# An extended Knowledge Compilation Map for Conditional Preference Statements-based and Generalized Additive Utilities-based Languages

Hélène Fargier, Stefan Mengel, Jérôme Mengin

# An extended Knowledge Compilation Map for Conditional Preference Statements-based and Generalized Additive Utilities-based Languages

## Research Report IRIT/RR–2023–03–FR

Hélène Fargier,
Helene.Fargier@irit.fr,
IRIT, Université Paul Sabatier, CNRS, 118 route de Narbonne, 31062 Toulouse Cedex 9, France

Stefan Mengel,
mengel@cril-lab.fr,
Université d'Artois, CNRS, Centre de Recherche en Informatique de Lens (CRIL), Lens, France

Jérôme Mengin,
Jerome.Mengin@irit.fr,
IRIT, Université Paul Sabatier, CNRS, 118 route de Narbonne, 31062 Toulouse Cedex 9, France

December 1st, 2023

**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, and transformations, like conditioning and variable elimination, thereby contributing to a knowledge compilation map for languages based on conditional preference statements. We also study the expressiveness and complexity of queries and transformations for generalized additive utilities, and introduce a new parameterized family of languages, which enables to balance expressiveness against the complexity of some queries. This paper is an extended version of [29] – in addition to the results of [29], it contains a study of several transformations (Section 7). We have also added the GAI language to the map.

# 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 [45], additive utilities and their generalizations [37, 15]. Ordinal models like CP nets and their generalizations [10, 49, 14], or lexicographic preferences and their generalizations [34, 46, 50, 9, 17, 30] 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 NP-hard for many of these models, and in fact even PSPACE-hard for some of them, 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 tackling this problem is Knowledge Compilation, which is a general approach in which 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: [23, 24] compare how various subsets of propositional logic can succinctly, or not, express propositional knowledge bases, and the complexity of queries of interest. [21] follow a similar approach to compare extensions of propositional logic which associate real values to models of a knowledge base; [31] consider value function-based models.

The aim of this paper is to initiate a *compilation map* for representations on preferences. To this end, we systematically study and compare different languages of conditional preference statements and models based on Generalized Additive Utilities (called GAIs). In particular, we analyze the expressiveness and succinctness of various languages based on these conditional preference statements and on GAIs, and the complexity of several queries and transformations of interest.

Section 2 recalls some basic definitions about combinatorial domains and pre-orders, and introduces notation that we will use throughout. Section 3 gives an overview of various languages based on conditional preference statements that have been studied in the literature. It introduces first a general language of conditional preference statements, and recalls the language of Generalized Additive Utilities. The remainder of this section then presents various language restrictions that have been studied in the literature and offer interesting compromises between expressiveness and querying complexity.

Section 4 and 5 respectively study expressiveness and succinctness for the languages we study. Sections 6 and 7 study the complexity of, respectively, queries and transformations for these languages.

This paper is an extended version of [29] – in addition to the results of [29], it contains a study of several transformations (Section 7). We have also added the GAI language to the map. (Unpublished) proofs are provided in the appendix.

# 2 Preliminaries

## 2.1 Combinatorial Domains

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 characterize the possible alternatives, each attribute $X \in \mathcal{X}$ having a finite set of possible values $\underline{X}$; we assume $|\underline{X}| \geq 2$ for every $X \in \mathcal{X}$; then $\underline{\mathcal{X}}$ denotes the Cartesian product of the domains of the attributes in $\mathcal{X}$, its elements are called alternatives. For a binary attribute $X$, we will often denote by $x, \bar{x}$ its two possible values. In the sequel, $n$ is the number of attributes in $\mathcal{X}$.

For a subset $U$ of $\mathcal{X}$, we will denote by $\underline{U}$ the Cartesian product of the domains of the attributes in $U$. The elements of $\underline{U}$ are called 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 atoms 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); we denote by $\top$ (resp. $\bot$) the formula always true (resp. false). 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 $\alpha$ is such a propositional formula over $\mathcal{X}$ and $o \in \underline{\mathcal{X}}$, we will write $o \models \alpha$ when $o$ satisfies $\alpha$, that is when, assigning to every literal $X = x$ that appears in $\alpha$ the value true if $o[X] = x$, and the value false otherwise, makes $\alpha$ true.

Given a formula $\alpha$, or a partial instantiation $u$, $\mathsf{Var}(\alpha)$ and $\mathsf{Var}(u)$ denote the set of attributes, the values of which appear in $\alpha$ 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

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 decision maker's preferences can be characterized by a preorder $\succeq$ over $\underline{\mathcal{X}}$, that is $\succeq$ is a binary, reflexive and transitive relation. For alternatives $o, o'$, we 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 $o \bowtie o'$ holds.

The relation $\sim$ defined in this way is called the *symmetric part* of $\succeq$; it is symmetric, reflexive and transitive. The relation $\bowtie$ is symmetric and irreflexive. The relation $\succ$ is called the *asymmetric part* of $\succeq$, and is what is usually called a strict partial order, i.e., it is irreflexive, transitive and asymmetric.

When the preorder $\succeq$ is complete, that is, when it is the case that $o \succeq o'$ or $o' \succeq o$ for every pair of alternatives $(o, o')$, it is called a *weak order*. A strict partial order that is complete is called a *linear order*.

When the preorder $\succeq$ is antisymmetric, that is when $o \sim o'$ only when $o = o'$, then it is called a *partial order*.

**Terminology and notation** We say that an alternative $o$ *dominates* an 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 notation 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 $o \succ o'$. Given two preorders $\succeq$ and $\succeq'$, we say that $\succeq$ *extends* $\succeq'$ when $o \succeq' o$ implies $o \succeq o'$, for every pair of alternatives $o, o'$.

# 3 Languages

## 3.1 Conditional Preference Statements

A *conditional preference statement* (short *CP statement*) over $\mathcal{X}$ is an expression of the form $\alpha \mid V : w \geq w'$, where $\alpha$ is a propositional formula over $U \subseteq \mathcal{X}$, $w, w' \in \underline{W}$ are such that $w[X] \neq w'[X]$ for every $X \in W$, and $U, V, W$ are disjoint subsets of $\mathcal{X}$, not necessarily forming a partition of $\mathcal{X}$. Informally, such a statement represents the piece of knowledge that, when comparing alternatives $o, o'$ that both satisfy $\alpha$, 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 attribute in $\mathcal{X} \setminus (V \cup W)$ being fixed. We call $\alpha$ the conditioning part of the statement; we call $W$ the swapped attributes, and $V$ the free part.

**Example 1** ((Example A in [51], slightly extended)). *Consider planning a holiday, with three choices / attributes: wait until next month ($W = w$) or leave now ($W = \bar{w}$), going to city 1, 2 or 3 ($C = c_1$, $C = c_2$ or $C = c_3$), travelling by plane ($P = p$) or by car ($P = \bar{p}$). I would rather go now, irrespective of the other attributes: $\top \mid \{CP\} : \bar{w} \geq w$. All else being equal, I prefer to go to city 3, city 1 being my second best choice: $\top \mid \emptyset : c_3 \geq c_1 \geq c_2$. Also, if I go now, I prefer to fly: $\bar{w} \mid \emptyset : p \geq \bar{p}$. Together, the last two statements imply that if I go now, I prefer to go to city 3 by plane than go to city 1 by car; however these statements do not say what I prefer between flying to city 1 or driving to city 3. In fact, I prefer the former, this tradeoff can be expressed with the statement $\bar{w} \mid \emptyset : c_1 p \geq c_3 \bar{p}$. Finally, if I go later, I prefer to drive, irrespective of the city: $w \mid \{C\} : \bar{p} \geq p$.*

Conditional preference statements have been studied in many works, under various language restrictions. They are the basis for CP-nets [12, 10] and their extensions, and have been studied in a more logic-based fashion by e.g. [36, 49, 48, 51]. [1] Closely related to them are the *Conditional Importance statements* studied in [13].

For the semantics of sets of CP statements, we use the definitions of [51]. Given a statement $\alpha \mid V : w \geq w'$, let $U = \mathsf{Var}(\alpha)$ and $W = \mathsf{Var}(w) = \mathsf{Var}(w')$: a *worsening swap* is any pair of alternatives $(o, o')$ such that $o[U] = o'[U] \models \alpha$, $o[W] = w$ and $o'[W] = w'$, and such that for every attribute $Y \notin U \cup V \cup W$ it holds that $o[Y] = o'[Y]$; we say that $\alpha \mid V : w \geq w'$ *sanctions* $(o, o')$. For a set of CP-statements $\varphi$, let $\varphi^*$ be the set of all worsening swaps sanctioned by statements of $\varphi$, and define $\succeq_\varphi$ to be the reflexive and transitive closure of $\varphi^*$. [51] proves that $o \succeq_\varphi o'$ holds if and only if $o = o'$ holds or $\varphi^*$ 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]

**Example 2** (Example 1, continued). *Let*

$$\varphi = \left\{ \begin{array}{c} \top \mid \{CP\} : \bar{w} \geq w \ , \ \top \mid \emptyset : c_3 \geq c_1 \geq c_2, \\ \bar{w} \mid \emptyset : p \geq \bar{p} \ , \ \bar{w} \mid \emptyset : c_1 p \geq c_3 \bar{p} \ , \ w \mid \{C\} : \bar{p} \geq p \} \end{array} \right\}.$$

*Then $\top \mid \{CP\} : \bar{w} \geq w$ sanctions for instance $(\bar{w}c_2p, wc_3\bar{p})$, so $\bar{w}c_2p \succeq_\varphi wc_3\bar{p}$. Also, $\top \mid \emptyset : c_3 \geq c_1 \geq c_2$ sanctions $(\bar{w}c_1p, \bar{w}c_2p)$, $\bar{w} \mid \emptyset : p \geq \bar{p}$ sanctions $(\bar{w}c_2p, \bar{w}c_2\bar{p})$, so, by transitivity, $\bar{w}c_1p \succeq_\varphi \bar{w}c_2\bar{p}$. It is not difficult to check that $\bar{w}c_2p \bowtie_\varphi \bar{w}c_1\bar{p}$.*

[1] The formula $u \mid V : x \geq x'$ is written $u : x > x'[V]$ by [51].

[2] Actually, [51] proves that $(o, o')$ is in the transitive closure of $\varphi^*$ 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.

Let us call CP the language where formulas are sets of statements of the general form $\alpha \mid V : w \geq w'$. This language is very expressive: it is possible to represent any preorder "in extension" with preference statements of the form $o \geq o'$ – they have $W = \{X \mid o[X] \neq o'[X]\}$ as set of swapped attributes, $\alpha = o[U] = o'[U]$ as condition where $U = \{X \mid o[X] = o'[X]\}$, and no free attribute.

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 / sublanguages have been studied in the literature, we review them below.

**(Strict) Consistency** Although the original definition of CP-nets by [12] does not impose it, many works on CP-nets, especially following [10], consider that they are intended to represent a strict partial order, that is, that $\succeq_\varphi$ should be antisymmetric. We say that a set $\varphi$ of CP-statements is *consistent* in this case. Note that in this case, for two different alternatives $o$ and $o'$, $o \succeq_\varphi o'$ implies that $o \succ_\varphi o'$.

**Notation** We write $\alpha : w \geq w'$ when $V$ is empty, and $w \geq w'$ when $V$ is empty and $\alpha = \top$. 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 present various sublanguages of CP. Some are defined by imposing various simple syntactical restrictions on the formulas, two are languages which have been well studied (CP-nets and lexicographic preference trees).

## 3.2 Statement-wise Restrictions

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

$\mathcal{L}\not\triangleright$ = only formulas with empty free parts ($V = \emptyset$) for every statement;[3]

$\mathcal{L}\wedge$ = only formulas where the condition $\alpha$ of every statement is a conjunction of literals;

**k-$\mathcal{L}$** = only formulas where the set of swapped attributes contains no more than $k$ attributes ($\mid W \mid \leq k$) for every statement; in particular, we call elements of 1-CP *unary* statements.

In particular, 1-CP$\wedge$ is the language studied by [51], and 1-CP$\not\triangleright$ is the language of generalized CP-nets as defined by [36].

## 3.3 Graphical Restrictions

Given $\varphi \in$ CP over set of attributes $\mathcal{X}$, we define $D_\varphi$ as the graph with sets of vertices $\mathcal{X}$, and such that there is an edge $(X, Y)$ if there is $\alpha \mid V : w \geq w' \in \varphi$ such that $X \in \mathsf{Var}(\alpha)$ and $Y \in \mathsf{Var}(w)$, or $X \in \mathsf{Var}(w)$ and $Y \in V$. We call $D_\varphi$ the *dependency graph* of $\varphi$. Note that $D_\varphi$ can be computed in polynomial time. This definition, inspired by [51, Def. 15], generalizes that of [10], which is restricted to the case where all CP statements are unary and have no free attributes, and that of [14], who study statements with free attributes. Many tractability results on sets of CP statements have been obtained when $D_\varphi$ has good properties. Given some language $\mathcal{L} \subseteq$ CP, we define:

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

$\mathcal{L}\not\circlearrowleft^{\mathbf{poly}}$ = the restriction of $\mathcal{L}$ to formulas where the dependency graph is a polytree.

## 3.4 CP-nets

In their seminal work, [10] 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;[5]
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 $\mid \underline{U} \mid$ rules $u : \geq$, for every $u \in \underline{U}$, where the $\geq$'s are linear orders over $\underline{X}$.

[3] In the literature, the symbol $\triangleright$ is sometimes used to represent an *importance* relation between attributes; and, as explained by [51], statement $\alpha \mid V : w \geq w'$ is a way to express that attributes in $\mathsf{Var}(w)$ are more important than those in $V$ (when $\alpha$ is true).

[4] This is *full acyclicity* in [51].

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

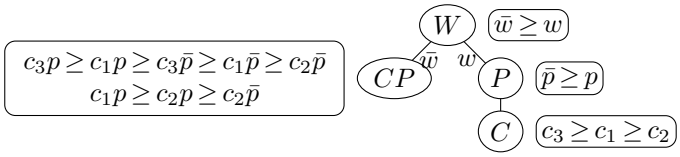



Figure 1: An LP-tree equivalent to the set of CP-statements of Example 2.

Therefore, as shown by [51], 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 $u \in \underline{U}$ where $U$ is the set of parents of $X$ in the graph, every $x, x' \in \underline{X}$ such that $x, x'$ are consecutive values in the linear order $\geq$ specified by the rule $u : \geq$ of $\mathcal{N}$. Then the dependency graph of $\varphi_{\mathcal{N}}$, as defined in Section 3.3, coincides with the graph of $\mathcal{N}$. We call

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

Note that CPnet $\subseteq$ 1-CP$\wedge\not\rhd$. For a given $\varphi \in$ 1-CP$\wedge\not\rhd$, being a CP-net necessitates a very strong form of local consistency and completeness: for every attribute $X$ with parents $U$ in $D_\varphi$, for every $u \in \underline{U}$, for every $x, x' \in \underline{X}$, $\varphi$ must explicitly, and uniquely, order $ux$ and $ux'$.

[14] define TCP-nets as an extension of CP-nets where it is possible to represent tradeoffs, by stating that, under some conditions, some attributes are more important than other ones. [51] describes how TCP-nets can be transformed, in polynomial time, into equivalent sets of 1-CP$\wedge$ statements.

## 3.5 Lexicographic Preference Trees

LP-trees generalize lexicographic orders, which have been widely studied in decision making – see e.g. [32]. As an inference mechanism, they are equivalent to search trees used by [11], and formalized by [48, 51]. As a preference representation, and elicitation, language, slightly different definitions for LP-trees have been proposed by [9, 17, 30]. We use here a definition which subsumes the others.

An LP-tree that is equivalent to the set of CP-statements of Example 2 is depicted on Figure 1. More generally, 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 of attributes, denoted $\mathsf{Var}(N)$;
- if $N$ is not a leaf, it can have one child, or $|\underline{\mathsf{Var}(N)}|$ children;
- in the latter case, the edges that connect $N$ to its children are labelled with the instantiations in $\underline{\mathsf{Var}(N)}$;
- if $N$ has one child only, the edge that connects $N$ to its child is not labelled: all instantiations in $\underline{\mathsf{Var}(N)}$ lead to the same subtree;
- we denote by $\mathsf{Anc}(N)$ the set of attributes that appear in the nodes between the root and $N$ (excluding those at $N$), and by $\mathsf{Inst}(N)$ (resp. $\mathsf{NonInst}(N)$) the set of attributes that appear in the nodes above $N$ that have more than one child (resp. only one child);
- 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 partial order over $\underline{\mathsf{Var}(N)}$, and $\alpha$ is a propositional formula over some attributes in $\mathsf{NonInst}(N)$.

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

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

Given an LP-tree $\varphi$ 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 either a leaf node is reached, or a node $N$ is reached which is labelled with some $W$ such that $o[W] \neq o'[W]$: in the latter case, we say that $N$ decides $\{o, o'\}$.

In order to define $\succeq_\varphi$ for some LP-tree $\varphi$, let $\varphi^*$ 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 $\varphi^*$. Note that if there is no node that decides $\{o, o'\}$, or if the node that decides that pair is labelled with some $W$ and if the local preference table is such that $o[W]$ and $o'[W]$ are incomparable, then $o \bowtie_\varphi o'$.

**Proposition 1.** *Let $\varphi$ be an LP-tree over $\mathcal{X}$, then $\succeq_\varphi$ as defined above is a partial order. Furthermore, $\succeq_\varphi$ is a linear order if and only if 1) every attribute appears on every branch and 2) every preference rule specifies a linear order.*

An LP-tree $\varphi$ is said to be *complete* if the two conditions in Proposition 1 hold, that is, if $\succeq_\varphi$ is a linear order.

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

- $u$ is the combination of values given to the attributes in $\mathsf{Inst}(N)$ along the edges between the root and $N$, and
- $w, w' \in \underline{W}$ such that $w \geq_N^\alpha w'$, and $W^{\neq}$ is the set of attributes on which $w$ and $w'$ have distinct values; and
- $V = [\mathcal{X} - (\mathsf{Anc}(N) \cup W)]$.

This set of statements indicate that alternatives that agree on $\mathsf{Anc}(N)$ and satisfy $u \wedge \alpha$, but have different values for $\mathsf{Var}(N)$, should be ordered according to $\geq_N^\alpha$, whatever their values for attributes in $V$.

**LPT** = the language of LP-trees as defined above; we consider that LPT is a subset of CP.[6]

Note that, using the notation defined above, k-LPT $=$ LPT $\cap$ $k$-CP is the restriction of LPT where every node has at most $k$ attributes, for every $k \in \mathbf{N}$; in particular, 1-LPT is the language of LP-trees with one attribute at each node; and LPT$\wedge$ $=$ LPT $\cap$ CP$\wedge$ is the restriction of LPT where the condition $\alpha$ in every rule at every node is a conjunction of literals. Search trees of [48, 51] and LP-trees as defined by [9, 40] are sublanguages of 1-LPT$\wedge$; LP-trees of [30] and [17] are sublanguages of LPT$\wedge$.

We also introduce a very restrictive class of LP-trees, which will turn out to have interesting properties when we look at transformations.

**k-LPT$^{\mathsf{lin}}$** = the language that contains all *linear* $k$-LP-trees, that is, LP-trees where every node has at most $k$ variables, at most one child, and where all conditional preference rules are *un*conditional.

Complete, linear 1-LP trees represent the usual lexicographic orderings.

### 3.6 GAI decompositions

We also consider *GAI decompostions* [4, 38]. This framework allows the representation of complete and transitive preference relations by a utility function, additively decomposed as a sum of local utility functions bearing on smaller subsets of attributes. Each local utility function can for instance represent a criterion, the global preference deriving from the additive aggregation of the satisfaction degrees provided by the different criteria.

A GAI decomposition over a set $\mathcal{X}$ of finite attributes is defined by a set $\varphi = \{g_{Z_1}, \ldots, g_{Z_m}\}$ of functions bearing on subsets $Z_i$ of $\mathcal{X}$ and taking their values in $R \cup \{-\infty\}$; for any alternative $o$, let $g_\varphi(o) = \Sigma_{i=1}^m g_{z_i}(o[Z_i])$. The set $\varphi$ represents the complete and transitive relation $\succeq_\varphi$ in which $o \succeq_\varphi o'$ if and only if $g_\varphi(o) \geq g_\varphi(o')$. Thus $\succeq_\varphi$ is a weak order.

The questions related to the succinctness of GAI representations depend on the way the local functions are represented – and so do all the questions related to the complexity of the operations on such representations. It is generally assumed that each $g_{Z_i}$ is represented by a table that associates to each tuple of the domain of $Z_i$ a real valued utility and the tuples not present in the table receive the utility 0.

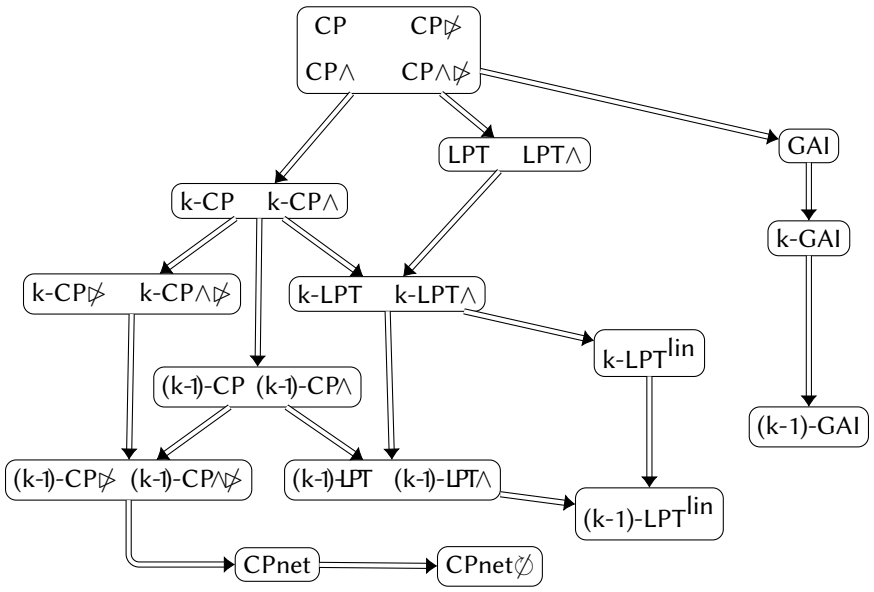
The most common restriction on the language of GAIs consists in bounding by some integer $k > 0$ the maximum number of attributes in a same subutility; we denote by $\mathsf{GAI}_k$ the corresponding language. In particular, $\mathsf{GAI}_1$ is the language of *Additive Utilities*.

## 4 Expressiveness

This section presents our results on the expressiveness of the various languages introduced above. To this end, let us introduce the way in which we compare different languages.

**Definition 1.** *Let $\mathcal{L}$ and $\mathcal{L}'$ be two languages for representing preorders. We say that $\mathcal{L}$ 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 $\mathcal{L}'$. We write $\mathcal{L} \sqsupseteq\!\sqsubseteq \mathcal{L}'$ when the two languages are equally expressive.*

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




$\mathcal{L} \Longrightarrow \mathcal{L}'$: $\mathcal{L}$ is strictly more expressive than $\mathcal{L}'$
Boxes contain languages that are equally expressive.
For $k > 2$.

Figure 2: Relative expressiveness.

We reserve the usual "rounded" symbols $\subset$ and $\subseteq$ for (strict) set inclusion, and $\supset$ and $\supseteq$ for the reverse inclusions. Note that $\sqsupseteq$ is a preorder, and obviously $\mathcal{L} \supseteq \mathcal{L}'$ implies $\mathcal{L} \sqsupseteq \mathcal{L}'$.

Figure 2 gives a summary of the expressiveness results we show in this section. Note that the fact that acyclicity restricts the expressiveness of CP-nets has been shown in e.g. [10].

Let us start exploring the relative expressiveness of different languages. Clearly, CP⊳ $\subset$ CP and CP∧ $\subset$ CP; however, these three languages have the same expressiveness, because of the following:

**Property 2.** *Given some preorder $\succeq$, define*

$$\varphi = \{o[\mathcal{X} - \Delta(o, o')] : o[\Delta(o, o')] \geq o'[\Delta(o, o')] \mid o \succeq o', o \neq o'\},$$

*where $\Delta(o, o')$ is the set of attributes that have different values in $o$ and $o'$, then $\varphi \in$ CP⊳∧, and $\succeq_\varphi = \succeq$.*

A large body of works on CP-statements since the seminal paper by [11] concentrate on various subsets of 1-CP. With this strong restriction on the number of swapped attributes, CP-statements have a reduced expressiveness.

**Example 3.** *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 $\varphi = \{ab \geq \bar{a}\bar{b}, \bar{b} : \bar{a} \geq a, a\bar{b} \geq \bar{a}b\}$. But it cannot be represented in 1-CP: $\{b : a \geq \bar{a}, \bar{b} : \bar{a} \geq a, a : b \geq \bar{b}, \bar{a} : \bar{b} \geq b\}^* \subseteq \varphi^*$, but this is not sufficient to compare $a\bar{b}$ with $\bar{a}b$. The four remaining formulas of 1-CP 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 $\varphi$ yields a preorder which would not be antisymmetric.*

Forbidding free parts incurs an additional loss in expressiveness:

**Example 4.** *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 1-CP with $\varphi = \{B : a \geq \bar{a}, b \geq \bar{b}\}$. But the "tradeoff" $a\bar{b} \succ \bar{a}b$ cannot be represented in 1-CP⊳, any formula of 1-CP⊳ that implies it will put some intermediate alternative between $a\bar{b}$ and $\bar{a}b$*

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

**Proposition 3.** *CP $= \bigcup_{k \in \mathbb{N}}$ k-CP and, for every $k \in \mathbb{N}$, $k \geq 2$:*

$$\textit{CP}\wedge \sqsubsetneq \textit{CP}\rhd \sqsubsetneq \textit{CP}\wedge\rhd \sqsubsetneq \textit{CP} \sqsupset \textit{k-CP} \sqsubsetneq \textit{k-CP}\wedge \sqsupset \textit{k-CP}\rhd \sqsubsetneq \textit{k-CP}\wedge\rhd$$
$$\textit{k-CP} \sqsupset \textit{(k-1)-CP} \quad \textit{and} \quad \textit{k-CP}\wedge\rhd \sqsupset \textit{(k-1)-CP}\wedge\rhd \sqsupset \textit{CPnet}.$$

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 partial order, LPT can represent any partial order. Limiting to conjunctive conditions in the rules is not restrictive. However, restricting to 1-LPT reduces expressiveness, even if one considers formulas of 1-CP that represent total, linear orders:

**Example 5.** *Let $\varphi = \{a \geq \bar{a}, \bar{c} \mid A : \bar{b} \geq b, \bar{a}c : \bar{b} \geq b, ac : b \geq \bar{b}, a : c \geq \bar{c}, \bar{a} \mid B : \bar{c} \geq c\}$. This yields the following linear order: $abc \succeq_\varphi a\bar{b}c \succeq_\varphi a\bar{b}\bar{c} \succeq_\varphi \bar{a}\bar{b}\bar{c} \succeq_\varphi ab\bar{c} \succeq_\varphi \bar{a}b\bar{c} \succeq_\varphi \bar{a}\bar{b}c \succeq_\varphi \bar{a}bc$. No $\psi \in$ 1-LPT can represent it: $A$ could not be at the root of such a tree because for instance $a\bar{b}\bar{c} \succeq_\varphi \bar{a}\bar{b}\bar{c}$ and $\bar{a}\bar{b}\bar{c} \succeq_\varphi ab\bar{c}$; neither could $C$, since $a\bar{b}c \succeq_\varphi a\bar{b}\bar{c}$ and $\bar{a}b\bar{c} \succeq_\varphi \bar{a}\bar{b}\bar{c}$; and finally $B$ could not be at the root either, because $abc \succeq_\varphi \bar{a}bc$ and $\bar{a}\bar{b}\bar{c} \succeq_\varphi ab\bar{c}$.*

**Proposition 4.** *LPT $= \bigcup_{k\in\mathbb{N}}$ k-LPT and, for every $k \in \mathbb{N}$:*

$$\text{CP} \supset \text{LPT} \sqsupset \text{LPT}\wedge \supset \text{k-LPT} \sqsupset \text{k-LPT}\wedge \supset \text{(k-1)-LPT}$$
$$\text{k-CP} \supset \text{k-LPT} \supset \text{k-LPT}^{lin} \supset \text{(k-1)-LPT}^{lin}.$$

Finally, because GAI decompositions are restricted to the representation of complete preference relations, their expressiveness is lower than the one of the general CP language; the latter can represent any transitive relation, so CP is strictly more expressive than GAI. Subclasses of the CP language may be incomparable with GAI. The same line of reasoning applies when comparing GAI and complete lexicographic trees: both target the representation of complete orders, but the former language allows the representation of any complete preorder, while the latter can represent linear orders only (antisymmetry is required). It follows that GAI are strictly more expressive than complete LP trees. We summarize these observations below.

**Proposition 5.** *CP $\supset$ GAI $\supset$ complete-LPT.*

The second source of limitations on expressiveness comes from the bounding of the number of attributes present in the expression of local preferences. Using the same counter example as those used for showing that CP is strictly more expressive than $k$-CP, one can show that GAI and $k$-CP restrictions are incomparable in terms of expressiveness.

**Proposition 6.** *For every $k \in \mathbb{N}$: GAI$_{k+1} \supset$ GAI$_k$, and (k-1)-CP $\not\supset$ GAI$_k$.*

# 5 Succinctness

Another criterion is the relative sizes of formulas that can represent the same preorder in different languages. This section details our results about the succinctness of the various languages introduced above.

Cadoli et al. [18] study the space efficiency of various propositional knowledge representation formalisms. An often used definition of succinctness [35, 24] makes it a particular case of expressiveness, which is not a problem when comparing languages of same expressiveness. However, we study here languages with very different expressiveness, so we need a more fine grained definition:

**Definition 2.** *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 $\varphi' \in \mathcal{L}'$, there exists $\varphi \in \mathcal{L}$ that represents the same preorder as $\varphi'$ and such that $|\varphi| < p(|\varphi'|)$.*[7] *Moreover, 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 of $\mathcal{L}'$ has an equivalent formula in $\mathcal{L}$ which is "no bigger" (up to some polynomial transformation of the size of $\varphi$), and there is at least one sequence of formulas (one formula for every polynomial $p$) in $\mathcal{L}$ that have equivalent formulas in $\mathcal{L}'$ but necessarily "much bigger".[8]

**Proposition 7.** *The following hold, for languages $\mathcal{L}$, $\mathcal{L}'$, $\mathcal{L}''$:*

- *if $\mathcal{L} \supseteq \mathcal{L}'$ then $\mathcal{L} \leqq \mathcal{L}'$; and if $\mathcal{L} \leqq \mathcal{L}'$, then $\mathcal{L} \sqsupseteq \mathcal{L}'$;*
- *if $\mathcal{L} \ll \mathcal{L}'$ then $\mathcal{L} \leqq \mathcal{L}'$ and $\mathcal{L}' \not\leqq \mathcal{L}$;*
- *if $\mathcal{L} \sqsubset \mathcal{L}'$, the reverse implication holds:*

  *if $\mathcal{L} \leqq \mathcal{L}'$ and $\mathcal{L}' \not\leqq \mathcal{L}$ then $\mathcal{L} \ll \mathcal{L}'$*

  *(otherwise, it might be that $\mathcal{L}' \not\leqq \mathcal{L}$ because $\mathcal{L}' \not\supseteq \mathcal{L}$);*
- *if $\mathcal{L} \supseteq \mathcal{L}'$ and $\mathcal{L}' \ll \mathcal{L}''$, then $\mathcal{L} \ll \mathcal{L}''$.*

Restricting the conditioning part of CP statements to be conjunctions of literals leads to a loss in succinctness.

[7] Where $|\varphi| = \sum_{\alpha \mid V : w \geq w' \in \varphi} (|\alpha| + |V| + 2|\text{Var}(w)|)$, with $|\alpha|$ = the number of connectives plus the number of atoms of $\alpha$.

[8] When $\ll$ is defined as the strict counterpart of $\leqq$, it can happen that $\mathcal{L} \ll \mathcal{L}'$ even if there is no real difference in representation size in the two languages, but $\mathcal{L} \sqsupset \mathcal{L}'$.

| | GAI | $GAI_k$ | $GAI_1$ | CP | 1-CP$\not\bowtie$ | 1-CP$\not\bowtie\wedge$ | CPnet | CPnet$\not\emptyset$ | CPnet$\not\emptyset$poly | LPT |
|---|---|---|---|---|---|---|---|---|---|---|
| CONSISTENCY | | | | ✘✘ | ✘✘ | ✘✘ | | ⊤ | ⊤ | ⊤ |
| R-COMPARISON, $R \in \{\succeq, \succ, \bowtie\}$ | ✓ | ✓ | ✓ | ✘✘ | ✘✘ | ✘✘ | ✘○ | ✘ | ✓ | ✓ |
| $\sim$-COMPARISON | ✓ | ✓ | ✓ | ✘✘ | ✘✘ | ✘✘ | | ⊥ | ⊥ | ✓ |
| EQUIVALENCE | | | | ✘✘ | ✘○ | ✘○ | | ✓ | ✓ | ✘○ |
| TOP-$p$ | ✓ | ✓ | ✓ | | | | | ✓ | ✓ | ✓ |
| UNDOMINATED CHECK | ✘ | ✘ | ✓ | ✘✘ | ✘✘ | ✘✘ | | ✓ | ✓ | ✓ |
| UNDOMINATED EXTRACT | ✘○ | ✘○ | ✓ | | | | | ✓ | ✓ | ✓ |
| $\succeq$-CUT EXTRACTION | ✘ | ✘ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $\succ$-CUT EXTRACTION | ✘ | ✘ | ✓ | ✘✘ | ✘✘ | ✘✘ | | ✓ | ✓ | ✓ |
| $\succ$-CUT COUNTING | #✘ | #✘ | #✘ | ✘✘○ | ✘✘○ | ✘✘○ | #✘○ | #✘○ | | ✓ |

Each column corresponds to one sublanguage of CP. They are sorted in order of decreasing expressiveness from left to right, except when columns are separated by double lines. For each query and sublanguage: $\top$ = always true for the language; $\bot$ = always false for the language; ✓ = polytime answer; ✘ = NP/coNP-complete query; ✘○ = NP/coNP-hard query; #✘= #P-complete query; #✘○= #P-hard query; ✘✘ = PSPACE-complete query; ✘✘○ = PSPACE-hard query.

Table 1: Complexity of queries.

**Example 6.** *Consider $2n+1$ binary attributes $X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_n, Z$, and let $\varphi$ 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 $\varphi \in$ 1-CP$\not\bowtie$, but $\varphi$ is not in conjunctive form. A set of conjunctive CP-statements equivalent to $\varphi$ 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 $\mu_i = y_i$ for every $i$.*

Also, free attributes enable the succinct representation of the relative importance of some attributes over others; disabling free attributes thus incurs a loss in succinctness:

**Example 7.** *Consider $n+1$ binary attributes $X_1, X_2, \ldots, X_n, Y$, let $U = \{X_1, X_2, \ldots, X_n\}$, and let $\varphi = \{U \mid y \geq \bar{y}\}$. Then $\varphi^* = \{(uy, u'\bar{y}) \mid u, u' \in \underline{U}\}$, and $\varphi^*$ is equal to its transitive closure, so, if $o \neq o'$, then $o \succeq_\varphi o'$ if and only if $o[Y] = y$ and $o'[Y] = \bar{y}$. This can be represented, without free attribute, only with formula $\psi$ that contains, for every $V \subseteq U$ and every $v \in \underline{V}$, the statement $vy \geq \bar{v}\bar{y}$, where $\bar{v}$ denotes the tuple obtained by inverting all values of $v$. For every $0 \leq i \leq n$ there are $\binom{n}{i}$ subsets of $V$ of size $i$, with $2^i$ ways to choose $v \in \underline{V}$, thus $\psi$ contains $\sum_0^n \binom{n}{i} 2^i = 3^n$ statements.*

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

**Example 8.** *Consider $n+1$ binary attributes $X_1, X_2, \ldots, X_n, Y$, and let $\varphi$ be the 1-CP$\not\bowtie\wedge$ 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 $\varphi$ is linear in $n$. Because preferences for $Y$ depend on all $X_i$'s, a CP-net equivalent to $\varphi$ will contain, in the table for $Y$, $2^n$ CP statements.*

**Proposition 8.** *The following hold:*

- *$\mathcal{L} \ll \mathcal{L}\wedge$ for every $\mathcal{L}$ such that 1-CP$\not\bowtie \subseteq \mathcal{L} \subseteq$ CP;*
- *$\mathcal{L} \ll \mathcal{L}\not\bowtie$ for every $\mathcal{L}$ such that 1-CP$\wedge \subseteq \mathcal{L} \subseteq$ CP;*
- *1-CP$\not\bowtie\wedge \ll$ CPnet.*

We have seen that any complete preorder, and in particular the preference captured by any complete LP-tree can be represented by a GAI. This representation comes with no increase in size.

**Proposition 9.** *Any complete LPT can be transformed in polytime and space into an equivalent GAI.*

# 6 Queries

Table 1 gives an overview of the tractability of the queries that we study in this section. We begin this section with the two queries that have generated most interest in the literature on CP statements.

## 6.1 Consistency

Knowing that a given $\varphi \in$ CP is consistent (that is, that $\succeq_\varphi$ is antisymmetric) is valuable, as it makes several other queries easier. It also gives some interesting insights into the semantics of $\varphi$. The following query has been addressed in many works on CP statements:

CONSISTENCY Given $\varphi$, is $\varphi$ consistent?

[10] prove that when its dependency graph $D_\varphi$ is acyclic, then a CP-net $\varphi$ is consistent. This result has been extended by [25, 14, 51], who give weaker, sufficient syntactical conditions that guarantee that a locally consistent set of unary, conjunctive CP statements is consistent. [36, Theorem 3 and 4] prove that CONSISTENCY is PSPACE-complete for 1-CP$\not\triangleright\wedge$. We have already seen that the preorder defined by any LP tree is antisymmetric.

## 6.2 Comparing alternatives

A basic question, given a formula $\varphi$ and two alternatives $o, o'$ is: how do $o$ and $o'$ compare, according to $\varphi$? 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 $\varphi$, alternatives $o \neq o'$, 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 down 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 node applies to order them, can both be done in polynomial time. For generalized additive utilites, two alternatives can be compared by computing their utilities, which is tractable.

**Proposition 10.** *$R$-COMPARISON is in P for LPT and for GAI for all relations $R \in \{\succ, \succeq, \sim, \bowtie\}$.*

For CP, tractability of comparisons, except in some trivial cases, comes at a heavy price in terms of expressiveness: $\succeq$-COMPARISON is tractable for CP-nets when the dependency graph is a polytree [10, Theorem 14], but [10, Theorems 15, 16] prove that $\succeq$-COMPARISON is already NP-hard for the quite restrictive language of binary-valued, directed-path singly connected CP-nets, which are acyclic. [36, Prop. 7, Corollary 1] prove that $\succeq$-COMPARISON, $\succ$-COMPARISON, $\bowtie$-COMPARISON and $\sim$-COMPARISON are PSPACE-complete for 1-CP$\not\triangleright\wedge$ and for consistent, locally complete formulas of 1-CP$\not\triangleright$. More precise hardness results for acyclic CP-nets are also shown in [41]. Proposition 11 completes the picture.

**Proposition 11.** *$\succ$-COMPARISON and $\bowtie$-COMPARISON are NP-hard for the language of acyclic CP-nets, and tractable for polytree CP-nets.*

## 6.3 Comparing theories

Checking if two theories yield the same preorder can be useful during the compilation process. We say that two formulas $\varphi$ and $\varphi'$ 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 $\varphi$ and $\varphi'$, are they equivalent?

Consider a formula $\varphi \in$ 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 a language $\mathcal{L} \subseteq$ CP is such that adding the 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 1-CP$\not\triangleright$, 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 9.** *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 $\varphi \equiv \varphi \cup \{\bar{a}b : c_1 \geq c_3\} \equiv \varphi \cup \{b : c_1 \geq c_3\}$: the last equivalence follows from the fact that $a \vee (\bar{a}b) \equiv a \vee b$.*

**Proposition 12.** *EQUIVALENCE is coNP-hard for 1-CP$\not\triangleright\not\wedge\not\emptyset$, and for 1-LPT$\wedge$, both restricted to binary attributes.*

As usual, comparing two formulas is easier for languages where there exists a canonical form. This is the case of acyclic CP-nets, as shown by [39, Lemma 2]; their proof makes it clear that the canonical form of any acyclic CP-net $\varphi$ can be computed in polynomial time. Hence:

**Proposition 13.** *EQUIVALENCE is in P for CP-net.*

## 6.4 Top $p$ alternatives

Given a set of alternatives $S$ and some integer $p$, we may be interested in finding a subset $S'$ of $S$ that contains $p$ "best" alternatives of $S$, in the sense that 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. The TOP-$p$ query is usually defined for totally ordered sets; a definition suited to partial orders is given in [51] (where it is called *ordering*), we adopt this definition here:

TOP-$p$ Given $S \subseteq \underline{\mathcal{X}}$, $p < |S|$, and $\varphi$, find $o_1, o_2, \ldots, o_p \in S$ such that for every $i \in 1, \ldots, p$, for every $o' \in S$, if $o' \succ_\varphi o_i$ then $o' \in \{o_1, \ldots, o_{i-1}\}$.

Note that if $o_1, o_2, \ldots, o_p$ is the answer to such query, if $1 \leq i < j \leq p$, 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 $o_j$.

[10] prove that TOP-$p$ is tractable for acyclic CP-nets for the specific case where $|S| = 2$. More generally, $\succ$-COMPARISON queries can be used to compute an answer to a TOP-$p$ query (by asking $\succ$-COMPARISON queries for every pair of elements of $S$, the number of such pairs being in $\Theta(|S|^2)$). Thus TOP-$p$ is tractable for every language where $\succ$-COMPARISON is tractable; this is the case in particular of GAI and LPT.

## 6.5 Optimization

Instead of ordering a given set, we may want to find a globally optimal alternative. We say that alternative $o$ is *undominated* if there is no $o' \in \underline{\mathcal{X}}$ such that $o' \succ_\varphi o$. [9]

Note that any finite set of alternatives always has at least one undominated alternative. We will consider the following queries:

UNDOMINATED EXTRACT Given $\varphi$, return an undominated alternative.

UNDOMINATED CHECK Given $\varphi$ and an alternative $o$, is $o$ undominated?

These queries are easily shown to be tractable for LPT.

For CP-nets, [10] give a polytime algorithm that computes the only undominated alternative when the dependency graph is acyclic.

[36] prove that UNDOMINATED CHECK is PSPACE-complete for 1-CP$\not\triangleright$, and their reductions for proving hardness indeed yield formulas of 1-CP$\not\triangleright\wedge$.

For $\mathsf{GAI}_1$, extracting an undominated alternative can be performed by separately maximizing the unary utilities; and checking if a given alternative is undominated can be done by comparing its utility to that of an extracted undominated alternative. Undominated extract and undominated check are both NP-hard for $\mathsf{GAI}_2$ and thus for GAI and $\mathsf{GAI}_k$ in the general case. We will see these results in the next subsection where we make several similar constructions (Proposition 18).

## 6.6 Cuts

*Cuts* are sets of alternatives that are at the same "level" with respect to $\succeq$. For rankings defined with real-valued functions, cuts are defined with respect to possible real values. In the case of pre-orders, we define cuts with respect to some alternative $o$: given $\varphi \in \mathsf{CP}$, for any $R \in \{\succ, \succeq\}$, for every alternative $o$, we define

$$\mathsf{CUT}^{R,o}(\varphi) = \{o' \in \underline{\mathcal{X}} \mid o' \neq o, o' R_\varphi o\}.$$

Following [31], we define two families of queries:

$R$-CUT EXTRACTION Given $\varphi, o$, return an element of $\mathsf{CUT}^{R,o}(\varphi)$ (or that it is empty)

$R$-CUT COUNTING Given $\varphi, o$, count the elements of $\mathsf{CUT}^{R,o}(\varphi)$

Note that

**Proposition 14.** *$\succeq$-CUT COUNTING and $\succ$-CUT COUNTING are #P-hard for CP-nets and acyclic CP-nets.*

**Proposition 15.** *$\succ$-CUT COUNTING is #P-complete for GAI, $\mathsf{GAI}_k$ and $\mathsf{GAI}_1$.*

**Proposition 16.** *$\succeq$-CUT EXTRACTION is tractable for CP, and $\succ$-CUT EXTRACTION is tractable for acyclic CP-nets. $\succ$-CUT COUNTING and $\succ$-CUT EXTRACTION are PSPACE-hard for 1-CP$\not\triangleright\wedge$. $\succ$-CUT EXTRACTION, $\succeq$-CUT EXTRACTION and $\succ$-CUT COUNTING are tractable for LP-trees.*

[9] [36] say that $o$ is in this case "weakly undominated". They also say that $o$ is: *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 $o' \neq o$, $o \succ_\varphi o'$. The complexity of queries related to the latter three definitions is studied in [29].

| | GAI | $GAI_k$ | $GAI_1$ | CP | 1-CP | 1-CP$\not\forall$ | 1-CP$\not\forall\wedge$ | CPnet | CPnet$\not\emptyset$ | CPnet$\not\emptyset$poly | LPT | 1-LPTlin | complete |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CONDITIONING | ✓ | ✓ | ✓ | | ╳ | ╳ | ╳ | ╳ | ╳ | ╳ | ✓ | ✓ | |
| CONJUNCTION | ╳ | ╳ | ╳ | | ╳ | ╳ | ╳ | ╳ | ╳ | ╳ | | ╳ | |
| DISJUNCTION | ╳ | ╳ | ╳ | ╳ | ╳ | ╳ | ╳ | ╳ | ╳ | ╳ | ╳ | ╳ | |
| LOWER PROJECTION | ╳ | ╳ | ✓ | | ╳ | ╳ | ╳ | ╳ | ╳ | ╳ | | ✓ | |
| WEAK OPT. PROJ. | ✘○ | ✘○ | ✓ | | ╳ | ╳ | ╳ | ╳ | ╳ | | | ✓ | |
| STRONG OPT. PROJ. | ✘○ | ✘○ | ✓ | | ╳ | ╳ | ╳ | ╳ | ╳ | | | ✓ | |

Each column corresponds to one sublanguage of CP. They are sorted in order of decreasing expressiveness from left to right, except when columns are separated by double lines. For each transformation and sublanguage: ✓ = polytime answer; ✘○ = NP/coNP-hard transformation; ╳= transformation result may be outside the language.

Table 2: Complexity of Transformations.

**Proposition 17.** *$\succeq$-CUT EXTRACTION, $\succ$-CUT EXTRACTION, UNDOMINATED CHECK, and UNDOMINATED EXTRACT are tractable for $GAI_1$.*

The problems above become intractable as soon as we allow utility functions of arity at least 2.

**Proposition 18.** *$\succeq$-CUT EXTRACTION and $\succ$-CUT EXTRACTION are NP-complete for $GAI_k$ for $k \geq 2$ and GAI. UNDOMINATED CHECK is coNP-complete and UNDOMINATED EXTRACT is NP-hard for $GAI_k$ for $k \geq 2$ and for GAI.*

# 7 Transformations

Several transformations have been studied in the literature on knowledge compilation. A transformation takes as input one or more formulas, and, possibly, other arguments like some attributes, and returns another formula. Table 2 summarizes our results on these transformations. As can be seen from the table, for many sublanguages of CP and transformations, the result of the transformation may be outside that sublanguage.

## 7.1 Conditioning

Several studies, in particular in the context of propositional logic like e.g. [24] work with a syntactic definition of this transformation; however, in logic, these definitions have a clear semantic counterpart. In the case of CP statements, we shall see that there are languages for which the transformation cannot always be applied, so we give a semantic description, similar to the one given by [31].

Given a preference relation $\succeq$ on $\mathcal{X}$ and a partial instantiation $u \in \underline{U}$ for some $U \subseteq \mathcal{X}$, let $\succeq^{|u}$ be the relation defined for every $r, r' \in \underline{\mathcal{X} - U}$ by $r \succeq^{|u} r'$ if and only if $ru \succeq r'u$. It is straightforward to check that $\succeq^{|u}$ is a preorder. Conditioning a formula $\varphi$ in a language consists in computing a formula of the same language representing $\succeq_{\varphi}^{|u}$.

CONDITIONING Given a language $\mathcal{L}$, a formula $\varphi$ of $\mathcal{L}$ and an instantiation $u \in \underline{U} \subseteq \mathcal{X}$, compute a formula $\varphi' \in \mathcal{L}$ that represents $\succeq_{\varphi}^{|u}$.

For LPT, a simple syntactic transformation on a formula $\varphi$ allows, for every attribute $X$ and every value $x \in \underline{X}$, to represent $\succeq_{\varphi}^{|x}$: for every node $N$, whose label contains $X$, remove $X$ from the label of $N$, remove the node if it contains no other attribute; if $N$ has several children, keep only those that correspond to instantiation $X = x$ (there will only be one if the label of $N$ contains no other attribute); at the nodes below $N$, replace every rule $\alpha : \geq$ by $\alpha^{|x} : \geq$, where $\alpha^{|x}$ is the result of conditioning applied to $\alpha$, as defined by e.g. [24]; remove the rule if $\alpha^{|x} \models \bot$; otherwise, since we assume that $\geq$ is given in extension, it is easy to keep only the pairs $(u, u')$ such that $u[X] = u'[X] = x$ and remove $x$ from them. This can be performed with a single traversal of the tree.

Even simpler, conditioning a $GAI$ with $X = x$ amounts to removing from every sub-utility that bears on $X$ the cases where $X \neq x$.

The next example is an acyclic CP-net, whose dependency graph is even linear, for which there is a conditioning transformation, the result of which cannot be expressed in 1-CP$\not\forall$.

**Example 10.** *Consider 3 binary attributes $A, B, C$, with respective domains $\{a, \bar{a}\}$, $\{b, \bar{b}\}$, $\{c, \bar{c}\}$, and let*

$$\varphi = \{a \geq \bar{a}, a : b \geq \bar{b}, \bar{a} : \bar{b} \geq b, b : c \geq \bar{c}, \bar{b} : \bar{c} \geq c\}.$$

*The underlying, acyclic dependency graph has set of edges $\{(A, B), (B, C)\}$. Then $abc \succeq_\varphi ab\bar{c} \succeq_\varphi a\bar{b}\bar{c} \succeq_\varphi \bar{a}\bar{b}\bar{c} \succeq_\varphi \bar{a}\bar{b}c \succeq_\varphi \bar{a}bc \succeq_\varphi \bar{a}b\bar{c}$, thus $abc \succeq_\varphi ab\bar{c} \succeq_\varphi \bar{a}bc \succeq_\varphi \bar{a}b\bar{c}$, that is: $ac \succeq_\varphi^{|b} a\bar{c} \succeq_\varphi^{|b} \bar{a}c \succeq_\varphi^{|b} \bar{a}\bar{c}$. However, $\succeq_\varphi^{|b}$ cannot be represented in 1-CP$\not\emptyset$.*

Note that, in the example above, $\succeq_\varphi^{|b}$ can be represented in 1-CP, with formula $\{| C : a \geq \bar{a}, c \geq \bar{c}, \}$. The next example is another CP-net, with a cycle in the dependency graph, for which there is a conditioning transformation, the result of which cannot be expressed in 1-CP.

**Example 11.** *Consider 3 binary attributes $A, B, C$, with respective domains $\{a, \bar{a}\}$, $\{b, \bar{b}\}$, $\{c, \bar{c}\}$, and let*

$$\varphi = \{b\bar{c} : \bar{a} \geq a, \neg(b\bar{c}) : a \geq \bar{a}, c : b \geq \bar{b}, \bar{c} : \bar{b} \geq b, a : c \geq \bar{c}, \bar{a} : \bar{c} \geq c\}.$$

*The underlying dependency graph has set of edges $\{(B, A), (C, A), (C, B), (A, C)\}$, it is not acyclic. $\varphi$ represents the preorder that is the transitive closure of $abc \succeq_\varphi a\bar{b}c \succeq_\varphi a\bar{b}\bar{c} \succeq_\varphi \bar{a}\bar{b}\bar{c} \succeq_\varphi \bar{a}b\bar{c} \succeq_\varphi \bar{a}\bar{b}c$ and $\bar{a}b\bar{c} \succeq_\varphi ab\bar{c}$, thus $ac \succeq_\varphi^{|b} \bar{a}\bar{c} \succeq_\varphi^{|b} \bar{a}c$ and $\bar{a}\bar{c} \succeq_\varphi^{|b} a\bar{c}$: $\varphi^*$ contains all swaps sanctioned by the 1-CP$\not\emptyset$ statements $c : a \geq \bar{a}$ (because $ac \succeq_\varphi^{|b} \bar{a}c$), $\bar{c} : \bar{a} \geq a$, $a : c \geq \bar{c}$ and $\bar{a} : \bar{c} \geq c$, but these statements do not entail that $ac$ is preferred over $\bar{a}\bar{c}$.*

Since CP can represent any preorder, the result of a conditioning transformation can be expressed in CP.

## 7.2 Conjunction

Conjunction is classical for Boolean functions: given two Boolean functions $f_\varphi^\mathcal{L}$ and $f_\psi^\mathcal{L}$ represented in a language $\mathcal{L}$, one looks for an $\mathcal{L}$ representation of $f_\varphi^\mathcal{L} \wedge f_\psi^\mathcal{L}$. An analogous definition is also possible when considering formulas representing preferences:

Conjunction Given a language $\mathcal{L}$, two formulas $\varphi$ and $\psi$ of $\mathcal{L}$ compute a formula $\chi$ of $\mathcal{L}$ such that $o \succeq_\chi o'$ if and only if $o \succeq_\varphi o'$ and $o \succeq_\psi o'$.

This definition corresponds to the classical unanimity rule used in ordinal aggregation.

The conjunction of two preorders is a preorder, thus CP is closed under conjunction. Furthermore, the conjunction of two antisymmetric preorders is antisymmetric too, thus LPT is closed under conjunction

The GAI language is not complete for such a transformation: the expressiveness of this languages is limited to complete relations whereas the conjunction of two complete preference relations is not complete in the general case. Consider for instance a GAI decomposition $\varphi$ such that there at least two alternatives $o$ and $o'$ with $o \succ_\varphi o'$, and let $\psi$ be the GAI decomposition defined by $g_\psi(o) = -g_\varphi(o)$ for every alternative $o$: then $o' \succ_\psi o$ and $o' \bowtie_\chi o$, where $\chi$ denotes the conjunction of $\varphi$ and $\psi$. Note that this also applies if $\varphi \in \mathsf{GAI}_1$, therefore neither GAI nor $\mathsf{GAI}_1$ are closed under conjunction.

The next example shows that the languages CPnet$\emptyset^{\mathsf{poly}}$, CPnet$\emptyset$ and CPnet are not closed under conjunction.

**Example 12.** *Consider the following two CP-nets in variables $A, B$: $\varphi$ with statements $a \geq \bar{a}$, $a : b \geq \bar{b}$ and $\bar{a} : \bar{b} \geq b$ and $\psi$ with statements $\bar{a} \geq a$, $a : b \geq \bar{b}$ and $\bar{a} : \bar{b} \geq b$. The directed graph over the variables in both cases is a polytree that has $A$ as the parent of $B$, so both $\varphi$ and $\psi$ are indeed polytree CP-nets. The complete orders induced by $\varphi$ and $\psi$ are respectively*

$$ab \succeq_\varphi a\bar{b} \succeq_\varphi \bar{a}\bar{b} \succeq_\varphi \bar{a}b,$$

$$\bar{a}\bar{b} \succeq_\psi \bar{a}b \succeq_\psi ab \succeq_\psi a\bar{b}.$$

*Then the only preferences in $\succeq_{\varphi\wedge\psi}$ are $ab \succeq_{\varphi\wedge\psi} a\bar{b}$ and $\bar{a}\bar{b} \succeq_{\varphi\wedge\psi} \bar{a}b$. This cannot be expressed by a CP-net, since any CP-net on $\{A, B\}$ orders the four unary swaps, in particular any CP-net must order for instance $\{\bar{a}\bar{b}, a\bar{b}\}$, which $\succeq_{\varphi\wedge\psi}$ does not. It cannot be represented with an LP-tree nor with a utility since it is not a complete relation.*

We now give an example that shows that 1-CP, 1-CP$\not\emptyset$, and 1-CP$\not\emptyset\wedge$ are not closed under conjunction.

**Example 13.** *Consider the following two sets of CP-statements over binary attributes $A$ and $B$: $\varphi = \{b : \bar{a} > a, a : b > \bar{b}, \bar{b} : a > \bar{a}\}$ and $\psi = \{a : \bar{b} > b, b : a > \bar{a}, \bar{a} : b > \bar{b}\}$. Both $\varphi$ and $\psi$ are in 1-CP$\not\emptyset\wedge$. The two sets respectively induce the orders*

$$\bar{a}b \succeq_\varphi ab \succeq_\varphi a\bar{b} \succeq_\varphi \bar{a}\bar{b},$$

$$a\bar{b} \succeq_\psi ab \succeq_\psi \bar{a}b \succeq_\psi \bar{a}\bar{b}.$$

*For the conjunction, we get that $ab \succeq_{\varphi\wedge\psi} \bar{a}\bar{b}$ and $ab$ is incomparable to the other two alternatives. Thus, this conjunction cannot be expressed by a 1-CP, since we cannot go from $ab$ to $\bar{a}\bar{b}$ in a 1-CP without any intermediate flips.*

Many rules of ordinal aggregation could be considered and this opens a large stream of research which is out of the scope of the present paper - e.g. scoring rules like Borda's, for which the GAI framework is obviously a good candidate language. LP trees on the other hand will probably fail to handle such rules, because the aggregation of several lexicographic orders is generally not a lexiocographic order. The CP language in itself as such is neither powerful enough to encompass most of the rules, but extensions have been proposed that typically address this question [44].

## 7.3 Disjunction

We can define the disjunction operation by symmetry:

DISJUNCTION Given a language $\mathcal{L}$ and two formulas $\varphi$ and $\psi$ of $\mathcal{L}$, compute a formula $\chi'$ of $\mathcal{L}$ such that $o \succeq_{\chi'} o'$ if and only if $o \succeq_\varphi o'$ or $o \succeq_\psi o'$.

Such a definition is nevertheless not really significant in the domain of preference handling, since the disjunction of two transitive relations is generally not transitive: it can happen that $o \succ_\varphi o'$, $o' \succ_\psi o$" but neither $o \succ_\varphi o$" nor $o \succ_\psi o$".

**Example 14.** *$\varphi$ is the linear 1-LPT where $B$ is more important than $A$ with $a$ preferred to $\bar{a}$ and $\bar{b}$ preferred to $b$; $\psi$ is the linear 1-LPT where $A$ is more important than $B$ with $a$ preferred to $\bar{a}$ and $b$ preferred to $\bar{b}$. We get*

$$a\bar{b} \succ_\varphi \bar{a}\bar{b} \succ_\varphi ab \succ_\varphi \bar{a}b \quad \textit{and} \quad ab \succ_\psi a\bar{b} \succ_\psi \bar{a}b \succ_\psi \bar{a}\bar{b}$$

*Now, $\bar{a}\bar{b} \succ_\varphi ab$ and $ab \succ_\psi a\bar{b}$, but $\bar{a}\bar{b} \not\succ_\varphi a\bar{b}$ and $\bar{a}\bar{b} \not\succ_\psi a\bar{b}$. Note that $\varphi$ and $\psi$ can be represented with additive utilities and with CP-nets.*

This shows that none of the languages studied in this paper is complete for disjunction.

## 7.4 Variable elimination

We next consider transformations where the information is *projected* onto a subset of the initial variables of interest. This is also called *variable elimination*. In an interactive setting, like product configuration, it enables the user to focus on her preferences over a subset of the variables, which may be less daunting than considering the preferences over the entire set of variables. Variable elimination is a well-known technique in propositional logic, as well as in many graphical models like Bayesian networks or constraint satisfaction problems (weighted or not), where it is a component of some efficient query answering algorithms, that can be used for instance for GAIs. (See e.g. [20] for a recent unified description and overview of algorithmic aspects of graphical models.)

Variable elimination has not been studied much in the context of preferences in general. In a pioneering work, [6, 5] distinguish several ways to define the projection of a preference relation onto a subset of variables. Given a preorder $\succeq$ and a set of variable $U \subseteq \mathcal{X}$, let $V = \mathcal{X} \setminus U$, they first consider two relations defined on $\underline{V}$:

**Lower projection** $v \succeq^{\downarrow V}_{\text{low}} v'$ if and only if $uv \succeq uv'$ for every $u \in \underline{U}$;

**Upper projection** $v \succeq^{\downarrow V}_{\text{up}} v'$ if and only if $uv \succeq uv'$ for some $u \in \underline{U}$.

It is easy to see that $\succeq^{\downarrow V}_{\text{low}}$ and $\succeq^{\downarrow V}_{\text{up}}$ are respectively the conjunction and the disjunction of the relations obtained by conditioning the original relation by every combination of value for $\mathcal{X} \setminus U$.

Let us also consider what [5] call the *optimistic* projections of $\succeq$ on $\underline{V}$:

**Weak optimistic projection** $v \succeq^{\downarrow V}_{\text{w.opt.}} v'$ if and only if for every $u' \in \underline{U}$, there is $u \in \underline{U}$, such that $uv \succeq u'v'$.

**Strong optimistic projection** $v \succeq^{\downarrow V}_{\text{s.opt.}} v'$ if and only if there is $u \in \underline{U}$, such that for every $u' \in \underline{U}$, $uv \succeq u'v'$.

[6] prove that $\succeq^{\downarrow V}_{\text{w.opt.}}$ extends $\succeq^{\downarrow V}_{\text{low}}$ and $\succeq^{\downarrow V}_{\text{s.opt.}}$ (if $o \succeq^{\downarrow V}_{\text{low}} o'$ (resp. $o \succeq^{\downarrow V}_{\text{s.opt.}} o'$), then $o \succeq^{\downarrow V}_{\text{w.opt.}} o'$), and that the weak and strong optimistic projections are identical when $\succeq$ is a weak order. [10]

**Proposition 19.** *Given a preorder $\succeq$ over $\underline{\mathcal{X}}$, given $V \subseteq \mathcal{X}$, let $U \subseteq \mathcal{X} \setminus V$. If $v, v' \in \underline{V}$ and $v \succeq^{\downarrow V}_{\text{w.opt.}} v'$, then there is some $u \in \underline{U}$ such that for no $u' \in \underline{U}$ it holds that $u'v' \succ uv$.*

The proposition above indicates that if $v \succeq^{\downarrow V}_{\text{w.opt.}} v'$, then a decision maker may safely focus on $v$, without risking missing a strictly better full alternative that would extend $v'$. Note that this holds too if $v \succeq^{\downarrow V}_{\text{s.opt.}} v'$ or if $v \succeq^{\downarrow V}_{\text{low}} v'$, since both imply $v \succeq^{\downarrow V}_{\text{w.opt.}} v'$. From a preference representation point of view, the weak optimistic projection seems more interesting, as it is the one that keeps the most information – it contains the other two.

We define the following transformations, for any projection $\pi \in \{\text{low}, \text{up}, \text{s.opt}, \text{w.opt}\}$:

$\pi$-PROJECTION Given some language $\mathcal{L}$, some formula $\varphi \in \mathcal{L}$, some subset of variables $V$, return a formula $\psi \in \mathcal{L}$ such that $\succeq_\psi = (\succeq_\varphi)^{\downarrow V}_\pi$.

[10] [6] also define *pessimistic* counterparts of the optimistic projections: $v \succeq^{\downarrow V}_{\text{w.pess.}} v'$ if and only if for every $u \in \underline{U}$, there is $u' \in \underline{U}$, such that $uv \succeq u'v'$; and $v \succeq^{\downarrow V}_{\text{s.pess.}} v'$ if and only if there is $u' \in \underline{U}$, such that for every $u \in \underline{U}$, $uv \succeq u'v'$. We do not consider them here because their significance, from a decision making point of view, is not clear.

The next example shows that the languages GAI and CP-net, CPnet$\not\emptyset$ and CPnet$\not\emptyset^{\mathsf{poly}}$ are not closed under lower projection.

**Example 15.** *Consider the preference relation $ab \succ a\bar{b} \succ \bar{a}\bar{b} \succ \bar{a}b$. This relation can be represented with a utility function, and by an acyclic, polytree CP-net where $A$ has no parent and is the only parent of $B$.*

*The elimination of $A$ by lower projection will lead to the preorder in which the only two alternatives $b$ and $\bar{b}$ are incomparable due to $ab \succ a\bar{b}$ and $\bar{a}b \prec \bar{a}\bar{b}$. Thus the lower projection cannot be expressed by a GAI, nor with a CP-net since a CP-net over $\{B\}$ has only one node labelled with $B$ and the associated table must order the pair of values $\{b, \bar{b}\}$; nor with a complete LP-tree.*

We will next see that the 1-CP families are not closed under lower projection either.

**Example 16.** *The idea is to modify the construction in Example 13 by adding an additional variable $C$ to simulate conjunction. So we consider the 1-CP$\not\vee\wedge$formula*

$$\varphi = \{cb : \bar{a} > a, ca : b > \bar{b}, c\bar{b} : a > \bar{a}, \bar{c}a : \bar{b} > b, \bar{c}b : a > \bar{a}, \bar{c}\bar{a} : b > \bar{b}\}.$$

*In the resulting preorder we have*

$$c\bar{a}b \succeq_\varphi cab \succeq_\varphi ca\bar{b} \succeq_\varphi c\bar{a}\bar{b}, \bar{c}a\bar{b} \succeq_\varphi \bar{c}ab \succeq_\varphi \bar{c}\bar{a}b \succeq_\varphi \bar{c}\bar{a}\bar{b},$$

*and all other pairs of alternatives are incomparable.*

*When eliminating $C$ by lower projection, we get $ab \succeq_{\varphi,low}^{\downarrow\{A,B\}} \bar{a}\bar{b}$ and $ab$ is incomparable to the other alternatives. However, as we have seen before, this cannot be expressed by a 1-CP, since we cannot go from $ab$ to $\bar{a}\bar{b}$ in a 1-CP without any intermediate flips.*

Most of the languages considered in this paper are not complete for the upper projection, because this projection may lead to a non transitive relation (since the disjunction of two relations is not necessarily transitive), except LPT$^{\mathsf{lin}}$ and GAI$_1$. [11]

**Example 17.** *We simply construct an LP tree on $\{A, B, C\}$ that has $C$ as the attribute in the root with two children; for the left child of $C$, the LPT tree for $\varphi$ from Example 14 is used, for the right child LPT for the relation $\psi$ is used. Then, when we apply upper projection on $C$, we get the disjunction of the orders $\varphi$ and $\psi$ which, as argued before, cannot be expressed as an LPT.*

*As previously, the same counter example holds when considering GAI nets.*

Interestingly, linear 1-LP trees and additive utilities (GAI$_1$) avoid the problems in these two counterexamples.

**Proposition 20.** *All four projections defined above are equivalent for the 1-GAI language and the language that contains complete LP-trees of 1-LPT$^{lin}$, and can be computed in polynomial time.*

We next show that if both the conditioning and the weak optimistic projection can be done in polynomial time on a language, then an undominated alternative $o$ can be obtained in polynomial time, as well.

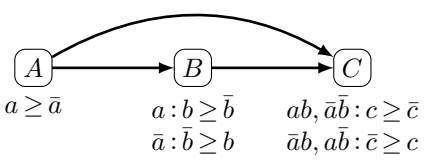
**Proposition 21.** *If conditioning can be done in polynomial time for language $\mathcal{L}$ but the extraction of an undominated alternative is NP-hard, then the strong optimistic projections cannot be computed in polynomial time for $\mathcal{L}$ (unless P = NP).*

A direct corollary is that GAI and GAI$_k$ ($k > 1$) fail to provide strong and weak optimistic projections in polynomial time (unless P = NP) - since (i) the extraction of an undominated alternative is NP-hard for these languages and (ii) they support conditioning in polytime.

**Proposition 22.** *The strong (resp. weak) projection cannot be computed in polytime for GAI and GAI$_k$ ($k > 1$) (unless P = NP).*

The next example shows that the weak and strong optimistic projections of the preorder induced by an acyclic CP-net cannot always be represented in 1-CP.

**Example 18.** *Consider a CP-net $N$ over three binary attributes $A$, $B$ and $C$, with respective domains $\{a, \bar{a}\}$, $\{b, \bar{b}\}$, $\{c, \bar{c}\}$:*




*Let $\succeq$ denote the linear order represented by $N$: $abc \succeq ab\bar{c} \succeq a\bar{b}\bar{c} \succeq a\bar{b}c \succeq \bar{a}\bar{b}c \succeq \bar{a}\bar{b}\bar{c} \succeq \bar{a}b\bar{c} \succeq \bar{a}bc$, and it can be checked that $\succeq_{w.opt.}^{\downarrow AC}$ is the relation that corresponds to the set of CP-statements $\{a \geq \bar{a}, c \geq \bar{c}, a\bar{c} \geq \bar{a}c\}$, which is not included in 1-CP: they correspond to a CP-net where $A$ and $C$ are preferentially independent but with the additional trade-off $a\bar{c} \geq \bar{a}c$. Since $\succeq$ is complete, $\succeq_{s.opt.}^{\downarrow AC}$ is the same as $\succeq_{w.opt.}^{\downarrow AC}$ [5].*

[11] [6, 5] in fact define the *upper projection* to be the transitive closure of the relation that we denote by $\succeq_U^{\downarrow V}$ above. However this means completing a relation when there is not necessarily a justification to do so.

# 8 Conclusion

The literature on languages on CP statements has long focused on statements with unary swaps. Several examples in Section 4 show that this strongly degrades expressiveness. Table 1 shows that comparison queries seem to resist tractability for CP-statements, but the TOP-$p$ query may be sufficient in many applications. The practical interest of CP-nets also lies in the fact that with this language, finding an optimal (undominated) alternative is easy [10].

Contrastingly, with GAIs, it is easy to compare alternatives, but computing an undominated alternative is only tractable in the very restrictive case of additive utilities ($\mathsf{GAI}_1$).

Tractability of the EQUIVALENCE query relies on the existence of canonical form: it is the case when the language enforces a structure like a dependency graph or a tree, and when the conditions of the statements are restricted to some propositional language with a canonical form.

As for transformations, the languages of (generalized) additive utilities and LP trees seem to offer better prospects, as in both cases conditioning is tractable – whereas conditioning a formula of the most studied sublanguages of CP does not always result in a formula in the same language. Note however that for projections, tractability necessitates very strong restrictions (it only holds for $\mathsf{GAI}_1$ and $\text{1-LPT}^{\mathsf{lin}}$.

An important direction for future work is to study the properties of the various languages studied here with respect to machine learning: in some context, preferences can be learnt, either through some interaction with the current user of a system, or from data gathered during past interactions. The complexity of this learning phase can influence the choice of preference model, depending on the type of interaction and on the amount of data available, and also on the computational complexity of the learning algorithms. Preliminary results about the complexity of learning CP-nets, GAIs, LP-trees can be found in e.g. [9, 39, 19, 7, 1, 3, 2, 28].

**Acknowledgements** We thank anonymous referees for their valuable comments. This work has benefited from the AI Interdisciplinary Institute ANITI. ANITI is funded by the French "Investing for the Future – PIA3" program under grant agreement ANR-19-PI3A-0004. This work has also been supported by the PING/ACK project of the French National Agency for Research, grant agreement ANR-18-CE40-0011.

# Bibliography


[1] Eisa Alanazi, Malek Mouhoub, and Sandra Zilles. The complexity of learning acyclic CP-nets. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI 2016)*, page 1361–1367. IJCAI/AAAI Press, 2016.

[2] Eisa Alanazi, Malek Mouhoub, and Sandra Zilles. The complexity of exact learning of acyclic conditional preference networks from swap examples. *Artififical Intelligence*, 278, 2020.

[3] Thomas E. Allen, Cory Siler, and Judy Goldsmith. Learning tree-structured CP-nets with local search. In Vasile Rus and Zdravko Markov, editors, *Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2017)*, pages 8–13. AAAI Press, 2017.

[4] Fahiem Bacchus and Adam J. Grove. Graphical models for preference and utility. In Philippe Besnard and Steve Hanks, editors, *UAI*, pages 3–10. Morgan Kaufmann, 1995.

[5] Philippe Besnard, Jérôme Lang, and Pierre Marquis. Variable forgetting in preference relations over combinatorial domains. In Brewka et al. [16], page 763–764.

[6] Philippe Besnard, Jérôme Lang, and Pierre Marquis. Variable forgetting in preference relations over combinatorial domains. In *Proceedings of the IJCAI Multidisciplinary Workshop on Advances in Preference Handling (MPREF'05)*, 2005.

[7] Damien Bigot, Hélène Fargier, Jérôme Mengin, and Bruno Zanuttini. Using and learning GAI-decompositions for representing ordinal rankings. In Fürnkranz and Hüllermeyer [33], page 5–10.

[8] Richard Booth, Yann Chevaleyre, Jérôme Lang, Jérôme Mengin, and Chattrakul Sombattheera. Learning various classes of models of lexicographic orderings. Rapport de recherche IRIT/RR–2009-21–FR, IRIT, Université Paul Sabatier, Toulouse, juin 2009.

[9] Richard Booth, Yann Chevaleyre, Jérôme Lang, Jérôme Mengin, and Chattrakul Sombattheera. Learning conditionally lexicographic preference relations. In Helder Coelho, Rudi Studer, and Michael Wooldridge, editors, *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, page 269–274. IOS Press, 2010.

[10] Craig Boutilier, Romen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.

[11] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. Preference-based constrained optimization with CP-nets. *Computational Intelligence*, 20(2):137–157, 2004.

[12] Craig Boutilier, Ronen I. Brafman, Holger H. Hoos, and David Poole. Reasoning with conditional ceteris paribus preference statements. In Kathryn B. Laskey and Henri Prade, editors, *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, page 71–80. Morgan Kaufmann, 1999.

[13] Sylvain Bouveret, Ulle Endriss, and Jérôme Lang. Conditional importance networks: A graphical language for representing ordinal, monotonic preferences over sets of goods. In Craig Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, page 67–72, 2009.

[14] Ronen I. Brafman, Carmel Domshlak, and Solomon E. Shimony. On graphical modeling of preference and importance. *Journal of Artificial Intelligence Research*, 25:389–424, 2006.

[15] Darius Braziunas and Craig Boutilier. Local utility elicitation in GAI models. In Fahiem Bacchus and Tommi Jaakkola, editors, *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI'05)*, page 42–49. AUAI Press, 2005.

[16] Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso, editors. *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006)*, Frontiers in Artificial Intelligence and Applications. IOS Press, 2006.

[17] Michael Bräuning and Eyke Hüllermeyer. Learning conditional lexicographic preference trees. In Fürnkranz and Hüllermeyer [33], page 11–15.

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

[19] Yann Chevaleyre, Frédéric Koriche, Jérôme Lang, Jérôme Mengin, and Bruno Zanuttini. Learning ordinal preferences on multiattribute domains: the case of CP-nets. In Johannes Fürnkranz and Heike Hüllermeier, editors, *Preference learning*, page 273–296. Springer, 2011.

[20] Martin C. Cooper, Simon de Givry, and Thomas Schiex. Graphical models: Queries, complexity, algorithms (tutorial). In Christophe Paul and Markus Bläser, editors, *Proceedings of the 37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020)*, volume 154 of *LIPIcs*, page 4:1–4:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[21] Sylvie Coste-Marquis, Jérôme Lang, Paolo Liberatore, and Pierre Marquis. Expressive power and succinctness of propositional languages for preference representation. In Dubois et al. [26], page 203–212.

[22] Yves Crama and Peter L. Hammer. *Boolean Functions - Theory, Algorithms, and Applications*, volume 142 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2011.

[23] Adnan Darwiche. Compiling knowledge into decomposable negation normal form. In Thomas Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI 99)*, page 284–289. Morgan Kaufmann, 1999.

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

[25] Carmel Domshlak and Ronen I. Brafman. CP-nets: Reasoning and consistency testing. In Dieter Fensel, Fausto Giunchiglia, Deborah L. McGuinness, and Mary-Anne Williams, editors, *Proceedings of the Eights International Conference on Principles of Knowledge Representation and Reasoning (KR-02)*, page 121–132. Morgan Kaufmann, 2002.

[26] Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors. *Proceedings of the Ninth International Conference on the Principles of Knowledge Representation and Reasoning*. AAAI Press, 2004.

[27] Piotr Faliszewski and Lane A. Hemaspaandra. The complexity of power-index comparison. *Theor. Comput. Sci.*, 410(1):101–107, 2009.

[28] Hélène Fargier, Pierre-François Gimenez, Jérôme Mengin, and Bao Ngoc Le Nguyen. The complexity of unsupervised learning of lexicographic preferences. In Meltem Öztürk, Paolo Viappiani, Christophe Labreuche, and Sébastien Destercke, editors, *Proceedings of the 13th Multidisciplinary Workshop on Advances in Preference Handling*, 2022. Also on CoRR: 10.48550/arXiv.2209.11505.

[29] Hélène Fargier and Jérôme Mengin. A knowledge compilation map for conditional preference statements-based languages. In Frank Dignum, Ulle Endriss, Alessio Lomuscio, and Ann Nowé, editors, *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2021.

[30] Hélène Fargier, Pierre Francois Gimenez, and Jérôme Mengin. Learning lexicographic preference trees from positive examples. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*, page 2959–2966. AAAI Press, 2018.

[31] Hélène Fargier, Pierre Marquis, Alexandre Niveau, and Nicolas Schmidt. A knowledge compilation map for ordered real-valued decision diagrams. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, page 1049–1055. AAAI Press, 2014.

[32] Peter C. Fishburn. Lexicographic orders, utilities and decision rules: A survey. *Management Science*, 20(11):pp. 1442–1471, 1974.

[33] Johannes Fürnkranz and Eyke Hüllermeyer, editors. *Preference Learning: Problems and Applications in AI. Proceedings of the ECAI 2012 workshop*, 2012.

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

[35] Goran Gogic, Henry A. Kautz, Christos H. Papadimitriou, and Bart Selman. The comparative linguistics of knowledge representation. In Mellish [42], page 862–869.

[36] Judy Goldsmith, Jérôme Lang, Miroslaw Truszczynski, and Nic Wilson. The computational complexity of dominance and consistency in CP-nets. *Journal of Artificial Intelligence Research*, 33:403–432, 2008.

[37] Christophe Gonzales and Patrice Perny. GAI networks for utility elicitation. In Dubois et al. [26], page 224–233.

[38] Christophe Gonzales and Patrice Perny. GAI networks for utility elicitation. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *KR2004)*, pages 224–234, 2004.

[39] Frédéric Koriche and Bruno Zanuttini. Learning conditional preference networks. *Artificial Intelligence*, 174(11):685–703, 2010.

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

[41] Thomas Lukasiewicz and Enrico Malizia. Complexity results for preference aggregation over (m)CP-nets: Pareto and majority voting. *Artificial Intelligence*, 272:101–142, 2019.

[42] Chris S. Mellish, editor. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 95)*. Morgan Kaufmann, 1995.

[43] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[44] Francesca Rossi, Kristen Brent Venable, and Toby Walsh. mCP nets: Representing and reasoning with preferences of multiple agents. In *AAAI*, volume 4, pages 729–734, 2004.

[45] Thomas Schiex, Hélène Fargier, and Gérard Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In Mellish [42], page 631–639.

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

[47] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.

[48] Nic Wilson. Consistency and constrained optimisation for conditional preferences. In Ramón López de Mántaras and Lorenza Saitta, editors, *Proceedings of the 16th Eureopean Conference on Artificial Intelligence (ECAI 2004)*, page 888–892. IOS Press, 2004.

[49] Nic Wilson. Extending CP-nets with stronger conditional preference statements. In Deborah L. McGuinness and George Ferguson, editors, *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI'04)*, page 735–741. AAAI Press / The MIT Press, 2004.

[50] Nic Wilson. An effcient upper approximation for conditional preference. In Brewka et al. [16].

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

# 1 Proofs for Section 3 (Languages)

**Proposition 1.** *Let $\varphi$ be an LP-tree over $\mathcal{X}$, then $\succeq_\varphi$ as defined above is a partial order. Furthermore, $\succeq_\varphi$ is a linear order if and only if 1) every attribute appears on every branch and 2) every preference rule specifies a linear order.*

*Proof.* By definition, $\succeq_\varphi$ is reflexive. The fact that it is antisymmetric follows from the antisymmetry of the local preference relations in the conditional preference tables. For transitivity, the proof given by [8] is for a restricted family of LP-trees, so we recast it here for our more general family of LP-trees. Suppose that $o \succeq_\varphi o' \succeq_\varphi o''$ and $o$, $o'$, $o''$ are distinct. There must be a node $N$ at which $\{o, o'\}$ is decided, let $W$ be the set of attributes that labels $N$, then $o[\mathsf{Anc}(N)] = o'[\mathsf{Anc}(N)]$, and there is one rule $\alpha : \geq$ such that $o[\mathsf{NonInst}(N)] = o'[\mathsf{NonInst}(N)] \models \alpha$ and $o[W] \geq o'[W]$. Similarly, let $N'$ be the node at which $\{o', o''\}$ is decided, let $W'$ be the set of attributes that labels $N'$, then $o[\mathsf{Anc}(N')] = o'[\mathsf{Anc}(N')]$, and there is one rule $\alpha' : \geq'$ s.t. $o'[\mathsf{NonInst}(N')] = o''[\mathsf{NonInst}(N')] \models \alpha'$ and $o'[W'] \geq' o''[W']$. If $N = N'$, then $o[\mathsf{Anc}(N)] = o'[\mathsf{Anc}(N)] = o''[\mathsf{Anc}(N)]$, and $o[W] > o'[W] > o''[W]$ since $\geq$ is antisymmetric, thus $o[W] > o''[W]$ because $\geq = \geq'$ is also transitive, hence $N$ decides $\{o, o''\}$ and $o \succeq_\varphi o''$. If $N \neq N'$, note that both nodes are in the unique branch in $\varphi$ that corresponds to $o'$, so one of $N$, $N'$ must be above the

other. Suppose that $N$ is above $N'$, then, it must be the case that $o'[W] = o''[W]$, and $o[W] \neq o'[W]$, thus $N$ decides $\{o, o''\}$; moreover, since $\mathsf{NonInst}(N) \subseteq \mathsf{NonInst}(N')$, $o[\mathsf{NonInst}(N)] = o'[\mathsf{NonInst}(N)] = o''[\mathsf{NonInst}(N)] \models \alpha$, and $o[W] \geq o'[W] = o''[W]$; hence $o \succeq_\varphi o''$. The case where $N'$ is above $N$ is similar.

For the second part of the proposition, suppose first that every attribute appears on every branch and that every preference rule specifies a linear order: we will show that $\succeq_\varphi$ is antisymmetric and connex. For antisymmetry, consider distinct alternatives $o, o' \in \underline{\mathcal{X}}$: because every attribute appears on every branch, there must be a node $N$, labelled with some $W \subseteq \mathcal{X}$, that decides $\{o, o'\}$, and a unique rule $\alpha : \geq$ at $N$ such that $o[\mathsf{NonInst}(N)] = o'[\mathsf{NonInst}(N)] \models \alpha$; $\geq$ must be a linear order over $\underline{W}$, so either $o[W] > o'[W]$ and $o \succ_\varphi o'$, or $o'[W] > o[W]$ and $o' \succ_\varphi o$: $\succeq_\varphi$ is connex and antisymmetric. For the converse, assuming that either there is some branch where some attribute does not appear, or that there is some rule at some node that does not define a linear order, it is not difficult to define two distinct alternatives that cannot be compared with $\succeq_\varphi$. □

# 2 Proofs for Section 4 (Expressiveness)

**Proposition 3.** *$CP = \bigcup_{k\in\mathbb{N}}$ k-CP and, for every $k \in \mathbb{N}$, $k \geq 2$:*

$$\textit{CP}\wedge \sqsubsetneq \textit{CP}\not\rhd \sqsubsetneq \textit{CP}\wedge\not\rhd \sqsubsetneq \textit{CP} \sqsupset \textit{k-CP} \sqsubsetneq \textit{k-CP}\wedge \sqsupset \textit{k-CP}\not\rhd \sqsubsetneq \textit{k-CP}\wedge\not\rhd$$
$$\textit{k-CP} \sqsupset \textit{(k-1)-CP} \quad \textit{and} \quad \textit{k-CP}\wedge\not\rhd \sqsupset \textit{(k-1)-CP}\wedge\not\rhd \sqsupset \textit{CPnet}.$$

*Proof.* That CP $\sqsubsetneq$ CP$\wedge$ $\sqsubsetneq$ CP$\not\rhd$ $\sqsubsetneq$ CP$\wedge\not\rhd$ follows from property 2.

By definition CP $\supset$ 1-CP $\supset$ 1-CP$\not\rhd$ $\supset$ 1-CP$\wedge\not\rhd$ and 1-CP $\supset$ 1-CP$\wedge$ $\supset$ 1-CP$\wedge\not\rhd$, thus CP $\sqsupseteq$ 1-CP $\sqsupseteq$ 1-CP$\not\rhd$ $\sqsupseteq$ 1-CP$\wedge\not\rhd$ and 1-CP $\sqsupseteq$ 1-CP$\wedge$ $\sqsupseteq$ 1-CP$\wedge\not\rhd$. Restricting to conjunction of literals does not induce a loss in expressiveness because, given a statement $\alpha \,|\, V : x \geq x'$, it is possible to compute a DNF logically equivalent to $\alpha$, and then consider a set of statements, each statement having one disjunct of the DNF as conditioning part. Example 3 prove that CP $\sqsupset$ 1-CP. Example 4 proves that 1-CP $\sqsupset$ 1-CP$\not\rhd$, it can be generalized to prove that k-CP$\wedge$ $\sqsupset$ k-CP$\not\rhd$ by considering $k$ binary attributes $A_1, \ldots, A_k$ instead of $A$, and the preorder $a_1 \ldots a_k b \succ a_1 \ldots a_k \bar{b} \succ \bar{a}_1 \ldots \bar{a}_k b \succ \bar{a}_1 \ldots \bar{a}_k \bar{b}$, which can be represented in k-CP$\wedge$ but not in k-CP$\not\rhd$.

To prove that k-CP $\sqsupset$ (k-1)-CP, simply consider $k$ binary attributes $A_1, \ldots, A_k$ and the preorder that contains a single pair: $a_1 \ldots a_k \succ \bar{a}_1 \ldots \bar{a}_k$, it can be represented in k-CP with the single statement $a_1 \ldots a_k \geq \bar{a}_1 \ldots \bar{a}_k$, but not in (k-1)-CP. Note that this statement is in k-CP$\wedge\not\rhd$, so it proves that k-CP$\wedge\not\rhd$ $\sqsupset$ (k-1)-CP$\wedge\not\rhd$. The fact that CP-net $\not\sqsupseteq$ (k-1)-CP follows from the "completeness" condition in the definition of CP-nets: in a CP-net, every attribute must have some local preference rules associated to it, whereas a formula in 1-CP$\wedge\not\rhd$ may consist of one rule only. □

**Proposition 4.** *$LPT = \bigcup_{k\in\mathbb{N}}$ k-LPT and, for every $k \in \mathbb{N}$:*

$$\textit{CP} \sqsupset \textit{LPT} \sqsubsetneq \textit{LPT}\wedge \sqsupset \textit{k-LPT} \sqsubsetneq \textit{k-LPT}\wedge \sqsupset \textit{(k-1)-LPT}$$
$$\textit{k-CP} \sqsupset \textit{k-LPT} \sqsupset \textit{k-LPT}^{\textit{lin}} \sqsupset \textit{(k-1)-LPT}^{\textit{lin}}.$$

*Proof.* LP trees can only represent antisymmetric preorders, so LPT is strictly less expressive than CP. That k-LPT$\wedge$ $\sqsupseteq$ k-LPT follows from the fact that the condition of every CP-statement in the set of CP-statements that correspond to some k-LPT can be represented with a DNF. To see that (k-1)-LPT $\not\sqsupseteq$ k-LPT for every $k \geq 1$, consider some k-LP-tree $\varphi$ with $k$ attributes $X_1, \ldots, X_k$ at the root, and a linear order with $x_1 \ldots x_k$ as top element, and $\bar{x}_1 \ldots \bar{x}_k$ as second best element: then $o \succeq_\varphi o'$ for every pair of alternatives $(o, o')$ such that $o[X_1 \ldots X_k] = x_1 \ldots x_k$ and $o'[X_1 \ldots X_k] = \bar{x}_1 \ldots \bar{x}_k$; no LP-tree in (k-1)-LPT can represent that. Note that we can choose $\varphi$ to be linear, so that proves that (k-1)-LPT$^{\mathsf{lin}}$ $\not\sqsupseteq$ k-LPT$^{\mathsf{lin}}$ too.

To show that k-LPT$^{\mathsf{lin}}$ $\not\sqsupseteq$ k-LPT, consider an LP-tree with attributes $X_1 \ldots X_k$ at the root, with $|\underline{\{X_1 \ldots X_k\}}|$ children, where every child is labelled with binary attribute $Y$, and at least two children order $y$ and $\bar{y}$ differently: no linear $k$-LP tree can represent the same order.

That k-CP $\sqsupseteq$ k-LPT follows from the remark below Proposition 1 that describes a set of CP-statements equivalent to a given LP-tree $\varphi$: it is not difficult to check that if every node in $\varphi$ has at most $k$ attributes, then the corresponding CP-statements are all in k-CP. To prove that k-LPT $\not\sqsupseteq$ k-CP, consider a CP-net $\varphi$ over $k+1$ binary attributes $X_1, \ldots, X_{k+1}$, with $x_i \geq \bar{x}_i$ for every $1 \leq i \leq k+1$ (thus the CP-net has no edge): clearly $\varphi \in$ 1-CP $\subseteq$ k-CP. Consider now some LP-tree $\psi$ with $j \leq k$ attributes at the root; w.l.o.g. we can assume that these attributes are $X_1, \ldots, X_j$; then the CPT at the root of $\psi$ must contain the preorder over $X_1, \ldots, X_j$ defined by the set of CP-statements $\{x_i \geq \bar{x}_i \,|\, 1 \leq i \leq j\}$. But then $x_1 \ldots x_j \bar{x}_{j+1} \ldots \bar{x}_{k+1} \succ_\psi \bar{x}_1 \ldots \bar{x}_j x_{j+1} \ldots x_{k+1}$, whereas $x_1 \ldots x_j \bar{x}_{j+1} \ldots \bar{x}_{k+1} \not\succ_\varphi \bar{x}_1 \ldots \bar{x}_j x_{j+1} \ldots x_{k+1}$. Since LPT $\sqsupseteq$ CP $\sqsupset$ k-CP, we can conclude too that LPT $\sqsupset$ k-LPT.

□

**Proposition 6.** *For every $k \in \mathbb{N}$: GAI$_{k+1}$ $\sqsupset$ GAI$_k$, and (k-1)-CP $\not\sqsupset$ GAI$_k$.*

*Proof.* We first prove the first statement. $\mathsf{GAI}_{k+1} \sqsupseteq \mathsf{GAI}_k$ is clear, since $\mathsf{GAI}_{k+1} \supseteq \mathsf{GAI}_k$, so it suffices to show that the increase in expressiveness is strict.

Fix the set of attributes $\mathcal{X} = \{X_1, \ldots, X_{k+1}\}$ and set as the domain of each attribute $\underline{X_i} = \{0, 1\}$. For every set $A \subseteq \mathcal{X}$, define the indicator function $I_A(X_1, \ldots, X_{k+1})$ as the function that, given as input an alternative $o \in \underline{\mathcal{X}}$, returns 1 if for all $X \in A$ we have $o(X) = 1$ and 0 otherwise. Set $\varphi = \{I_{\mathcal{X}}\}$, then we have that $g_\varphi(o) = I_{\mathcal{X}}(o)$. For every $A$ define $o_A$ to be the alternative that is 1 on exactly the attributes in $A$. Then $g_\varphi$ induces the total preorder $\succeq$ in which $o_{\mathcal{X}}$ strictly dominates all other alternatives, whereas for all pairs $o, o'$ both different from $o_{\mathcal{X}}$ we have $o \sim o'$.

Clearly, $\succeq$ is expressed in $\mathsf{GAI}_{k+1}$. We claim that it cannot be expressed in $\mathsf{GAI}_k$. To this end, assume that this were wrong, then there is a set $\varphi = \{g_{Z_1}, \ldots, g_{Z_m}\}$ of real valued functions bearing on strict subsets $Z_i$ of $\mathcal{X}$ such that $g_\varphi$ induces the order $\succeq$ on $\underline{\mathcal{X}}$. Without loss of generality, assume that for every $A \subset \mathcal{X}$ the set $\mathcal{X}$ contains exactly one function $g_A$. It will be convenient to represent $g_A$ as a weighted sum of indicator functions.

We use the following representation result for functions from $\{0,1\}^\ell \to \mathbb{R}$ whose proof can e.g. be found in [22, Section 13.2].

**Lemma 1.** *For every function $f : \{0,1\}^\ell \to \mathbb{R}$ with $\ell \in \mathbb{N}$ and in variables $x'_1, \ldots, x'_\ell$, there are coefficients $c_A \in \mathbb{R}$ for $A \subseteq \{x'_1, \ldots, x'_\ell\}$ such that*

$$f(x'_1, \ldots, x'_\ell) = \sum_{A \subseteq \{x'_1, \ldots, x'_\ell\}} c_A I_A(x'_1, \ldots, x'_\ell).$$

Applying this to the utility functions, it follows directly that, for every $A \subset \mathcal{X}$, there are coefficients $\lambda_{A,B}$ for $B \subseteq A$ such that for all alternatives $o \in \underline{\mathcal{X}}$ we have

$$g_A(o) = \sum_{B \subseteq A} \lambda_{A,B} I_B(o).$$

We get by summing the $g_A$ that there are coefficients $\lambda_B$ such that for all $o \in \underline{\mathcal{X}}$

$$g_\varphi(o) = \sum_{B \subset \mathcal{X}} \lambda_B I_B(o). \tag{1}$$

By subtracting values in some of the $g_A$, we may assume w.l.o.g. that $g_\varphi(o) = 0$ for all $o \neq o_{\mathcal{X}}$.

We claim that, for all $B \subset \mathcal{X}$, we have $\lambda_B = 0$. We show this by induction on the size of $B$. For $B = \emptyset$, we have with (1) that $0 = g(o_\emptyset) = \lambda_\emptyset I_\emptyset(o_\emptyset) = \lambda_\emptyset$. For non-empty $B \subset \mathcal{X}$, we have $g_\varphi(o_B) = \sum_{C \subset \mathcal{X}} \lambda_C I_C(o_B) = \sum_{C \subseteq B} \lambda_C I_C(o_B)$. However, by the induction hypothesis, we know that for $C \subset B$ we have $\lambda_C = 0$, so $0 = \lambda_B I_B(o_B) = \lambda_B$.

Plugging the $\lambda_B$ into (1), we get that $g_\varphi(o_{\mathcal{X}}) = 0$ which contradicts the assumption that in $\succeq$ the alternative $o_{\mathcal{X}}$ strictly dominates all others.

For the second statement, consider $k$ binary attributes $A_1, \ldots, A_k$ such that $a_1 \ldots a_k \succ \bar{a}_1 \ldots \bar{a}_k$. Extend this to an arbitrary complete preference relation such that for all other alternatives $o$ we have $o \succ a_1 \ldots a_k$. Clearly, any such order can be expressed as a $\mathsf{GAI}_k$ by simply giving all alternatives $o$ a utility that yields this order in a single $k$-ary function $g_{\{A_1, \ldots, A_k\}}$. We claim that this order cannot be expressed by a (k-1)-CP. Assume this were false, so there is a set of preference statements defining the order and in which the set of swapped attributes never contains more than $k-1$ attributes. In particular, there is such a statement $\alpha \mid V : w \geq w'$ that sanctions $a_1 \ldots a_k \succ \bar{a}_1 \ldots \bar{a}_k$ (this comparison cannot be obtained by transitivity, since all other alternatives have a utility that is strictly greater than that of $a_1 \ldots a_k$). By assumption $w$ cannot contain all attributes, so there is one attribute, say w.l.o.g. $A_1$ that does not appear in $w$. If $A_1$ is not in $V$, then, by definition, applying the statement cannot swap the value of $A_1$, so it cannot justify $a_1 \ldots a_k \succ \bar{a}_1 \ldots \bar{a}_k$. So $A_1$ must appear in $V$. Then $A_1 \notin \mathsf{Var}(\alpha)$, thus the statement also sanctions $a_1 a_2 \ldots a_k \succ a_1 \bar{a}_2 \ldots \bar{a}_k$ which contradicts the order we want to define. So as we claimed, $\succ$ is not defined by any (k-1)-CP. □

# 3 Proofs for Section 5 (Succinctness)

**Proposition 9.** *Any complete LPT can be transformed in polytime and space into an equivalent GAI.*

*Proof.* A complete LP tree $\varphi$ induces a linear order over $\underline{\mathcal{X}}$, thus we can define the *rank* of alternative $o$ w.r.t. $\succeq_\varphi$: $\mathsf{rank}(\varphi, o) = 1+$ the number of alternatives strictly preferred to $o$, so that the most preferred alternative has rank 1, the least preferred has rank $|\underline{\mathcal{X}}|$:

$$\mathsf{rank}(\varphi, o) = 1 + |\{o' \in \underline{\mathcal{X}} \mid o' \succ_\varphi o\}|.$$

[28] explain how $\mathsf{rank}(\varphi, o)$ can be decomposed as a weighted sum of "local" ranks associated to the nodes of $\varphi$:

$$\mathsf{rank}(\varphi, o) = 1 + \sum_{N \in \mathsf{nodes}(\varphi)}^{\alpha : \geq \in \mathsf{CPT}(N)} [\![ o[\mathsf{Inst}(N)] = \mathsf{inst}(N) \wedge o \models \alpha ]\!] \times \big(r(\geq, o[\mathsf{Var}(N)]) - 1\big) \times |\underline{\mathsf{Desc}(N)}|$$

where :

- $\mathsf{nodes}(\varphi)$ denotes the set of nodes of $\varphi$;
- $[\![ o[\mathsf{Inst}(N)] = \mathsf{inst}(N) \wedge o \models \alpha ]\!]$ is an indicator function, that equals 1 when the condition $o[\mathsf{Inst}(N)] = \mathsf{inst}(N) \wedge o \models \alpha$ is true; that is, when $N$ is on the branch of $\varphi$ that corresponds to $o$, and $\alpha : \geq$ is the rule that orders at $N$ alternatives that have same values as $o$ for the attributes in the ancestor nodes of $N$; and equals 0 otherwise;
- $r(\geq, o[\mathsf{Var}(N)])$ denotes the rank in $\underline{\mathsf{Var}(N)}$ with respect to $\geq$ of the instantiation given by $o$ to $\mathsf{Var}(N)$; so that $r(\geq, o[\mathsf{Var}(N)]) - 1$ is the number of subtrees rooted at children of $N$ that are less preferred than $o$ at $N$;
- $\mathsf{Desc}(N) = \mathcal{X} - (\mathsf{Anc}(N) \cup \mathsf{Var}(N))$ is the set of attributes that appear below $N$ in that branch, so that $|\underline{\mathsf{Desc}(N)}|$ is the number of instantiations that are "contained" in every subtree of $\varphi$ rooted at any one child of $N$.

Thus we can define, for every node $N$ of $\varphi$, and every rule $\alpha : \geq \in \mathsf{CPT}(N)$, a sub-utility $u_{N,\alpha}$ as follows:

$$u_{N,\alpha}(o) = \begin{cases} \big(r(\geq, o[\mathsf{Var}(N)]) - 1\big) \times |\underline{\mathsf{Desc}(N)}| \text{ if } o[\mathsf{Inst}(N)] = \mathsf{inst}(N) \ \&\ o \models \alpha \\ 0 \text{ otherwise} \end{cases}$$

and define a utility $u_\varphi$ that orders the alternatives as $\varphi$ as follows:

$$u_\varphi = - \sum_{N \in \mathsf{nodes}(\varphi)}^{\alpha : \geq \in \mathsf{CPT}(N)} u_{N,\alpha}$$

The number of non-null entries in the table of every $u_{N,\alpha}$ is equal to $\underline{\mathsf{Var}(N)} - 1$, which also corresponds to the space needed to represent the linear order $\geq$ of the rule $\alpha : \geq$. Assuming that $\underline{\mathsf{Var}}(N)$ (resp. $\mathsf{Desc}(N)$) contains $p$ (resp. $q$) attributes, the largest entry cannot be larger than $d^{p+q} \leq d^n$, where $d$ is the size of the largest attribute domain, so the number of digits needed for representing the non-null values is polynomial in $n$ and $d$. Thus the size of the representation of $u_\varphi$ is polynomial in the size of $\varphi$. □

# 4 Proofs for Section 6 (Queries)

**Proposition 11.** *$\succ$-comparison and $\bowtie$-comparison are NP-hard for the language of acyclic CP-nets, and tractable for polytree CP-nets.*

Proposition 11 is proved using a result about the ordering query introduced in [10]: it is a particular case of the top-$p$ that is recalled in section 6.4.

Ordering Given $S \subseteq \underline{\mathcal{X}}$ with $|S| = 2$, and $\varphi$, return some $o \in S$ such that $o' \not\succ_\varphi o$, where $o'$ is the other element of $S$.

Note that when $S$ contains exactly two elements, at least one of them is not strictly dominated by the other; it the two elements in $S$ are incomparable, then the ordering query may return any one of them.

*Proof.* Note that for ayclic CP-nets (and thus for polytree CP-nets), $\succeq$-comparison and $\succ$-comparison are "almost" equivalent, in the sense that for different alternatives $o$ and $o'$, $o \succ_\varphi o'$ iff $o \succeq_\varphi o'$ (because acyclic CP-nets are consistent). In particular, $\succeq$-comparison can be reduced to $\succ$-comparison for consistent languages, thus $\succ$-comparison is NP hard for acyclic CP-nets because $\succeq$-comparison is hard for this language [10, Theorems 15, 16].

$\succ$-comparison can also be reduced, still for languages that guarantee consistency, to $\bowtie$-comparison: consider alternatives $o \neq o'$, in order to check if $o \succ o'$ we can ask if $o \bowtie o'$: if the answer is "yes", then $o \not\succ o'$; if the answer is "no", ask the ordering query for $S = \{o, o'\}$: the answer must be, in polynomial time for acyclic CP-nets [10, Theorem 5], that $o \not\succ o'$ or $o' \not\succ o$: if the answer is that $o \not\succ o'$, it answers the initial query; if the answer is that $o' \not\succ o$, since we know that $o \succ o'$ or $o' \succ o$ because $o$ and $o'$ are not incomparable and $\succeq$ is antisymmetric, it must be the case that $o \succ o'$.

Finally, $\succeq$-comparison is tractable for polytree CP-nets, and two calls of this query at most can answer $\succ$-comparison and $\bowtie$-comparison.

□

**Proposition 12.** *EQUIVALENCE is coNP-hard for 1-CP⊭/∧⊀, and for 1-LPT∧, both restricted to binary attributes.*

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

1-CP⊭$\mathcal{P}$ is 1-CP⊭ restricted to those statements such that the condition is in $\mathcal{P}$

1-LPT$\mathcal{P}$ is 1-LPT restricted to those LP-trees such that the condition of every rule is in $\mathcal{P}$.

The proof of the proposition is based on the following lemma, which formalizes the intuition suggested by Example 9.

**Lemma 2.** *Given a propositional language $\mathcal{P}$ closed for conjunction, EQUIVALENCE for $\mathcal{P}^\vee$ (in the sense of propositional logic), reduces to EQUIVALENCE for 1-CP⊭$\mathcal{P}$ restricted to acyclic formulas, and to EQUIVALENCE for 1-LPT$\mathcal{P}$.*

*Proof.* Consider two formulas $\alpha = \bigvee_{i\in I} \alpha_i$ and $\alpha' = \bigvee_{i\in I'} \alpha'_i$ over a set $\mathcal{X}$ of binary attributes, with all $\alpha_i$'s and $\alpha'_i$'s in $\mathcal{P}$; take some binary attribute $X \notin \mathcal{X}$, with values $x$ and $\bar{x}$, and let $\varphi = \{\alpha_i : x \geq \bar{x} \mid i \in I\}$ and $\varphi' = \{\alpha'_i : x \geq \bar{x} \mid i \in I'\}$. Note that $\varphi, \varphi' \in$ 1-CP⊭$\mathcal{P}$, that they are acyclic, and that they can be computed in time polynomial in $|\alpha| + |\alpha'|$. Then $\varphi^* = \{(ox, o\bar{x}) \mid o \in \underline{\mathcal{X}}, o \models \alpha\}$ and for every $o_1, o_2 \in \underline{\mathcal{X}}$, for every $x_1, x_2 \in \underline{X}$, $o_1x_1 \succeq_\varphi o_2x_2$ if and only if $o_1 = o_2$, $x_1 = x$, $x_2 = \bar{x}$ and $o_1 \models \alpha$; similarly, $o_1x_1 \succeq_{\varphi'} o_2x_2$ if and only if $o_1 = o_2$, $x_1 = x$, $x_2 = \bar{x}$ and $o_1 \models \alpha'$. Thus $\alpha \equiv \alpha'$ if and only if for every $o \in \underline{\mathcal{X}}$, $o \models \alpha \Leftrightarrow o \models \alpha'$, iff for every $o \in \underline{\mathcal{X}}$, $ox \succeq_\varphi o\bar{x} \Leftrightarrow ox \succeq_{\varphi'} o\bar{x}$, if and only if $\varphi \equiv \varphi'$.

Similarly, we can define two linear 1-LP-trees $\psi$ and $\psi'$ as follows: the top $|\mathcal{X}|$ nodes are labelled with attributes from $\mathcal{X}$, in any order and with no rule; then there is one node labelled with $X$, and the same preference rules as above. □

**Proposition 14.** *$\succeq$-CUT COUNTING and $\succ$-CUT COUNTING are #P-hard for CP-nets and acyclic CP-nets.*

*Proof.* Remember that a vertex cover in a graph $G = (V, E)$ is a set $S \subseteq V$ such that for each edge $uv \in E$ we have $u \in S$ or $v \in S$. The problem #VertexCover is, given a graph $G$, to count its vertex covers. #VertexCover is well-known to be #P-hard [47], so we will use a reduction from #VertexCover to $\succ$-CUT-COUNTING to establish the claim.

So let $G = (V, E)$ be a graph. For every vertex $v \in V$ we introduce an attribute $V_v$ and for every edge $e = uv \in E$ we introduce an attribute $E_{uv}$. Note that for convenience we denote $E_{uv}$ also by $E_e$ sometimes. Finally, we introduce attributes $D_i$ for $i \in [|V| + |E| + 1]$. The attributes $V_v$ have no parents. Let $e_1, \ldots, e_m$ be an order of the edges in $E$ where $e_i = u_iv_i$. For $i > 1$ the attribute $E_{e_i}$ has the parents $V_{u_i}, V_{v_i}, E_{e_{i-1}}$. The attribute $E_{e_1}$ has parents $V_{u_1}, V_{v_1}$. Finally, the attributes $D_j$ all have the single parent $E_{e_m}$.

We next describe the CPTs for all attributes: all attributes have values in $\{0, 1\}$. All $V_v$ have the order $1 \geq 0$. For all $D_i$, we have that if $E_{e_m}$ has value 0 then the order is $0 \geq 1$ and if $E_{e_m}$ has value 1, then $1 \geq 0$. For $E_{e_1}$ we have the order $1 \geq 0$ if and only if at least one of $V_{u_1}, V_{v_1}$ has value 1 and the order $0 \geq 1$ otherwise. Finally, for $i > 1$, we have the order $1 \geq 0$ if and only if $E_{i-1}$ has value 1 and at least one of $V_{u_i}, V_{v_i}$ has value 1. Otherwise $E_{e_i}$ has the order $0 \geq 1$. Call the resulting CP-net $\varphi$.

Note that one can easily see that no attribute in an increasing flipping sequence can ever be flipped back to 0 from 0: for the attributes $V_v$ this is immediate. For the $E_{e_j}$ it follows with an easy induction and the fact that it is true for the $V_v$. For the $D_j$ finally it follows from the fact that $E_{e_m}$ can never flip back to 0.

Let $o$ be the assignment that assigns 0 to all attributes. Let $o'$ be an assignment such that $o'$ is reachable from $o$ by an increasing flipping sequence, or equivalently $o' \succeq_\varphi o$. We claim that if $E_{e_m}$ has value 1 in $o'$, then $S := \{v \in V \mid o'(V_v) = 1\}$ is a vertex cover of $G$. To see this, first observe that in fact all $E_{e_j}$ must take the value 1 in $o'$: to flip $E_{e_j}$ to 1, we must have flipped $E_{j-1}$ before (if it exists) and since we can never flip back to 0, $E_{j-1}$ must take 1 in $o'$. But then when we flipped $E_{e_j}$ to 1, at least one of $V_{v_j}, V_{u_j}$ must have had value 1 and since we cannot flip it back, in $o'$ one of $V_{v_j}, V_{u_j}$ must have value 1. So for every $e_j$ we have that one of $V_{v_j}, V_{u_j}$ must have value 1 which proves that $S$ is a vertex cover as claimed.

Now for $S \subseteq V$, define $o_S$ to be the assignment that assigns 1 to $V_v$ if and only if $v \in S$, assigns 1 to all $E_{e_i}$ and assigns 0 to all $D_j$. We claim that $o_S \succeq_\varphi o$ if and only if $S$ is a vertex cover of $G$. First note that if $S$ is a vertex cover, we can flip all $V_v$ accordingly and then iteratively flip all $E_{e_j}$ to reach $o_S$. The other direction is clear from what we saw above, observing that $E_{e_m}$ takes value 1 in $o'$.

Observe that for every $o_S$, where $S$ is a vertex cover, we can flip an arbitrary subset of the $D_j$ to 1 to reach an assignment $o' \succeq_\varphi o_S \succeq_\varphi o$. Note that for different vertex covers $S_1, S_2$, there is no such $o' \succeq o_{S_1}$ and $o' \succeq o_{S_2}$ since $o_{S_1}$ and $o_{S_2}$ differ on the $V_v$ and in the construction of the $o'$ from the $o_S$ we do not change those. It follows that

$$\{o' \mid o' \succeq_\varphi o'', o''(E_{e_m}) = 1\} = \bigcup_{S \text{ vertex cover of } G} \{o' \mid o' \succeq_\varphi o_S, \forall V_v : o_S(V_v) = o'(V_v)\}$$

and the union is disjoint. Now for every vertex cover $S$ of $G$, we have

$$|\{o'' \mid o'' \succeq_\varphi o_S, \forall V_v : o_S(V_v) = o''(V_v)\}| = 2^{|V|+|E|+1}.$$

Let $s$ be the number of vertex covers of $G$. It follows that

$$\begin{aligned}|\{o' \mid o' \succeq_\varphi o\}| &= |\{o' \mid o' \succeq_\varphi o, o'(E_{e_m}) = 0\}| + |\{o' \mid o'' \succeq_\varphi o, o''(E_{e_m}) = 1\}| \\ &= |\{o' \mid o' \succeq_\varphi o, o'(E_{e_m}) = 0\}| + s2^{|V|+|E|+1}.\end{aligned}$$

Now since in no $o'$ with $o'(E_{e_m}) = 0$ any of the $D_j$ can be flipped to 1 in any increasing flipping sequence, we have

$$|\{o' \mid o' \succeq_\varphi o, o'(E_{e_m}) = 0\}| < 2^{|V|+|E|},$$

since such $o'$ have only $|V| + |E| - 1$ attributes with domain $\{0, 1\}$ which are not forced to be constant 0. Consequently, $s$ can be inferred from $|\{o' \mid o' \succeq_\varphi o\}|$ by a single integer division which completes the reduction.

This proves that $\succeq$-CUT COUNTING for acyclic CP-nets is as hard as #VERTEXCOVER; this holds for $\succ$-CUT COUNTING since in the case of acyclic CP-nets, $\succeq$ is antisymmetric. And this hardness result extends to the larger class of CP-nets. □

**Proposition 15.** *$\succ$-CUT COUNTING is #P-complete for GAI, $\text{GAI}_k$ and $\text{GAI}_1$.*

*Proof.* For containment in #P, observe that all elements in $\mathsf{CUT}^{\succ,o}(\varphi)$ have polynomial size, so we can easily guess them and compare in polynomial time to $o$ since $\succ$-COMPARISON can be solved in polynomial time for GAI.

For hardness, we reduce from the problem #SUBSETSUM which is, given a set $S = \{s_1, \ldots, s_n\}$ of positive integers and an additional integer $k$, to count the number of subsets of $S$ that sum up to $k$. #SUBSETSUM is well-known to be #P-complete, see e.g. [27]. It will be convenient to work with a slight variant which we call $\#\text{SUBSETSUM}_>$ and which is, given the same type of input as for #SUBSETSUM, to count the number of subsets of $S$ which sum up to a value greater than $k$. There is an easy oracle reduction from #SUBSETSUM to $\#\text{SUBSETSUM}_>$: given an input $S$, $k$, call an oracle for $\#\text{SUBSETSUM}_>$ on the two inputs $S, k-1$ and $S, k$. Then the answer to the #SUBSETSUM instance is the difference of the answers of the oracle calls. It follows that $\#\text{SUBSETSUM}_>$ is #P-hard.

We now reduce $\#\text{SUBSETSUM}_>$ to $\succ$-CUT COUNTING for $\text{GAI}_1$. So let $S = \{s_1, \ldots, s_n\}$ and $k$ be an instance of $\#\text{SUBSETSUM}_>$. We construct $n$ functions $g_i(X_i)$ for $i = 1, \ldots, n$ where $\underline{X_i} = \{0, 1\}$. We set $g_i(0) = 0$ and $g_i(1) = s_i$. Moreover, we add a function $g_Y(Y)$ where $\underline{Y} = \{0, 1\}$ and $g_Y(0) = 1$ and $g_Y(1) = k$. Set $\varphi = \{g_1, \ldots, g_n, g_Y\}$ and $\mathcal{X} = \{X_1, \ldots, X_n, Y\}$. This completes the construction of the GAI. Call the induced relation $\succeq$.

To complete the reduction, let $o^* \in \underline{\mathcal{X}}$ be the alternative that sets $Y$ to 1 and all other attributes to 0. Then $g_\varphi(o^*) = k$. Moreover, for $o \in \underline{\mathcal{X}}$, we have that $g_\varphi(o) > k$ if and only if $o(Y) = 1$ and there is an $i \in [n]$ such that $o(X_i) = 1$—i.e. the set $\{i \in [n] \mid o(X_i) = 1\}$ is non-empty—, or $o(Y) = 0$ and $\sum_{i\in[n]} g_i(o[X_i]) = \sum_{i\in[n]:\, o(X_i)=1} s_i > k$. Note that there are $2^n - 1$ alternatives of the former type, corresponding to the non-empty subsets of $[n]$, so the number of subsets of $S$ that sum up to values greater than $k$ is $|\,\mathsf{CUT}^{\succ,o^*}(\varphi)\,| - 2^n + 1$. Thus, one oracle call to $\succ$-CUT COUNTING allows solving $\#\text{SUBSETSUM}_>$ in polynomial time which completes the reduction. □

**Proposition 16.** *$\succeq$-CUT EXTRACTION is tractable for CP, and $\succ$-CUT EXTRACTION is tractable for acyclic CP-nets. $\succ$-CUT COUNTING and $\succ$-CUT EXTRACTION are PSPACE-hard for 1-CP$\not\succ\wedge$. $\succ$-CUT EXTRACTION, $\succeq$-CUT EXTRACTION and $\succ$-CUT COUNTING are tractable for LP-trees.*

*Proof.* $\succeq$-CUT EXTRACTION is easy for CP: given $o$ and $\varphi$, in order to return an element of $\mathsf{CUT}^{\succeq,o}(\varphi)$, it is sufficient to find one statement in $\varphi$ which sanctions an improving swap for $o$. For acyclic CP-nets (and more generally for any language that guarantees consistency), $\succ$ is the asymmetric part of $\succeq$, thus $\succ$-CUT EXTRACTION is equivalent to $\succeq$-CUT EXTRACTION and is tractable.

Note that alternative $o$ is undominated iff $\mathsf{CUT}^{\succ,o}(\varphi) = \emptyset$, iff $|\,\mathsf{CUT}^{\succ,o}(\varphi)\,| = 0$; therefore, $\succ$-CUT COUNTING and $\succ$-CUT EXTRACTION are at least as hard as UNDOMINATED CHECK, they are therefore PSPACE-hard for 1-CP$\not\succ\wedge$.

Finally, $\succ$-CUT EXTRACTION, $\succeq$-CUT EXTRACTION and $\succ$-CUT COUNTING are tractable for LP-trees: for LP-tree $\varphi$, given $o$, in order to find some $o'$ such that $o' \succ_\varphi o$ (resp. $o' \succeq_\varphi o$), it is possible to traverse the tree, starting at the root, guided by the values assigned by $o$, until reaching a node where the value(s) assigned by $o$ for the attributes at that node is/are strictly dominated (resp. dominated) by other values at that node. Also, when going down $\varphi$ in the branch that corresponds to $o$, it is possible, at each node $N$ encountered, labelled with $T$, to count the number of $t'$ in $\underline{T}$ such that $t > o[t']$ (according to the preference rule $\beta : \geq^\beta$ at $N$ such that $o \models \beta$), and to multiply this number by the sizes of the domains of the attributes that have not been encountered yet; adding these sums of products along the branch will give the number of alternatives $o$ such that $o' \succ_\varphi o$.

□

**Proposition 17.** *$\succeq$-CUT EXTRACTION, $\succ$-CUT EXTRACTION, UNDOMINATED CHECK, and UNDOMINATED EXTRACT are tractable for $\text{GAI}_1$.*

*Proof.* Given a GAI $\varphi$, we can simply choose the values for the attributes in such a way that the utilities are maximized. Since the utilities are unary, this leads to a consistent and thus also maximal alternative $o^*$. For $\succeq$-CUT EXTRACTION, $o^*$ is always a valid output, so it solves the problem independent of the additional input alternative $o$. For $\succ$-CUT EXTRACTION,

we check if $g_\varphi(o^*) > g_\varphi(o)$. If so, we return $o^*$ again. Otherwise, due to the maximality of $o^*$, we have $g_\varphi(o^*) = g_\varphi(o)$ and thus there is no alternative strictly dominating $o$ and thus no valid output.

For UNDOMINATED CHECK we have that $o$ is undominated if and only if $g_\varphi(o) = g_\varphi(o^*)$ which we can check efficiently. Finally, for undominated extract we can simply return $o^*$. □

**Proposition 18.** *$\succeq$-CUT EXTRACTION and $\succ$-CUT EXTRACTION are NP-complete for GAI$_k$ for $k \geq 2$ and GAI. UNDOMINATED CHECK is coNP-complete and UNDOMINATED EXTRACT is NP-hard for GAI$_k$ for $k \geq 2$ and for GAI.*

*Proof.* Containment in NP, resp. coNP, is easy to see in all cases since alternatives can be compared efficiently

We show hardness for all problems by reduction from 3-COLORING which is, given a simple, undirected graph $G = (V, E)$, to decide if there is an assignment $c : V \to \{r, g, b\}$ such that for all edges $uv \in E$ we have $c(v) \neq c(u)$. The mapping $c$ is called a *coloring* and it is said to be valid if it satisfies the condition on the edges. 3-COLORING is well-known to be NP-complete, see e.g. [43, Theorem 9.8].

We use the same construction of a GAI$_2$ $\varphi_G$ from a graph $G$ for all problems. So let a graph $G$ be given in which w.l.o.g. every vertex has at least two neighbors (vertices with fewer than two neighbors can iteratively be deleted without changing the answer to the 3-COLORING question). We also assume that $G$ is connected; if it is not, we can connect the different connected components iteratively by adding edges without changing the answer to the 3-COLORING question. We construct a GAI$_2$ representation as follows: for every vertex $v \in V$, we introduce an attribute $X_v$ with domain $\underline{X_v} = \{r, g, b, d\}$. For every edge $e = uv$, we construct a utility function $g_{uv}$ in the variables $X_u, X_v$ and which takes value 1 on inputs $rb, rg, br, bg, gr, gb, dd$ and 0 on all other inputs. Setting $\varphi_G = \{g_{uv} \mid uv \in E\}$ completes the construction of the GAI $\varphi_G$. Let $\succeq$ be the order that $\varphi_G$ induces.

We first show hardness for $\succeq$-CUT EXTRACTION. To this end, let $o_d$ be the alternative in which all attributes take value $d$. Then all $g_{uv}$ evaluate to 1 on $o_d$, so $g_{\varphi_G}(o_d) = |E|$. Now consider $o \in \mathsf{CUT}^{\succeq, o_d}$. Assume first that some attribute of $o$ takes value $d$. Since not all attributes can take value $d$ and $G$ is connected, there must be an edge $uv$ such that $d = o(u) \neq o(v)$. Then $g_{uv}(o) = 0$ and $g_{\varphi_G}(o) < |E| = g_{\varphi_G}(o_d)$, so $o \notin \mathsf{CUT}^{\succeq, o_d}$ which contradicts the choice of $o$. Consequently, we must have that all $X_v$ takes values in $\{r, g, b\}$ in $o$. Moreover, for all $uv \in E$, we must have that $o(X_u) \neq o(X_v)$. Thus, setting $c(v) = o(X_v)$ for all $v \in V$ yields a valid coloring of $G$. So if there is an element in $\mathsf{CUT}^{\succeq, o_d}$, the graph $G$ is 3-colorable. The other way round, if $G$ has a valid 3-coloring $c$, then defining $o$ for all $X_v$ by $o(X_v) = c(v)$ yields an alternative in $\mathsf{CUT}^{\succeq, o_d}$. This shows NP-hardness of $\succeq$-CUT EXTRACTION for GAI$_2$ and thus for all GAI$_k$ with $k \geq 2$ and GAI.

The reasoning for $\succ$-CUT EXTRACTION is similar. The only difference is that for one arbitrary edge $uv$ we set $g_{uv}(d, d)$ to 0. Call the resulting GAI $\varphi'_G$. We have $g_{\varphi'_G}(o_d) = |E| - 1$. The rest of the reduction and the argument for completeness is exactly as that for $\succeq$-CUT EXTRACTION.

For UNDOMINATED CHECK, observe that $o_d$ is dominating for $\varphi'_G$ if and only if there is no alternative $o$ with $g_{\varphi'_G}(o_d) < g_{\varphi'_G}(o)$. Reasoning as above, this is exactly the case if and only if $G$ has no valid 3-coloring. Thus UNDOMINATED CHECK is coNP-hard.

Finally, to show hardness of UNDOMINATED EXTRACT, multiply all utility values in $g_G$ by 2. Then, for one arbitrary edge $uv$ set $g_{uv}(d, d)$ to 1. Call the resulting GAI $\varphi''_G$. Then we have $g_{\varphi''_G}(o_d) = 2|E| - 1$. Moreover, for all alternatives $o$ encoding a valid 3-coloring, we have $g_{\varphi''_G}(o) = 2|E|$. Finally, for all other alternatives $o$, we have $g_{\varphi''_G}(o) \leq 2|E| - 2$. So in any case an undominated alternative is either a valid 3-coloring of the graph $G$ or $o_d$, hence $o_d$ is undominated if and only if $G$ is not 3-colorable which shows that UNDOMINATED EXTRACT is NP-hard. □

# 5 Proofs for Section 7 (Transformations)

**Proposition 19.** *Given a preorder $\succeq$ over $\underline{\mathcal{X}}$, given $V \subseteq \mathcal{X}$, let $U \subseteq \mathcal{X} \setminus V$. If $v, v' \in \underline{V}$ and $v \succeq^{\downarrow V}_{\text{w.opt.}} v'$, then there is some $u \in \underline{U}$ such that for no $u' \in \underline{U}$ it holds that $u'v' \succ uv$.*

*Proof.* Assume that $v \succeq^{\downarrow V}_{\text{w.opt.}} v'$, and let $u \in \underline{U}$ be such that for no other $u_1 \in \underline{U}$ it is the case that $u_1 v \succ uv$: such a $u$ must exist because $\underline{U}$ is finite; if there is some completion $u'$ of $v'$ such that $u'v' \succ uv$, then, since $v \succeq^{\downarrow V}_{\text{w.opt.}} v'$, there must be some $u_1 \in \underline{U}$ such that $u_1 v \succeq u'v'$, but then $u_1 v \succ uv$, which is a contradiction. □

**Proposition 20.** *All four projections defined above are equivalent for the 1-GAI language and the language that contains complete LP-trees of 1-LPT$^{\mathsf{lin}}$, and can be computed in polynomial time.*

*Proof.* Let $\varphi \in$ 1-GAI, let $X$ be any attribute, $\varphi$ is dfined by a function of the form $g(o) = g_X(o[X]) + \sum_{Y \neq X} g_Y(o[Y])$. Let $\succeq$ denote the associated weak order over the set of alternatives. Let $\psi$ be defined by $h(o) = \sum_{Y \neq X} g_Y(o[Y])$. We

show that $o \succeq_{\text{low}} o'$ iff $o \succeq_{\text{up}} o'$ iff $o \succeq_{\text{w.opt.}} o'$ iff $o \succeq_{\text{s.opt}} o'$ iff $o \succeq_{\psi} o'$:

$$
\begin{aligned}
o \succeq_{\psi} o' &\Leftrightarrow h(o) \geq h(o') \\
&\Leftrightarrow (\forall x \in \underline{X} : g_X(x) + h(o) \geq g_X(x) + h(o')) \Leftrightarrow o \succeq_{\text{low}} o' \\
&\Leftrightarrow (\exists x \in \underline{X} : g_X(x) + h(o) \geq g_X(x) + h(o')) \Leftrightarrow o \succeq_{\text{up}} o' \\
&\Leftrightarrow (\forall x' \in \underline{X} \exists x \in \underline{X} : g_X(x) + h(o) \geq g_X(x') + h(o')) \text{ (take } x = x' \text{)} \\
&\qquad \Leftrightarrow o \succeq_{\text{w.opt.}} o' \\
&\Leftrightarrow (\forall x' \in \underline{X} : g_X(\operatorname*{argmax}_{x' \in \underline{X}} g_X(x')) + h(o) \geq g_X(x') + h(o')) \\
&\qquad \Leftrightarrow o \succeq_{\text{s.opt.}} o'
\end{aligned}
$$

Suppose now that $\varphi$ is a complete, linear LP-tree in 1-LPT over $\mathcal{X}$, and let $X \in \mathcal{X}$. Let $\psi$ be the LP-tree defined by removing node $X$, redirecting the parent of $X$ to the unique child of $X$ when $X$ is an internal node of $\varphi$. Consider alternatives $o, o' \in \mathcal{X} \setminus X$, let $x \in \underline{X}$. Let $Y$ be the attribute that decides the pair $\{o, o'\}$ in $\psi$ then $o \succeq_{\psi} o'$ iff $o[Y] > o'[Y]$ in $\mathsf{CPT}(\overline{Y})$. Suppose first that $Y$ is an ancestor of $X$ in $\varphi$, then $\{ox, o'x'\}$ is decided at $Y$ in $\varphi$ forall $x, x \in \underline{X}$, thus $o[Y] > o'[Y] \Rightarrow (\forall x, x' \in \underline{X} : ox \succeq_{\varphi} o'x') \Rightarrow o \succeq_{\pi} o'$ for all four projections; and $o \succeq_{\pi} o' \Rightarrow o \succeq_{\psi} o'$ for any of the four projections. Suppose now that $X$ is an ancestor of $Y$ in $\varphi$, then for all $x, x' \in \underline{X}$ 1) the pair $\{ox, o'x\}$ is decided at $Y$, whereas 2) pair $\{ox, o'x'\}$ with $x \neq x'$ is decided at $X$. From 1) it follows that $o \succeq_{\psi} o' \Leftrightarrow o \succeq_{\text{low}} o' \Leftrightarrow o \succeq_{\text{up}} o'$; moreover, let $x_0$ be the optimal value for $\underline{X}$ in $\mathsf{CPT}(X)$ in $\varphi$ (which exists and is unique because $\varphi$ is a complete LP-tree, thus the linear order over $\underline{X}$ in $\mathsf{CPT}(X)$ is a linear order), then $ox_0 \succeq_{\varphi} o'x'$ for all $x' \in \underline{X}$, thus $o \succeq_{\text{s.opt.}} o'$; [5] mention that $o \succeq_{\text{s.opt}} \Rightarrow o \succeq_{\text{w.opt}} o'$; lastly, if $o \succeq_{\text{w.opt}} o'$ then there exists $x$ such that $ox \succeq_{\varphi} o'x_0$, and $ox_0 \succeq_{\varphi} ox$, thus $ox_0 \succeq_{\varphi} o'x_0$, thus $o[Y] > o'[Y]$. □

**Proposition 21.** *If conditioning can be done in polynomial time for language $\mathcal{L}$ but the extraction of an undominated alternative is NP-hard, then the strong optimistic projections cannot be computed in polynomial time for $\mathcal{L}$ (unless P = NP).*

*Proof.* We assume that for any preorder expressed in the language $\mathcal{L}$, any strong optimistic projection leads to a preorder that again can be expressed in $\mathcal{L}$. If this is not true, then the statement of the theorem is trivially true, even without the assumption $\mathsf{P} \neq \mathsf{NP}$.

We give an algorithm that, given a preorder $\succeq$ encoded in $\mathcal{L}$, computes an undominated alternative $o$ in polynomial time, assuming polynomial time algorithms for conditioning and computation of strong optimistic projection. The algorithm considers the attributes of $\succeq$ in sequence, say from $V_1$ to $V_n$. The value $v_1$ of $V_1$ is obtained by projecting $\succeq$ onto $V_1$ - then an undominated value $v_1$ is chosen for $V_1$; indeed, $v_1 \succeq_{\text{s.opt.}}^{\downarrow\{V1\}} v_1'$ means that there exist an assignment $v$ of $\{V_2, \ldots, V_n\}$ such that $v_1.v \succeq v_1'.v'$ for all $v' - V_1 = v_1$ in one of the non dominated solutions. Then the original formula is conditioned: value $v_1$ is assigned to $V_1$ and the procedure is repeated for the next variable - and this until all the variables have been assigned. So, if a language offers the conditioning transformation in polytime but not the undominated query, there cannot be any polynomial algorithm for performing the strong optimistic projection within this language (unless $P = NP$). □

**Proposition 22.** *The strong (resp. weak) projection cannot be computed in polytime for GAI and GAI$_k$ ($k > 1$) (unless P = NP).*

*Proof.* The extraction of an undominated alternative is NP-hard for GAI and GAI$_k$, $k > 1$ (Proposition 18) while conditioning can be done in polytime for these languages. From Proposition 21 we deduce the strong optimistic projection cannot be computed in polytime unless $P = NP$. Because $GAI$'s encode complete and transitive relations, the strong and weak optimistic projections are identical [6] - hence the weak optimistic projection cannot be computed in polytime unless $P = NP$.

□