowledge compilation map for languages based on conditional preference statements.

1 Introduction

Preference handling is a key component in several areas of Artificial Intelligence, notably for decision-aid systems. Research in Artificial Intelligence has led to the development of several languages that enable compact representation of preferences over complex, combinatorial domains. Some preference models rank alternatives according to their values given by some multivariate function; this is the case for instance with valued constraints [26], additive utilities and their generalizations [24, 10]. Ordinal models like CP nets and their generalisations [5, 29, 8], or lexicographic preferences and their generalisations [21, 27, 30, 4, 11, 18] use sets of conditional preference statements to represent a pre-order over the set of alternatives.

Many problems of interest, like comparing alternatives or finding optimal alternatives, are at least NP-hard for some of these models, which makes these representations difficult to use in some decision-aid systems like configurators, where real-time interaction with a decision maker is needed. One approach to tackle this problem is Knowledge Compilation, whereby a model, or a part of it, is *compiled*, off-line, into another representation which enables fast query answering, even if the compiled representation has a much bigger size. This approach has first been studied in propositional logic: [14, 15] compare how various subsets of propositional logic can succinctly, or not, express some propositional knowledge bases, and the complexity of queries of interest. [13] follow a similar approach to compare extensions of propositional logic which associate real values to models of a knowledge base; [19] provide such a map for value function-based models.

The aim of this paper is to initiate such a compilation map for ordinal models of preferences. Specifically, we compare the expressiveness and succinctness of various languages based on conditional preference statements, and the complexity of several queries of interest for these languages.

The next section recalls some basic definitions about combinatorial domains and pre-orders, and introduces notations that will be used throughout. Section 3 gives an overview of various languages based on conditional preference statements that have been studied in the literature. Section 4 and 5 respectively study expressiveness and succinctness for languages based on conditional preference statements . Sections 6 study the complexity of queries for these languages. Proofs are omitted due to lack of space.

2 Preliminaries

2.1 Combinatorial domain

We consider languages that can be used to represent the preferences of a decision maker over a combinatorial space $\underline{\mathcal{X}}$: here $\mathcal{X}$ is a set of attributes that characterise the possible alternatives, each attribute $X \in \mathcal{X}$ having a finite set of possible values $\underline{X}$; then $\underline{\mathcal{X}}$ denotes the cartesian product of the domains of the attributes in $\mathcal{X}$, its elements are called alternatives. For binary attribute X, we will often denote by $x, \bar{x}$ its two possible values.

For a subset U of $\mathcal{X}$, we will denote by $\underline{U}$ the cartesian product of the domains of the attributes in U, called instantiations of U, or partial instantiations (of $\mathcal{X}$). If v is an instantiation of some $V \subseteq \mathcal{X}$, $v[U]$ denotes the restriction of v to the attributes in $V \cap U$; we say that instantiation $u \in \underline{U}$ and v are compatible if $v[U \cap V] = u[U \cap V]$; if $U \subseteq V$ and $v[U] = u$, we say that v extends u.

Sets of partial instantiations can often be conveniently, and compactly, specified with propositional formulas: the propositions are $X = x$ for every $X \in \mathcal{X}$ and $x \in \underline{X}$, and we use the standard connectives $\wedge$ (conjunction), $\vee$ (disjunction), $\rightarrow$ (implication), $\leftrightarrow$ (equivalence) and $\neg$ (negation). Implicitly, this propositional logic is equipped with a theory that enforces that every attribute has precisely one value from its domain; so, for two distinct values x, x' of attribute X, the formula $X = x \wedge X = x'$ is a contradiction; also, the interpretations are thus in one-to-one correspondence with $\underline{\mathcal{X}}$. If α is such a propositional formula over $\mathcal{X}$ and $o \in \underline{\mathcal{X}}$, we will write $o \models \alpha$ when o satisfies α, that is when, assigning to every literal $X = x$ that appears in α the value true if $o[X] = x$, and the value false otherwise, makes α true.

We assume that the domains of the attributes in $\mathcal{X}$ are disjoint, so that, given a formula α, or a partial instantiation u, we can unambiguously define $\mathsf{Var}(\alpha)$ and $\mathsf{Var}(u)$ as the set of variables, the values of which appear in α and u respectively.

When it is not ambiguous, we will use x as a shorthand for the literal $X = x$; also, for a conjunction of such literals, we will omit the $\wedge$ symbol, thus $X = x \wedge Y = \bar{y}$ for instance will be denoted $x\bar{y}$.

2.2 Preference relations

2.2.1 Preorders

Depending on the knowledge that we have about a decision maker's preferences, given any pair of distinct alternatives $o, o' \in \underline{\mathcal{X}}$, one of the following situations must hold: one may be strictly preferred over the other, or o and o' may be equally preferred, or o and o' may be incomparable.

Assuming that preferences are transitive, such a state of knowledge about the DM's preferences can be characterised by a preorder $\succeq$ over $\underline{\mathcal{X}}$: $\succeq$ is a binary, reflexive and transitive relation; for alternatives o, o', we then write $o \succeq o'$ when $(o, o') \in \succeq$; $o \succ o'$ when $(o, o') \in \succeq$ and $(o', o) \notin \succeq$; $o \sim o'$ when $(o, o') \in \succeq$ and $(o', o) \in \succeq$; $o \bowtie o'$ when $(o, o') \notin \succeq$ and $(o', o) \notin \succeq$. Note that for any pair of alternatives $o, o' \in \underline{\mathcal{X}}$ either $o \succ o'$, or $o' \succ o$, or $o \sim o'$ or $o \bowtie o'$. The relation $\sim$ defined in this way is the *symmetric part* of $\succeq$, it is reflexive and transitive, $\bowtie$ is irreflexive, they are both symmetric. The relation $\succ$ is the *irreflexive part* of $\succeq$, it is what is usually called a strict partial order: it is irreflexive and transitive.

Terminology and notations We say that alternative o *dominates* alternative o' (w.r.t. $\succeq$) if and only if $o \succeq o'$; if $o \succ o'$, then we say that o *strictly dominates* o'. We use standard notations for the complements of $\succ$ and $\succeq$: we write $o \not\succeq o'$ when it is not the case that $o \succeq o'$, and $o \not\succ o'$ when it is not the case that $o \succ$. We will denote by $\preceq$ (respectively $\prec$) the dual of $\succeq$ (resp. of $\succ$), also called its converse or transpose: $o \preceq o'$ if and only if $o' \succeq o$, and $o \prec o'$ iff $o' \succ o$. Note that since $\bowtie$ and $\sim$ are symmetric, they are equal to their dual.

Following [23] we say that alternative o is:

- weakly undominated if there is no $o' \in \underline{\mathcal{X}}$ such that $o' \succ o$;
- undominated if there is no $o' \in \underline{\mathcal{X}}$, $o' \neq o$, such that $o' \succeq_\varphi o$;
- dominating if for every $o' \in \underline{\mathcal{X}}$, $o \succeq_\varphi o'$;
- strongly dominating if for every $o' \in \underline{\mathcal{X}}$ with $o' \neq o$, $o \succ_\varphi o'$.

Note that o is strongly dominating if and only if it is dominating and undominated; and that if o is dominating or undominated, then it is weakly undominated.

2.2.2 Antisymmetric preorders

Some languages designed to represent preferences are associated with a primitive relation which is not a preorder but a strict partial order $\succ$. However, it is possible to define a preorder $\succeq$ as the reflexive closure of $\succ$: that is, $o \succeq o'$ holds when $o \succ o'$ or $o = o'$. The relation $\succeq$ defined in this way is antisymmetric.

3 Languages

3.1 Conditional preference statements

Let us call *conditional preference statement*, or CP statements, over $\mathcal{X}$ any expression of the form $\alpha \,|\, V : w \geq w'$, where α is a propositional formula over $U \subseteq \mathcal{X}$, $w, w' \in \underline{W}$, $w \neq w'$, and U, V, W are disjoint subsets of $\mathcal{X}$. Informally, such a statement represents the piece of knowledge that, when comparing alternatives o, o' that both satisfy α, the one that has values w for W is preferred to the one that has values w' for W, irrespective of the values of the

attributes in V, every other attribute being fixed. We call α the conditioning part of the statement; we call W the swapped attributes, and V the free part.

Conditional preference statements have been studied in many works. They are the basis for CP-nets [7, 5] and their extensions, and have been studied in a more logic-based fashion by e.g. [23] and [29, 28, 31]. In all these works, a syntactic restriction is put on W: it must be the case that $|W| = 1$. Also, [23] do not consider any free part ($V = \emptyset$), and [29, 28, 31] only considers statements with a conjunctive conditioning part (α must be a consistent conjunction of literals).[1]

The semantics of a set φ of conditional preference statements can be defined as follows: consider a pair of alternatives (o, o') such that there is a statement $\alpha \,|\, V : w \geq w' \in \varphi$ with $o[U] = o'[U] \models \alpha$, $o[W] = w$ and $o'[W] = w'$, and such that for every attribute $Y \notin U \cup V \cup W$ it holds that $o[Y] = o'[Y]$; following [31] we say that (o, o') is a *worsening swap*. We also say that the statement $\alpha \,|\, V : w \geq w' \in \varphi$ *sanctions* (o, o'). Let φ^* be the set of all worsening swaps that φ sanctions, and define $\succeq_\varphi$ to be the reflexive and transitive closure of φ^*. [31] proves that $o \succeq_\varphi o'$ if and only $o = o'$ or φ^* contains a finite sequence of worsening swaps $(o_i, o_{i+1})_{0 \leq i \leq k-1}$ with $o_0 = o$ and $o_k = o'$.[2]

The language of the above statements is very expressive: in fact, by considering a set $W = \mathcal{X}$, and $\alpha = \top$ and $V = \emptyset$, it is possible to represent any preorder "in extension" with preference statements of the form $o \geq o'$. Let us call:

- CP the language where formulas are sets of statements of the general form $\alpha \,|\, V : w \geq w'$;

This expressiveness has a cost: we will see that many queries about pre-orders represented by CP-statements are PSPACE-hard for the language CP. Several restrictions / sub-languages have been studied in the literature, we review them below. Note that formulas in CP are not required to verify any form of consistency or completeness; such conditions, will be imposed for some sublanguages defined below.

Notations We write $\alpha : w \geq w'$ when V is empty, and $w \geq w'$ when V is empty and $\alpha = \top$, the formula always true. Note that we reserve the symbol $\geq$ for conditional preference statements, whereas "curly" symbols $\succ$, $\not\succ$, $\succeq$, $\not\succeq$ are used to represent relations over the set of alternatives.

In the remainder of this section, we introduce various restrictions on formulas. Table 1 gives an overview of these restrictions, as well as some complexity results that will be detailed in section 6.

[1] The formula $u \,|\, V : x \geq x'$ is written $u : x > x'[V]$ by [31].

[2] Actually, [31] proves that (o, o') is in the transitive closure of φ^* if and only there is such a worsening sequence from o to o', but adding the reflexive closure to this transitive closure does not change the result, since we can add any pair (o, o) to, or remove it from, any sequence of worsening swaps without changing the validity of the sequence.

3.2 Statement-wise restrictions

Some restrictions apply on the syntactical form of statements allowed; they bear on the size of the set of free variables, or on the size of the set of swapped variables, or on the type of conditioning formulas allowed. Given some language $\mathcal{L} \subseteq \mathsf{CP}$, we define the following restrictions:

- $\mathcal{L}\not\emptyset$ is the restriction of $\mathcal{L}$ to formulas with empty free parts ($V = \emptyset$) for every statement;[3]
- $\mathcal{L}\wedge$ is the restriction of $\mathcal{L}$ to formulas where the condition α of every statement is a conjunction of literals;
- $\mathcal{L}\mathsf{k}$ is the restriction of $\mathcal{L}$ to formulas where the set of swapped attributes contains no more than k attributes ($|W| \leq k$) for every statement; in particular, we call elements of $\mathsf{CP1}$ *unary* statements.

In particular, $\mathsf{CP1}\wedge$ corresponds to the language studied by [31], and $\mathsf{CP1}\not\emptyset$ is the language of generalized CP-nets as defined by [23].

3.3 Graphical restrictions

CP-statements describe some interactions between attributes. Many tractability results on CP-statements based languages require that the graph of these interactions has some "good" properties. In their seminal work, [5] consider that these interaction can be elicited first from a decision-maker, and that this structure can be used to render easier the elicitation of $\mathsf{CP1}$ statements that represent her preferences. However, these interactions can also be extracted *a posteriori*, for any set of CP statements.

The graph defined by [5] is restricted to the case where all CP statements are unary and have no free variables. This definition has been extended by [8, 31] to cover the case of statements with free variables. The following definition is inspired by [31, Def. 15]. Given $\varphi \in \mathsf{CP}$ over set of attributes $\mathcal{X}$, we define the following graphs with sets of vertices $\mathcal{X}$: given $X, Y \in \mathcal{X}$

- $(X, Y) \in D_\varphi^{\mathsf{uncond}}$ if there is some statement $\alpha \,|\, V : w \geq w' \in \varphi$ such that $X \in \mathsf{Var}(\alpha)$ and $Y \in \mathsf{Var}(w) \cup V$;
- for every alternative o, $(X, Y) \in D_\varphi^{\mathsf{cond}}(o)$ if there is some statement $\alpha \,|\, V : w \geq w' \in \varphi$ such that $o \models \alpha$, $X \in \mathsf{Var}(w)$ and $Y \in V$;
- $D_\varphi = D_\varphi^{\mathsf{uncond}} \cup \bigcup_{o \in \mathcal{X}} D_\varphi^{\mathsf{cond}}(o)$.

Given some language $\mathcal{L} \subseteq \mathsf{CP}$, we define:

- $\mathcal{L}\not\circlearrowleft$ the restriction of $\mathcal{L}$ to *acyclic* formulas, which are those φ such that D_φ is acyclic;[4];

[3] In the literature, the symbol $\rhd$ is sometimes used to represent an *importance* relation between attributes; and, as explained by [31], statement $\alpha \,|\, V : w \geq w'$ is a way to express that attributes in $\mathsf{Var}(w)$ have more importance those in V (when α is true).

[4] This is *full acyclicity* in [31].

Properties/Restrictions									
Base language				CPnet		CPnet	CPnet	LPT	LTP
Unary swaps / nodes		1	1	(1)	1	(1)	(1)		1
No free variable		$\not\rhd$	$\not\rhd$	($\not\rhd$)		($\not\rhd$)	($\not\rhd$)		
Condition in conjunctive form		$\wedge$		($\wedge$)	$\wedge$	($\wedge$)	($\wedge$)		
Local consistency			$\not\perp^{\mathsf{loc}}$	($\not\perp^{\mathsf{loc}}$)	$\not\perp^{\mathsf{loc}}$	($\not\perp^{\mathsf{loc}}$)	($\not\perp^{\mathsf{loc}}$)		
Local completeness			$\top^{\mathsf{loc}}$	($\top^{\mathsf{loc}}$)		($\top^{\mathsf{loc}}$)	($\top^{\mathsf{loc}}$)		
Acyclicity					cuc	full	polytree		
Queries									
LINEARISABILITY	✘!	✘!	✘!		$\top$	$\top$	$\top$	$\top$	$\top$
EQUIVALENCE	✘!	✘!		✓	✘!	✓	✓	✓	✓
R-COMPARISON, $R \in \{\succeq, \succ, \bowtie\}$	✘!	✘!	✘!	✘!	✘!	✘!	✓	✓	✓
$\sim$-COMPARISON	✘!	✘!	✘!		✓	✓	✓	✓	✓
2-ORDERING					✓	✓	✓	✓	✓
UNDOMINATED CHECK	✓	✓	✓	✓	✓	✓	✓	✓	✓
S. DOM., DOM., W. UNDOM. CHECK	✘!	✘!	✘!			✓	✓	✓	✓
S. DOM. $\exists$, DOM. $\exists$	✘!		✘!			$\top$	$\top$	✓	✓
UNDOMINATED $\exists$	✘!	✘!			$\top$	$\top$	$\top$	$\top$	$\top$
W. UNDOMINATED $\exists$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$

Table 1: Language restrictions and complexity of queries: ✓ = indicates that the query can be solved in time polynomial in $|\varphi| + |\mathsf{result}|$; ✘! = no such algorithm unless P= NP, or PSPACE= P; $\top$ = always true for the language.

- $\mathcal{L}\ \not\circlearrowleft^{\mathsf{cuc}}$ the restriction of $\mathcal{L}$ to cuc-acyclic formulas, which are those φ such that for every alternative o, $D_\varphi^{\mathsf{uncond}} \cup D_\varphi^{\mathsf{cond}}(o)$ is acyclic.[5]

Note that D_φ can be computed in polynomial time. Moreover, in the case of $\mathsf{CP1}\not\rhd$, the $D_\varphi^{\mathsf{cond}}(o)$'s are all empty, so cuc-acyclicity reduces to D_φ being acyclic, and D_φ is the set of all (X, Y) such φ contains some statement $u : y > y'$ with $X \in \mathsf{Var}(u)$ – which is the definition used by [5].

In the more general case of CP1, checking cuc-acyclicity can be hard [8, Th. 3], [31, Prop. 24].

3.4 Attribute-wise restrictions

It is possible, especially with CP1 statements, to consider restrictions that guarantee some form of completeness and consistency on the conditions that sanction swaps on a given variable X. In other words, the idea is that for every pair of alternatives $o, o' \in \underline{\mathcal{X}}$ such that o and o' are equal except for their value for one attribute, there must be exactly one statement in a CP-net that orders o and o'. These conditions are implicit in CP-nets defined by [5], and have been formally defined by [23] in a slightly more restrictive context (binary attributes) and, in part, by [31].

Definition 1 (Local completeness and local consistency [23, 31]). Let $\varphi \in \mathsf{CP1}$. For every attribute $X \in \mathcal{X}$ and every partial instantiation u, define $\geq_\varphi^{X,u}$ to be the reflexive and transitive closure of the set of all pairs $(x, x') \in \underline{\mathcal{X}}^2$ such that there exists some $\alpha \,|\, V : x \geq x' \in \varphi$ with $u \models \alpha$; then φ is *locally consistent* if $\geq_\varphi^{X,o}$ is antisymmetric for every attribute $X \in \mathcal{X}$ and every alternative $o \in \underline{\mathcal{X}}$; and φ is *locally complete* if $\geq_\varphi^{X,o}$ is a total preorder for every attribute X and every alternative $o \in \underline{\mathcal{X}}$.

Given some language $\mathcal{L} \subseteq \mathsf{CP}$, we define:

- $\mathcal{L}\ \not\perp^{\mathsf{loc}}$ is the restriction of $\mathcal{L}$ to those formulas that are locally consistent;
- $\mathcal{L}\top^{\mathsf{loc}}$ is the restriction of $\mathcal{L}$ to those formulas that are locally complete.

Note that the problem of checking if a formula of $\mathsf{CP1}\wedge$ is locally consistent is coNP complete [31, Prop. 11]. Locally complete and locally consistent formulas of $\mathsf{CP1}\not\rhd$, that is, formulas of $\mathsf{CP1}\not\rhd\ \not\perp^{\mathsf{loc}}\ \top^{\mathsf{loc}}$, are called *CP-nets* by [23]. However, we recall next the original definition of CP-nets, which is slightly different.

3.5 CP-nets

In their seminal work, [5] define a CP-net over a set of attributes $\mathcal{X}$ to be composed of two elements:

1. a directed graph over $\mathcal{X}$, which should represent *preferential dependencies* between attributes;[6]

[5] This definition is weaker than the one given by [31], who also imposes local consistency as will be defined shortly; it corresponds to the definition of *conditional acyclicity* as given by [8].

[6] Given some pre-order $\succeq$ over $\mathcal{X}$, attribute X is said to be preferentially dependent on attribute Y if there exist $x, x' \in \underline{X}$, $y, y' \in \underline{Y}$, $z \in \underline{\mathcal{X}\ (\{X, Y\})}$ such that $xyz \succeq_\varphi x'yz$ but $xy'z \not\succeq_\varphi xy'z$.

2. a set of conditional preference tables, one for every attribute X: if U is the set of parents of X in the graph, the conditional preference table for X contains exactly $|\underline{U}|$ rules $u:\geq$, for every $u \in \underline{U}$, where the $\geq$'s are linear orders over $\underline{X}$.

Therefore, as shown by [31], CP-nets can be seen as sets of unary CP statements in conjunctive form with no free attribute. Specifically, given a CP-net $\mathcal{N}$ over $\mathcal{X}$, define $\varphi_{\mathcal{N}}$ to be the set of all CP statements $u:x \geq x'$, for every attribute X, every $u \in \mathsf{Pa}(X)$, every $x, x' \in \underline{X}$ such that x are consecutive values in the linear order $\geq$ specified by the rule $u:\geq$ of $\mathcal{N}$. $\mathcal{N}$ being a CP-net enforces a very strong form of local consistency and completeness: it must the case that, for every attribute X with parents U, for every $u \in \underline{U}$, for every $x, x' \in$, the CP-net must explicitly, and uniquely, order ux and ux'.

Thus we call

- CPnet the language that contains all $\varphi_{\mathcal{N}}$, for every CP-net $\mathcal{N}$.

Note that $\mathsf{CPnet} \subseteq \mathsf{CP1}\not\triangleright\wedge \not\perp^{\mathsf{loc}} \top^{\mathsf{loc}}$. [8] define TCP-nets as an extension of CP-nets where it is possible to represent tradeoffs, by stating that, under some condition, some attribute is more important than another one. [31] describes how TCP-nets can be transformed, in polynomial times, into equivalent sets of $\mathsf{CP1}\wedge$ statements.

3.6 Semantic restriction

Although the original definition of CP-nets by [7] does not impose it, many works on CP-nets, especially following [5], consider that they are intended to represent a strict partial order, that is, that $\succeq_\varphi$ should be antisymmetric; equivalently, this means that φ^* can be extended to a linear order. We say that a set φ of CP-statements is *linearisable* if φ^* can be extended to a linear order;[7] in this case, several authors define $\succ_\varphi$ to be the transitive closure of φ^*, which leads to the same definition of preorder $\succeq_\varphi$ as ours. But, like [23], we use the same definition for $\succeq_\varphi$ even when φ^* is not acyclic; note that in this case $\succeq_\varphi$ is not antisymmetric.

Note that, if φ is linearisable, then it is locally consistent.

3.7 Lexicographic preference trees

LP-trees generalise lexicographic orders. As an inference mechanism, they are equivalent to search trees used by [6], and formalised by [28, 31]. As a preference representation, and elicitation, language, slightly different definitions for LP-trees have been proposed by [4, 11, 18]. We use here a definition which subsumes the others.

[7] Such sets of CP-statements are often called *consistent* in the standard terminology on CP-nets, but we prefer to depart from this definition which only makes sense when one asserts that φ should indeed represent a strict partial order.

An LP-tree over $\mathcal{X}$ is a rooted tree with labelled nodes and edges, and a set of preference tables; specifically

- every node N is labelled with a set $W \subseteq \mathcal{X}$;
- we denote by $\mathsf{Anc}(N)$ the set of attributes that appear in the nodes above N (excluding those at N), and by $\mathsf{NonInst}(N)$ the set of attributes that appear in the nodes above N that have only one child;
- if N is not a leaf, it can have one child, or $|\underline{W}|$ children;
- in the latter case, the edges that connect N to its children are labelled with the instantiations in $\underline{W}$;
- if N has one child only, the edge that connect N to its child is not labelled;
- a conditional preference table $\mathsf{CPT}(N)$ is associated with N: it contains local preference rules of the form $\alpha:\geq$, where $\geq$ is a preorder over $\underline{W}$, and α is a propositional formula over some attributes in $U \subseteq \mathsf{NonInst}(N)$.

We assume that the rules in $\mathsf{CPT}(N)$ define their preorder in extension. Additionally, two constraints guarantee that an LP-tree φ defines a unique preorder over $\underline{\mathcal{X}}$:

- no attribute can appear at more than one node on any branch of φ; and,
- at every node N of φ, for every $u \in \mathsf{NonInst}(N)$, $\mathsf{CPT}(N)$ must contain exactly one rule $\alpha:\geq$ such that $u \models \alpha$.

Given an LP-tree φ and an alternative $o \in \underline{\mathcal{X}}$, there is a unique way to traverse the tree, starting at the root, and along edges that are either not labelled, or labelled with instantiations that agree with o, until a leaf is reached. Now, given two distinct alternatives o, o', it is possible to traverse the tree along the same edges as long as o and o' agree, until a node N is reached which is labelled with some W such that $o[W] \neq o'[W]$: we say that N decides $\{o, o'\}$.

In order to define $\succeq_\varphi$ for some LP-tree φ, we define φ^* to be the set of all pairs of distinct alternatives (o, o') such that there is a node N that decides $\{o, o'\}$ and the only rule $\alpha:\geq \in \mathsf{CPT}(N)$ with $o[\mathsf{NonInst}(N)] = o'[\mathsf{NonInst}(N)] \models \alpha$ is such that $o[W] \geq o'[W]$. Then $\succeq_\varphi$ is the reflexive closure of φ^*.

Proposition 1 ([3]). *Let φ be an LP-tree over $\mathcal{X}$, then $\succeq_\varphi$ as defined above is a preorder.*

An LP-tree is said to be complete if every attribute appears on every branch, and if every preference rule specifies a linear order; $\succeq_\varphi$ is then a linear order too.

From a semantic point of view, an LP-tree φ is equivalent to the set that contains, for every node N of φ labelled with W , and every rule $\alpha:\geq$ in $\mathsf{CPT}(N)$, all CP statements of the form $\alpha \wedge u \,|\, V : w \geq w'$, where

- u is the combination of values given to the attributes in $\mathsf{Anc}(N) - \mathsf{NonInst}(N)$ along the edges between the root and N, and

- $w, w' \in \underline{W}$ such that $w \geq w'$, and
- $V = [\mathcal{X} - (\mathsf{Anc}(N) \cup W)]$.

We define the following languages:

- LPT is the language of LP-trees as defined above; we consider that LPTis a subset of CP.[8]

Note that, using the notations defined above:

- LPTk = LPT $\cap$ CPk is the restriction of LPT where every node has at most k attributes, for every $k \in \mathbf{N}$; in particular, LPT1 is the language of LP-trees with one attribute at each node;
- LPT$\wedge$ = LPT$\cap$CP$\wedge$ is the restriction of LPT where the condition α in every rule at every node is a conjunction of literals.

LP-trees as defined by [28, 4, 25] are sublanguages of LPT1$\wedge$; and those of [18] and [11] are sublanguages of LPT$\wedge$.

4 Expressiveness

A first criterium for comparing languages is expressiveness:

Definition 2. Let $\mathcal{L}$ and $\mathcal{L}'$ be two languages for representing preorders. We say that:

- $\mathcal{L}'$ is a sublanguage of $\mathcal{L}$ if $\mathcal{L} \supseteq \mathcal{L}'$; it is a proper sublanguage if $\mathcal{L} \supset \mathcal{L}'$, that is, if $\mathcal{L} \supseteq \mathcal{L}'$ and $\mathcal{L}' \not\supseteq \mathcal{L}$;
- $\mathcal{L}$ is *at least as expressive as* $\mathcal{L}'$, written $\mathcal{L} \sqsupseteq \mathcal{L}'$, if every preorder that can be represented with a formula of $\mathcal{L}'$ can also be represented with a formula of $\mathcal{L}$; we write $\mathcal{L} \sqsupset \mathcal{L}'$ if $\mathcal{L} \sqsupseteq \mathcal{L}'$ but it is not the case that $\mathcal{L}' \sqsupseteq \mathcal{L}$, and say in this case that $\mathcal{L}$ is strictly more expressive than $\mathcal{L}'$.

Note that $\sqsupseteq$ is a preorder, and obviously $\mathcal{L} \supseteq \mathcal{L}'$ implies $\mathcal{L} \sqsupseteq \mathcal{L}'$, but the converse does not hold in general.

Clearly, CP$\not\vartriangleright$ $\subseteq$ CP and CP$\wedge$ $\subseteq$ CP; however, these three language have the same expressiveness:

Proposition 2. *CP, CP$\wedge$ and CP$\not\vartriangleright$ can all three represent every preorder, thus CP $\sqsupseteq$ CP$\not\vartriangleright$ $\sqsupseteq$ CP$\wedge$ $\sqsupseteq$ CP.*

A large body of works on CP-statements since the seminal paper by [6] concentrate on various subsets of CP1. With this strong restriction on the number of swapped variables, CP-theories have a reduced expressiveness.

Example 1 (CP1 $\not\sqsupseteq$ CP). Consider two binary attributes A and B, with respective domains $\{a, \bar{a}\}$ and $\{b, \bar{b}\}$. Define preorder $\succeq$ such that $ab \succ \bar{a}\bar{b}$, with the two remaining alternatives being incomparable to the former and to each other. This can be represented in CP with $\varphi = \{ab \geq \bar{a}\bar{b}\}$. But it cannot be represented in CP1, because this would require at least two rules: one to flip the value of A, the other one to flip the value of B; but then there must be one intermediate alternative comparable with ab and $\bar{a}\bar{b}$.

[8] Strictly speaking, for LPT $\subseteq$ CP to hold, we can add the possibility to augment every formula in CP with a tree structure.

Example 2 (CP1 $\not\sqsupseteq$ CP even if restricted to linear orders). Consider two binary attributes A and B, with respective domains $\{a, \bar{a}\}$ and $\{b, \bar{b}\}$. Define preorder $\succeq$ such that $ab \succ \bar{a}\bar{b} \succ a\bar{b} \succ \bar{a}b$. This can be represented in CP with $\varphi = \{ab \geq \bar{a}\bar{b}, \bar{a}\bar{b} \geq a\bar{b}, a\bar{b} \geq \bar{a}b\}$. But it cannot be represented in CP1: $\{b : a \geq \bar{a}, \bar{b} : \bar{a} \geq a, a : b \geq \bar{b}, \bar{a} : \bar{b} \geq b\}^* \subseteq \varphi^*$, but this is not sufficient to compare $a\bar{b}$ with $\bar{a}b$. The four remaining formulas of CP1 over these two attributes are $B : a \geq \bar{a}$, $B : \bar{a} \geq a$, $A : b \geq \bar{b}$, $A : \bar{b} \geq b$, adding any of them to φ yields a preorder which would not be antisymmetric.

Forbidding free parts incurs an additional loss in expressiveness:

Example 3 (CP1$\not\vartriangleright$ $\not\sqsupseteq$ CP1). Consider two binary attributes A and B, with respective domains $\{a, \bar{a}\}$ and $\{b, \bar{b}\}$. Define preorder $\succeq$ such that $ab \succ a\bar{b} \succ \bar{a}b \succ \bar{a}\bar{b}$. This can be represented in CP1 with $\varphi = \{B : a \geq \bar{a}, b \geq \bar{b}\}$. But it cannot be represented in CP1$\not\vartriangleright$, and it cannot be obtained by transitivity from the comparisons that can be expressed in CP1$\not\vartriangleright$.

However, restricting to conjunctive statements does not incur a loss in expressiveness.

Proposition 3. *CP $\sqsupset$ CP1 $\sqsupset$ CP1$\not\vartriangleright$, CP1$\wedge$ $\sqsupset$ CP1$\wedge\not\vartriangleright$, but CP1 $\sqsupseteq$ CP1$\wedge$ $\sqsupseteq$ CP1 and CP1$\not\vartriangleright$ $\sqsupseteq$ CP1$\wedge\not\vartriangleright$ $\sqsupseteq$ CP1$\not\vartriangleright$.*

LP trees Because an LP-tree can be a single node labelled with $\mathcal{X}$, and a single preference rule $\top : \geq$ where $\geq$ can be any preorder, LPT can represent any preorder.

Proposition 4. *LPT $\sqsupseteq$ CP $\sqsupset$ LPT.*

For LP-trees too, limiting the number of attributes per node reduces expressiveness:

Proposition 5. *LPT $\sqsupset$ LPT$\wedge$ $\sqsupseteq$ LPT, LPT $\sqsupset$ LPTk $\sqsupset$ LPTk$\wedge$ $\sqsupseteq$ LPTk.*

5 Succinctness

Another criterium is the relative sizes of formulas that can represent the same preorder in different languages. [12] study the space efficiency of various propositional knowledge representation formalisms. An often used definition of succinctness makes it a particular case of expressiveness:

Definition 3 ([22, 15]). Let $\mathcal{L}$ and $\mathcal{L}'$ be two languages for representing preorders. We say that $\mathcal{L}$ is *at least as succinct as* $\mathcal{L}'$, written $\mathcal{L} \leqq \mathcal{L}'$, if there exists a polynomial p such that for every formula $\varphi' \in \mathcal{L}'$, there exists a formula $\varphi \in \mathcal{L}$ that represent the same preorder as φ' and such that $|\varphi| < p(|\varphi'|)$.

With this definition, if $\mathcal{L}' \subseteq \mathcal{L}$ then $\mathcal{L} \leqq \mathcal{L}'$; and if $\mathcal{L} \leqq \mathcal{L}'$ then $\mathcal{L} \sqsupseteq \mathcal{L}'$. In particular, if we have two languages such that $\mathcal{L} \supset \mathcal{L}'$ and $\mathcal{L}' \not\supseteq \mathcal{L}$, then $\mathcal{L} \leqq \mathcal{L}'$ and $\mathcal{L}' \not\leqq \mathcal{L}$, even if there is no *real* succinctness hierarchy between the two, it is just that one is strictly more expressive than the other. Therefore, we introduce the following definition for strict succinctness, more restrictive than taking the strict partial order induced by $\leqq$.

Definition 4. Let $\mathcal{L}$ and $\mathcal{L}'$ be two languages for representing preorders. We say that $\mathcal{L}$ is *strictly more succinct than* $\mathcal{L}'$, written $\mathcal{L} \ll \mathcal{L}'$, if $\mathcal{L} \leqq \mathcal{L}'$ and for every polynomial p, there exists $\varphi \in \mathcal{L}$ such that:

- there exists $\varphi' \in \mathcal{L}'$ such that $\succeq_\varphi = \succeq_{\varphi'}$, but
- for every $\varphi' \in \mathcal{L}'$ such that $\succeq_\varphi = \succeq_{\varphi'}$, $|\varphi'| > p(|\varphi|)$.

With this definition, $\mathcal{L} \ll \mathcal{L}'$ if every formula $\varphi \in \mathcal{L}'$ has an equivalent formula in $\mathcal{L}$ which is "no bigger"[9], and there is at least one sequence of formulas[10] in $\mathcal{L}$ that have equivalent formulas in $\mathcal{L}'$ but necessarily "exponentially bigger". Note that $\mathcal{L} \ll \mathcal{L}'$ implies that $\mathcal{L} \leqq \mathcal{L}'$ and $\mathcal{L}' \not\leqq \mathcal{L}$.

Restricting the conditioning part of the statements to be conjunctions of literals does induce a loss in succinctness, because propositional logic is strictly more succinct than the language of DNFs.

Example 4. Consider $2n+1$ binary attributes $X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_n, Z$, and let φ contain $2n+2$ unary CP-statements with no free attribute: $(x_1 \vee y_1) \wedge (x_2 \vee y_2) \wedge \ldots \wedge (x_n \vee y_n) : z \geq \bar{z}$, $\neg[(x_1 \vee y_1) \wedge (x_2 \vee y_2) \wedge \ldots \wedge (x_n \vee y_n)] : \bar{z} \geq z$ and $\bar{x}_i \geq x_i$ and $\bar{y}_i \geq y_i$ for every $i \in \{1, \ldots, n\}$. Then $\varphi \in \mathsf{CP1}\not\triangleright \not\perp^{\mathsf{loc}} \top^{\mathsf{loc}}$, but φ is not in conjunctive form. A set of conjunctive CP-statements equivalent to φ has to contain all 2^n statements of the form $\mu_1\mu_2\ldots\mu_n : z \geq \bar{z}$ with $\mu_i = x_i$ or $\mu_i = \bar{x}_i$ for every i.

Restricting to CP-nets induces a further loss in expressiveness, as the next example shows:

Example 5. Consider $n+1$ binary attributes $X_1, X_2, \ldots, X_n, Y$, and let φ be the $\mathsf{CP1}\not\triangleright\wedge$ formula that contains the following statements: $x_i \geq \bar{x}_i$ for $i = 1, \ldots, n$; $x_1 x_2 \ldots x_n : y \geq \bar{y}$; $\bar{x}_i : \bar{y} \geq y$ for $i = 1, \ldots, n$. The size of φ is linear in n. Because preferences for Y depend on all X_i's, a CP-net equivalent to φ will contain, in the table for Y, 2^n CP statements.

Proposition 6. *For every language such that* $\mathsf{CP1}\not\triangleright \not\perp^{\mathsf{loc}} \top^{\mathsf{loc}} \subseteq \mathcal{L} \subseteq \mathsf{CP}$, $\mathcal{L} \ll \mathsf{CP}\wedge \cap \mathcal{L}$. *Moreover,* $\mathsf{CP1}\not\triangleright\wedge \not\perp^{\mathsf{loc}} \top^{\mathsf{loc}} \ll \mathsf{CP\text{-}net}$.

6 Queries

Linearisability Checking if a given $\varphi \in \mathsf{CP}$ is linearisable, that is, if $\succeq_\varphi$ is antisymmetric, can give some interesting insights into the semantics of φ. The following query has been addressed in many works on CP statements:[11]

LINEARISABILITY Given φ, is φ linearisable?

[5] prove that when its dependency graph D_φ is acyclic, then a CP-net φ is linearisable. This result has been extended by [16, 8, 31], who give weaker graphical conditions that guarantee that a locally consistent set of unary, conjunctive CP statements, that is, a formula of $\mathsf{CP1}\wedge \not\perp^{\mathsf{loc}}$ is linearisable: specifically, every formula of $\mathsf{CP1}\wedge \not\perp^{\mathsf{loc}} \not\circlearrowleft^{\mathsf{cuc}}$ is linearisable. However, checking these conditions is a hard problem. [23, Theorem 3 and 4] prove that LINEARISABILITY is PSPACE-complete for $\mathsf{CP1}\not\triangleright\wedge$, $\mathsf{CP1}\not\triangleright \not\perp^{\mathsf{loc}} \top^{\mathsf{loc}}$.

Comparing theories Checking if two theories yield the same preorder can be useful during the compilation process. We say that two formulas φ and φ' are equivalent if they represent the same preorder, that is, if $\succeq_\varphi$ and $\succeq_{\varphi'}$ are identical; we then write $\varphi \equiv \varphi'$.

EQUIVALENCE Given two formulas φ and φ', are they equivalent?

Consider a formula $\varphi \in \mathsf{CP}$, two alternatives o, o', and let $\varphi' = \varphi \cup \{o \geq o'\}$: clearly $o \succeq_{\varphi'} o'$, thus $\varphi \equiv \varphi'$ if and only if $o \succeq_\varphi o'$. Therefore, if language $\mathcal{L}$ is such that adding CP statement $o \geq o'$ to any of its formulas yields a formula that is still in $\mathcal{L}$, then EQUIVALENCE has to be at least as hard as $\succeq$-COMPARISON for $\mathcal{L}$. This is the case of CP. The problem remains hard for $\mathsf{CP1}\not\triangleright$, because it is hard to check the equivalence, in propositional logic, of the conditions of statements that entail a particular swap $x \geq x'$.

Example 6. Consider three attributes A, B and C with respective domains $\{a, \bar{a}\}$, $\{b, \bar{b}\}$ and $\{c_1, c_2, c_3\}$. Consider two CP statements $s = \bar{a} : c_1 \geq c_2$ and $s' = b : c_2 \geq c_3$, and let $\varphi = \{s, s', a : c_1 \geq c_3\}$. Because of statements s and s' we have $\bar{a}bc_1 \geq_\varphi \bar{a}bc_2 \geq_\varphi \bar{a}bc_3$; also, $abc_1 \geq_\varphi abc_3$ because of statement $a : c_1 \geq c_3$. Hence, for any $u \in \underline{A} \times \underline{B}$, if $u \models a \vee (\bar{a}b)$ then $uc_1 \geq uc_3$. Thus $\varphi \equiv \{s, s'\} \cup \{a \vee (\bar{a}b) : c_1 \geq c_3\} \equiv \varphi \cup \{b : c_1 \geq c_3\}$.

In general, given a propositional language $\mathcal{P}$ we define $\mathcal{P}^\vee$ to be the set of finite disjunctions of formulas in $\mathcal{P}$, and:

- $\mathsf{CP1}\not\triangleright\mathcal{P}$ is $\mathsf{CP1}\not\triangleright$ restricted to those statements such that the condition is in $\mathcal{P}$.

Proposition 7. *Given a propositional language $\mathcal{P}$ closed for conjunction,* EQUIVALENCE *for $\mathcal{P}^\vee$ (in the sense of propositional logic), restricted to consistent formulas, reduces to* EQUIVALENCE *for $\mathsf{CP1}\not\triangleright\mathcal{P}$ restricted to fully acyclic, locally consistent formulas.*

[9] up to some polynomial transformation of the size of φ

[10] one formula for every polynomial p

[11] This query is often called *consistency*

In particular, EQUIVALENCE is NP-hard for $\mathsf{CP1}\not\triangleright\wedge\not\emptyset\not\perp^{\mathsf{loc}}$ restricted to binary attributes, because checking if two propositional, consistent DNFs are equivalent is NP-hard.

However, equivalence is not hard to check for CP-nets, thanks to the existence of a canonical, minimal form: given a CP-net with attributes X and Y such that $X \in \mathsf{Pa}(Y)$, it is easy to check if the preferences that appear in the conditional preference table for Y truly depend on X: if not, the table can be simplified and the edge (X,Y) can be removed. This can be done in polynomial time for all edges (X,Y) of the dependency graph of the CP-net.

For LP-trees too, EQUIVALENCE is easy to check because of the existence of a canonical form: given a node of an LP-tree φ labelled with set of variables S, it is possible to check if it can be split into a "root" node and one or more several children, using an approach like that proposed by [18] for learning an LP-tree from positive examples; this can be done in time polynomial w.r.t. to $|\underline{S}|$, which is itself polynomially bounded by the size of the preference table at S, since we assume that the pre-orders over $\underline{S}$ are given in extension in this table. This procedure can be iterated until no node of the tree can be split. Moreover, if all subtrees of a node are identical, they can be merged into one subtree; applying this in a bottom-up fashion, one obtains the canonical form of the tree; two LP-trees are then equivalent if and only if they have the same canonical form.

Comparing alternatives A basic question, given a formula φ and two alternatives o, o' is: how do o and o' compare, according to φ? Is it the case that $o \succ_\varphi o'$, or $o' \succ_\varphi o$, or $o \bowtie_\varphi o'$, or $o \sim_\varphi o'$? We define the following query, for any relation $R \in \{\succ, \succeq, \sim, \bowtie\}$:

R-COMPARISON Given formula φ, alternatives o, o', is it the case that $oR_\varphi o'$?

For LP-trees, in order to compare alternatives o and o', one only has to traverse the tree from the root downwards until a node that decides the pair is reached, or up to a leaf if no such node is encountered: in this case o and o' are incomparable. Note that checking if a node decides the pair, and checking if a rule at that nodes applies to order them, can both be done in polynomial time.

Proposition 8. *R-COMPARISON is in P for LPT for every $R \in \{\succ, \succeq, \sim, \bowtie\}$.*

The complexity of comparisons has been studied by [5] for CP nets, by [23] for $\mathsf{CP1}\not\triangleright$ and by [31] for $\mathsf{CP1}\wedge$. [23] propose a simple non-deterministic algorithm to prove membership in PSPACE of $\succeq$-COMPARISON; we rewrite the algorithm here for our more general preference statements:

Algorithm : $\succeq$-comparison. Input: o, o', φ

1. Repeat:
   (a) guess o'', $\alpha \,|\, V : w \geq w' \in \varphi$; $Y \leftarrow \mathcal{X} - (U \cup V \cup W)$;
   (b) if $\alpha \,|\, V : w \geq w' \in \varphi$ sanctions (o, o''): $o \leftarrow o''$;
   until $o'' = o'$.

This algorithm only needs space to store two outcomes at any iteration, and checks sanctioning w.r.t. one rule at every iteration. Repeated applications of this algorithm can answer R-COMPARISON queries for $R \in \{\succ, \sim, \bowtie\}$; for instance, to check if $o \bowtie_\varphi o'$, we check that $o \succeq_\varphi o'$ does not hold and that $o' \succeq_\varphi o$ does not hold either. [12]

Tractability of comparisons, except in some trivial cases, comes at a heavy price in terms of expressiveness: the only positive result for $\succeq$-COMPARISON is about CP-nets when the dependency graph is a polytree [5, Theorem 14]; clearly, this entails a positive results for the other comparison queries for this language. The next proposition shows that most comparisons are hard for a vast family of CP statements; it follows from hardness results proved by [5] and [23].

Proposition 9. *R-COMPARISON for $R \in \{\succ, \succeq, \bowtie\}$ is NP hard for the language of fully acyclic CP-nets. $\sim$-COMPARISON is trivial for linearisable CP formulas, but hard for $\mathsf{CP1}\not\triangleright\wedge$, and for the language of linearisable (hence locally consistent) and locally complete $\mathsf{CP1}\not\triangleright$ formulas.*

Optimisation Comparison queries can be used to compute, in a given set $S \subseteq \underline{\mathcal{X}}$, an alternative that is not dominated by any other alternative in S: this can be achieved by asking at most $|S|(|S|-1)/2$ dominance queries (such query can return failure when S contains no such alternative). More generally, given some integer k, we may be interested in finding a subset S' of S that contains k "best" alternatives of S, in the following sense: we say that $S' \subseteq S$ is *weakly undominated* in S if for every $o \in S'$, for every $o' \in S \setminus S'$ it is not the case that $o' \succ_\varphi o$. Note that such a set must exist, because $\succ_\varphi$ is acyclic. [31] proposes a stronger query:

ORDERING Given k, $S \subseteq \underline{\mathcal{X}}$ and φ, find $o_1, o_2, \ldots, o_k \in S$ such that for every $i \in 1, \ldots, k$, for every $o' \in S$, if $o' \succ_\varphi o$ then $o' \in \{o_1, \ldots, o_i\}$.

Note that if $o_1, o_2, \ldots, o_k$ is the answer to such query, if $1 \leq i < j \leq k$, then it can be the case that $o_i \bowtie o_j$, but it is guaranteed that $o_j \not\succ o_i$: in the context of a recommender system for instance, where one would expect alternatives to be presented in order of non-increasing preference, o_i could be safely presented before o_j.

[5] consider a specific case of the above query, when $|S| = 2$: note that given two alternatives $o, o' \in \underline{\mathcal{X}}$ it must be the case that at least one of $o \succ_\varphi o'$ or $o' \succ_\varphi o$ must be false, since $\succ_\varphi$ is irreflexive and transitive.

[12] Recall that NPSPACE = co-NPSPACE = PSPACE .

2-ORDERING Given $o, o' \in \underline{\mathcal{X}}$, return a pair $(o_1, o_2) \in \{(o, o'), (o', o)\}$ such that $o_2 \not\succ_\varphi o_1$.

[5] prove that 2-ORDERING is tractable for acyclic CP-nets. This result can be generalised to cuc-acyclic formulas of CP1$\wedge$:

Proposition 10 (Generalisation of Theorem 5 in [5])**.** 2-ORDERING *and* ORDERING *can be answered in time which is polynomial in the size of φ and the size of S for cuc-acyclic, locally consistent formulas of CP1$\wedge$; and for LPT.*

This approach for optimisation is of course not practical when $S = \underline{\mathcal{X}}$, because in this case the size of S is exponential in the number of attributes. When k is fixed, $k = 1$ and $S = \underline{\mathcal{X}}$, the ORDERING query amounts to finding a weakly undominated alternative. Based on the notions of (weakly) undominated / (strongly) dominating alternatives (defined in section 2.2.1), [23] define two types of queries: 1) given φ and o, is o (weakly) undominated, or is it (strongly) dominated? 2) given φ, is there a (weakly) undominated, or a (strongly) dominated, alternative? We call these queries W-UNDOMINATED CHECK, UNDOMINATED CHECK, and so on, for queries of type 1); and W-UNDOMINATED-$\exists$, and so on, for queries of type 2). Note also that there is always at least one weakly undominated alternative (because $\underline{\mathcal{X}}$ is finite), so WEAKLY UNDOMINATED-$\exists$ is trivial (always true).

All these queries are easily shown to be tractable for LPT. The problem UNDOMINATED CHECK is tractable for CP; in fact, the proof, originally given by [5] for CP-nets, and generalised to CP1$\not\rhd$ by [23], can be generalised further to CP.

Proposition 11. UNDOMINATED CHECK *is in P for CP.*

That all other, dominance related, queries are in PSPACE for CP can be proved using again the algorithm for checking $\succeq$-comparison. Checking for instance that o is *not* undominated can be done by guessing some o' and checking if $o' \succeq_\varphi o$. [23] prove several hardness results:

Proposition 12. *[23] The problems* W. UNDOMINATED CHECK, DOMINATING CHECK, S. DOMINATING CHECK *are PSPACE complete for CP1$\not\rhd\wedge$. These problems, as well as* DOMINATING $\exists$, S. DOMINATING $\exists$*, are PSPACE complete for CP1$\not\rhd$, even if restricted to locally consistent and locally complete formulas.* UNDOMINATED $\exists$ *is NP complete for CP1$\not\rhd\wedge$.*

For CP-nets, [5] give a polytime algorithm that computes the only dominating alternative when the dependency graph is acyclic; in this case, this alternative is also the only strongly dominating one, the only undominated, and the only weakly undominated one, since the CP-net is linearisable.

7 Conclusion

We have not studied here transformations, like conditioning or other forms of projection for instance. Some initial results on projections can be found in [2]. Note that the result of transformations like conditionning for CP1 formulas if often not expressible in CP1. However, this preliminary study shows that, for conditional preference statements, gains in terms of query complexity is not only at the cost of a loss in succinctness, but often at the cost of big losses in expressiveness. This may indicate that the language of conditional preference statement is not an adequate target language for compilation, but that other languages may be more suitable for that. However, existing real-valued languages in general force a complete ordering of the alternatives, thus a target language for the compact representation of possibly incomplete preorders has yet to be defined, possibly using combinations of real-valued formulas, used as multiple criteria, as in the definition of "partial order rationalizable" choice functions by [1]; or as approximation of the preorder represented by a set of CP statements.

Acknowledgments We thank Hélène Fargier for fruitful discussions, anonymous reviewers for their helpful comments and suggestions. This work has benefitted from the AI Interdisciplinary Institute ANITI. ANITI is funded by the French "Investing for the Future – PIA3" program under grant agreement n° ANR-19-PI3A-0004. This work has also been supported by the PING/ACK project of the French National Agency for Researchgrant agreement n° ANR-18-CE40-0011.

References

[1] F. Aleskerov, D. Bouyssou, and B. Monjardet. *Utility Maximization, Choice and Preference*. Springer-Verlag Berlin Heidelberg, 2nd edition, 2007.

[2] P. Besnard, J. Lang, and P. Marquis. Variable forgetting in preference relations over combinatorial domains. In *Proc. IJCAI Multidisciplinary Workshop on Advances in Preference Handling*, 2005.

[3] R. Booth, Y. Chevaleyre, J. Lang, J. Mengin, and Chattrakul Sombattheera. Learning various classes of models of lexicographic orderings. Tech. report IRIT/RR–2009-21–FR, IRIT, Toulouse, 2009.

[4] R. Booth, Y. Chevaleyre, J. Lang, J. Mengin, and C. Sombattheera. Learning conditionally lexicographic preference relations. In Proc. ECAI 2010, pages 269–274. IOS Press, 2010.

[5] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. CP-nets: a tool for representing

and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.

[6] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. Preference-based constrained optimization with cp-nets. *Computational Intelligence*, 20(2):137–157, 2004.

[7] C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole. Reasoning with conditional ceteris paribus preference statements. In Proc. UAI 99, pages 71–80. Morgan Kaufmann, 1999.

[8] R. I. Brafman, C. Domshlak, and S. E. Shimony. On graphical modeling of preference and importance. *Journal of Artificial Intelligence Research*, 25:389–424, 2006.

[9] R. I. Brafman, F. Rossi, D. Salvagnin, K. B. Venable, and Toby Walsh. Finding the next solution in constraint- and preference-based knowledge representation formalisms. In Proc. KR'10. 2010.

[10] D. Braziunas and C. Boutilier. Local utility elicitation in gai models. In Proc. UAI'05, pages 42–49. 2005.

[11] M. Bräuning and E. Hüllermeyer. Learning conditional lexicographic preference trees. In Proc. ECAI 2012 workshop on Preference Learning: Problems and Applications in AI, pages 11–15, 2012.

[12] M. Cadoli, F. M. Donini, P. Liberatore, and M. Schaerf. Space efficiency of propositional knowledge representation formalisms. *Journal of Artificial Intelligence Research*, 13:1–31, 2000.

[13] S. Coste-Marquis, J. Lang, P. Liberatore, and P. Marquis. Expressive power and succinctness of propositional languages for preference representation. In Proc. KR'04, pages 203–212. AAAI Press, 2004.

[14] A. Darwiche. Compiling knowledge into decomposable negation normal form. In Proc. IJCAI 99, pages 284–289. Morgan Kaufmann, 1999.

[15] A. Darwiche and P. Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[16] C. Domshlak and R. I. Brafman. CP-nets: Reasoning and consistency testing. In Proc. KR'02, pages 121–132. Morgan Kaufmann, 2002.

[17] Domshlak, C. , Prestwich, S. , Rossi, F. , Venable, K. , Walsh, T. Hard and soft constraints for reasoning about qualitative conditional preferences J. Heuristics 12(4-5), 2006, 263–285

[18] H. Fargier, P. F. Gimenez, and J. Mengin. Learning lexicographic preference trees from positive examples. In Proc. AAAI'18, pages 2959–2966. 2018.

[19] H. Fargier, P. Marquis, A. Niveau, and N. Schmidt. A knowledge compilation map for ordered real-valued decision diagrams. In Proc. AAAI'14, pages 1049–1055. AAAI Press, 2014.

[20] E. C. Freuder, R. Heffernan, R. J. Wallace, and N. Wilson. Lexicographically-ordered constraint satisfaction problems. *Constraints*, 15(1):1–28, 2010.

[21] G. Gigerenzer and D. G. Goldstein. Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review*, 103(4):650–669, 1996.

[22] G. Gogic, H. A. Kautz, C. H. Papadimitriou, and B. Selman. The comparative linguistics of knowledge representation. In Proc. IJCAI'95, pages 862–869. Morgan Kaufmann, 1995.

[23] J., Jérôme Lang, M. Truszczynski, and N. Wilson. The computational complexity of dominance and consistency in CP-nets. *Journal of Artificial Intelligence Research*, 33:403–432, 2008.

[24] C. Gonzales and P. Perny. Gai networks for utility elicitation. In Proc. KR'04, pages 224–233. AAAI Press, 2004.

[25] J. Lang, J. Mengin, and L. Xia. Voting on multi-issue domains with conditionally lexicographic preferences. *Artificial Intelligence*, 265:18–44, 2018.

[26] T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In Proc. IJCAI'95, pages 631–639. Morgan Kaufmann, 1995.

[27] M. Schmitt and L. Martignon. On the complexity of learning lexicographic strategies. *Journal of Machine Learning Research*, 7:55–83, 2006.

[28] N. Wilson. Consistency and constrained optimisation for conditional preferences. In Proc. ECAI'04, pages 888–892. IOS Press, 2004.

[29] N. Wilson. Extending cp-nets with stronger conditional preference statements. In Proc. AAAI'04, pages 735–741. AAAI Press / The MIT Press, 2004.

[30] N. Wilson. An effcient upper approximation for conditional preference. In Proc. ECAI'06. IOS Press, 2006.

[31] N. Wilson. Computational techniques for a simple theory of conditional preferences. *Artificial Intelligence*, 175:1053–1091, 2011.