Logical conditional preference theories

Cristina Cornelio
University of Padova
cornelio@math.unipd.it

Andrea Loreggia
University of Padova
IBM T.J. Watson Research Center
andrea.loreggia@gmail.com

Vijay Saraswat
IBM T.J. Watson Research Center
vijay@saraswat.org

Abstract

This paper introduces a comprehensive new framework for conditional preferences: *logical conditional preference theories* (LCP theories). To express preferences, the user specifies arbitrary (constraint) Datalog programs over a binary ordering relation on outcomes. We show how LCP theories unify and generalize existing conditional preference proposals, and leverage the rich semantic, algorithmic and implementation frameworks of Datalog.

1 Introduction

Qualitative conditional preferences on combinatorial domains are an important area of study. The main framework is *CP-nets* [Boutilier *et al.*, 2004b]. Given a finite set of features, each with a finite domain of values, the user specifies her preference order over the domain of each feature using rules such as $a\, b : c \succ \bar{c}$ which specify that if the attribute A has value a and attribute B has value b then c is to be preferred to $\bar{c}$ for attribute C. *Outcomes* (that is, assignment of values to all the features) are ordered according to the so-called *ceteris paribus* interpretation: an outcome s is preferred to another outcome t if they differ on the value of just one feature, and the value in s is preferred to the one in t (given the rest of s, t). Thus in CP-nets, preferences are always of the form "I prefer fish to meat, *all else being equal*". The key computational tasks are checking for consistency (preference ordering is acyclic), as well as optimizing and comparing outcomes. CP-nets represent one end of the expressiveness/tractability spectrum. While useful in practice, CP-nets are limited in expressiveness: a rule may specify a preference for exactly one value over another (in the same feature domain). *General CP-nets* [Goldsmith *et al.*, 2008] define CP-nets that can be incomplete or locally inconsistent. *CP-theories* [Wilson, 2004] are a specialized formalism, with its own *ad hoc* syntax (rules are of the form $u : x \succ x'\ [W]$) and semantics, in which preferences may be conditioned on *indifference* to certain features (those in the set W above). For example, one may say "I prefer fish to meat, no matter what the dessert is". *Comparative preference theories* [Wilson, 2009] permit preferences to be defined on a set of features simultaneously. Along another direction, [Boutilier *et al.*, 2004a] and [Domshlak *et al.*, 2006] introduce the idea of adding (hard and soft) constraints to CP-nets. Roughly speaking, the added constraints need to be respected by valid outcomes. This paper develops the fundamental idea that conditional preferences can be directly expressed in standard first order logic, as constrained Datalog theories [Ceri *et al.*, 1989; Kanellakis *et al.*, 1995; Toman, 1998] involving a binary preference relation `d(_,_)` on pairs of outcomes. For instance, in the context of preferences over entrees and desserts, the rule "I prefer fish to meat, all else being equal" may be written as:[1]

```
d(o(fish,X),o(meat,X)).
```

and the rule "I prefer fish to meat, no matter what the dessert is" is written as:

```
d(o(fish,X),o(meat,Y)).
```

More generally, a *Logical Conditional Preference* (LCP) rule is of the form:

```
d(o(X1,...Xn),o(Y1,...,Yn)) :- c, g1,...,gn.
```

where `c` is a constraint (possibly involving equalities), and `g1`, ..., `gn` are possibly recursively defined predicates involving `d/2`. A predicate `dom/2` is defined to be the transitive closure of `d/2` and represents the dominance relation.

```
dom(X,Z):- d(X,Z), outcome(X), outcome(Z).
dom(X,Z):- d(X,Y), dom(Y,Z).
```

The theory can be checked for consistency by simply ensuring that given the clause

```
inconsistent :- dom(X,X).
```

the goal `inconsistent` cannot be established. Hard constraints `C` are specified by adding them to the body of the clause defining legal outcomes.

The fundamental advantage of introducing conditional preference theories as Datalog programs is that Datalog's rich semantic, algorithmic and implementation framework is now available in service of conditional preferences. The semantics of LCP theories is that of (constrained) first-order logic theories. The framework is rich enough to express CP-nets and each of its extensions discussed above, including algorithms for consistency, dominance and optimality (Sec-

[1] The examples in this paper are written in Prolog and run in the XSB Prolog system, using tabling. Note that the clause is strictly speaking not a Datalog clause because it uses a function symbol `o/n`. This function symbol is used just for convenience, and can be eliminated at the cost of increasing the arity of predicates .

tion 3). Using outcomes rather than assertions of preference over individual features permits the formalization of the semantics (e.g. *ceteris paribus* or indifference) internally, as just a certain pattern of quantification over variables. Constraints fit in naturally and do not have to be introduced after the fact in an *ad hoc* fashion as in [Domshlak *et al.*, 2006; Boutilier *et al.*, 2004a; Prestwich *et al.*, 2004]. For example, [Prestwich *et al.*, 2004] provides a dominance algorithm using the notion of a *consistent flipping worsening sequence*: they allow worsening flips only between consistent outcomes.

In our formulation the constraints are already built into basic definitions (constraints are additional goals in the body of preference clauses, and in the body of the clause defining outcomes) and no changes are necessary. Additionally, recursive LCP-rules (rules with goals `gi` in the body) offer a powerful new form of *dependent* conditional preference statements (Section 3.1), particularly useful in multi-agent contexts [Rossi *et al.*, 2004; Maran *et al.*, 2013]. They support rules such as "If Alice prefers to drive to Oxford today, Bob will prefer to fly to Manchester tomorrow". The rich complexity theory developed for Datalog [Vardi, 1982; Feder and Vardi, 1999; Gottlob and Papadimitriou, 2003; Dantsin *et al.*, 2001] applies *inter alia* to conditional preference theories – in particular we discuss the notion of *data-complexity* in Section 2.3. General results about (linear) Datalog programs lead to complexity bounds for consistency, dominance and optimization extending current known bounds, and in some cases, providing new, simpler proofs for existing complexity bounds (Section 4). Further, tabled Prolog systems such as XSB Prolog[Swift and Warren, 2010; 2012a] implement constrained Datalog with sophisticated features such as partial order answer subsumption that are directly usable in an implementation of LCP. One of the reasons that CP-nets are popular in practice is that useful special cases have been identified (acyclic nets, tree-structured nets) which can be implemented efficiently. In Section 4 we show how some of these special cases can be extended to the richer language we consider. Further, we provide a compiler for LCP theories that can recognize these special cases and generate custom code for consistency, dominance and optimization (Section 5). We present some scalability numbers. In summary, we believe our formalization of extensions of CP-nets permits an integrated treatment of preferences in constraint (logic) programming, leading to more powerful reasoning systems which can deal with both preferences and hard constraints. Please refer to the full version of the paper for details [Cornelio *et al.*, 2015].

2 Background

Below we assume given a set of N *features* (or variables), $\texttt{Var} = \{X_1, \ldots, X_N\}$. We assume for simplicity that the values in each feature X are either $x, \bar{x}$ (handling multiple values is easy).

2.1 GCP-nets and CP-nets

A **Generalized CP-net** (**GCP-net**) [Goldsmith *et al.*, 2008; Domshlak *et al.*, 2003] C over `Var` is a set of conditional preference rules. A **conditional preference rule** is an expression $p : l > \bar{l}$, where l is a literal of some atom $X \in \texttt{Var}$ and p is a propositional formula over `Var` that does not involve variable X.

A GCP-net corresponds to a directed graph (dependency graph) where each node is associated with a feature and the edges are pairs (Y, X) where Y appears in p in some rule $p : x > \bar{x}$ or $p : \bar{x} > x$. Each node X is associated with a *CP-table* which expresses the user preference over X and in which each row corresponds to a rule. The CP-tables can be *incomplete* (for some values of some variables' parents, the preferred value of X may not be specified) and/or *locally inconsistent* (for some values of some variables' parents, the table may both contain the information $x > \bar{x}$ and $\bar{x} > x$). CP-nets [Boutilier *et al.*, 2004b] are a special case of GCP-net in which the preferences are locally consistent and locally complete. An *outcome* in a CP-net is a complete assignment to all features.

Given any GCP-net and CP-net the problems of consistency checking and outcome optimizazion are PSPACE-complete [Goldsmith *et al.*, 2008]. Moreover, there could be several different maximal elements. When the dependency graph has no cycle the CP-net is called *acyclic*. The optimal outcomes for such nets are unique and can be found in polynomial time in N. The procedure used to this purpose is usually called a *sweep forward* and takes N steps [Boutilier *et al.*, 2004b]. The problem of dominance, determining if one outcome is preferred to another, is PSPACE-complete for both GCP-nets and CP-nets. It is polynomial if the CP-nets are tree or poly-tree structured [Domshlak and Brafman, 2002; Goldsmith *et al.*, 2008].

2.2 CP-theories and comparative preference languages

CP-theories are introduced in [Wilson, 2004] as a logic of conditional preference which generalizes CP-nets. (In what follow we use the notation of [Wilson, 2004].) Given a set of variables $\texttt{Var} = \{X_1, \cdots, X_N\}$ with domains D_i, $i = 1, \ldots, n$, the language $L_{\texttt{Var}}$ is defined by all the statements of the form: $u : x_i \succ x_i'[W]$ where u is an assignment of a set of variables $U \subseteq \texttt{Var} \setminus \{X_i\}$, $x_i \neq x_i' \in D_i$ and W is a set of variables such that $W \subseteq (\texttt{Var} \setminus U \setminus \{X_i\})$. Given a language $L_{\texttt{Var}}$ as defined above, a *conditional preference theory (CP-theory)* Γ on `Var` is a subset of $L_{\texttt{Var}}$. Two graphs are associated to a CP-theory: $H(\Gamma) = \{(X_j, X_i) | \exists \varphi \in \Gamma \text{ s.t. } \varphi = u : x_i \succ x_i'[W] \text{ and } X_j \in U\}$ and $G(\Gamma) = H \cup \{(X_i, X_j) | \exists \varphi \in \Gamma \text{ s.t. } \varphi = u : x_i \succ x_i'[W] \text{ and } X_j \in W\}$.

The semantics of CP-theories depends on the notion of a *worsening swap*, which is a change in the assignment of a set of variables to an assignment which is less preferred by a rule $\varphi \in \Gamma$. We say that one outcome o is better than another outcome o' ($\Gamma \vdash o \succ o'$) if and only if there is a chain of worsening swaps (a *worsening swapping sequence*) from o to o'. A CP-theory Γ is *locally consistent* if and only if for all $X_i \in \texttt{Var}$ and $u \in Pa(X_i)$ in the graph $H(\Gamma)$, $\succ_u^{X_i}$ is irreflexive. Local consistency can be determined in time proportional to $|\Gamma|^2 N$. Given a CP-theory Γ, if the graph $G(\Gamma)$ is acyclic, Γ is consistent if and only if Γ is locally consistent; in this case global consistency has the same complexity as local consistency. A CP-net is a particular case of a CP-theory where $W = \emptyset$ for all $\varphi \in \Gamma$.

Comparative preference theories [Wilson, 2009] are an extension of CP-theories. The comparative preference language $\mathcal{CL}_{\texttt{Var}}$ is defined by all statements of the form: $p > q||T$ where P, Q and T are subsets of `Var` and p and q are assignments respectively of the variables in P and in Q. Informally, such a rule says that p is preferred to q if T is held constant. Given a language $\mathcal{CL}_{\texttt{Var}}$ as defined above, a *comparative preference theory* Λ on `Var` is a subset of $\mathcal{CL}_{\texttt{Var}}$.

2.3 Datalog and tabled logic programming

A Datalog program consists of a collection of definite clauses in a language with no function symbols, hence a finite Herbrand domain. Datalog programs can be implemented using *tabled Logic Programming (TLP)*. Tabling maintains a memo table of subgoals produced in a query evaluation and their answers. If a subgoal is reached again, the information in the table can be reused, without recomputing the subgoal. This method ensures termination and improves the computational complexity for a large class of problems [Swift and Warren, 2012a] (at the expense of additional space consumption). Answer subsumption extends the functionality of tabling. *Answer variance* adds a new answer to a table only if the new answer is not a variant of any other answer already in the table. *Partial order answer subsumption* adds a new answer to a table only if the new answer is maximal with respect to the answers in the table, given a partial order `po/2`. Traditionally, predicates are divided into *extensional* and *intensional* predicates. The extensional predicates define a database, and intensional predicates define (possibly recursively defined) queries over the database. In our context, `d/2` and `outcome/1` will be considered extensional predicates (in Flat LCP) and other predicates such as `dom/2`, `inconsistent` and user-defined predicates are considered intensional. Given an intensional program P, database D and query q, *data-complexity* [Vardi, 1982] is the complexity of answering $P, D \vdash q$ as a function of the size of D and q (thus the program is considered fixed). *Combined complexity* is the complexity of answering $P, D \vdash q$ as a function of the size of P, D and q (thus nothing is taken to be fixed). The basic results are that for data-complexity, general Datalog programs are PTIME-complete, and linear programs are NLOGSPACE-complete [Gottlob and Papadimitriou, 2003]. For combined complexity, general Datalog programs are EXPTIME-complete, and linear programs are PSPACE-complete.

3 Logical Conditional Preference Theories

We assume given a set of N features, and a logical vocabulary $\mathcal{V}$ with unary predicates `d1`, ..., `dN` (corresponding to the domains of the features), constants for every value in the domains `di`, a single function symbol `o/N`, and a single binary predicate `d/2`. The user specifies preferences between two outcomes `S,T` (expressed as `o/N` terms) by supplying clauses for the atom `d(S,T)`:

```
d(S,T) :- c, g1, ..., gk.
```

where `c` is a constraint (possibly involving equality), and `g1`, ..., `gk` are possibly recursively defined predicates involving `d/2`. The clause is said to be *flat* if `k=0`, else it is *recursive*.

The user specifies hard constraints on features by providing clauses for the `outcome/1` predicate, typically of the form

```
outcome(o(X1,...,Xn)) :- C,
     d1(X1), ..., dn(Xn).
```

where `C` is a constraint and the `di` are domain predicates.

The LCP runtime supplies the following definition for the `dom/2` predicate, expressing (tabled) transitive closure over `d/2`, and for consistency and optimal outcomes:

```
:- table(dom(_,_)).
dom(X,Y):- d(X,Y), outcome(X), outcome(Y).
dom(X,Y):- d(X,Z), dom(Z,Y).
consistent :- \+ dom(X,X).
:- table(optimal(po(dom/2))).
optimal(X):- outcome(X).
```

Note that the clauses above are linear. Below, given an LCP theory (Datalog program) P, by $\mathcal{L}(P)$ we will mean P together with the LCP runtime clauses specified above.

Given these definitions, the problem solver may use `consistent` to determine whether the supplied preference clauses are consistent, `dom(S,T)` to determine whether outcome `S` is preferred to `T`, and `optimal(S)` (where `S` may be a partially instantiated `o/N` structure) to determine an optimal completion of `S`.

Example 1 (Dinner, modified from [Boutilier *et al.*, 2004b])**.** *Two components of a meal are the soup (`fish` or `veg`) and wine (`white` or `red`). I prefer `fish` to `veg` all else being equal. If I am having `fish`, I prefer `white` wine to `red`. I simply do not want to consider `veg` with `red`. This may be formulated as the LCP theory:*

```
soup(fish). soup(veg). wine(white). wine(red).
outcome(o(X,Y)):- soup(X), wine(Y),
     (X\== veg; Y\==red).
d(o(fish, X), o(veg, X)).
d(o(fish, white),o(fish, red)).
```

On this theory, the query `?-consistent.` returns `yes`. The `dom/2` predicates order outcomes as:

```
o(fish,white) > o(fish,red) o(fish,white) >
o(veg,white)
```

Note because of hard constraints the outcomes are not totally ordered. The query `optimal(X)` returns the instantiation of `O` that is highest in the `dom/2` order, in this case it returns the single answer `X=o(fish,white)`.

Example 2 (Holiday Planning, [Wilson, 2004])**.** *There are three features: `time`, with values later `l` and now `n`; `place`, with values Manchester `m` and Oxford `o`, and `mode` with values fly `f` and drive `d`. The rule "I would prefer to fly rather than drive, unless I go later in the year to Manchester." translates to clauses 1-2. The CP-theory rule "I would prefer to go next week, regardless of other choices." corresponds translated to clause 3, and the comparative preference rule "All other things being equal, I would prefer to fly now, rather than to drive later." to clause 4:*

```
time(n). time(l). place(o).
place(m). mode(f). mode(d).
outcome(o(T,P,M)):- time(T), place(P), mode(M).
(1) d(o(T,P,f),o(T,P,d)):- T=n;P=o.
(2) d(o(l,m,d),o(l,m,f)).
(3) d(o(n,_,_),o(l,_,_)).
```

```
(4) d(o(n,X,f),o(l,X,d)).
```

We now show that the conditional preference rules discussed in the literature can be expressed as special cases of LCP rules. Let $p : l > l'$ be a conditional preference rule where l, l' are distinct literals for some variable X_i. Let $[p]$ represent the constraint obtained from p by replacing every condition $i = v$ (for i a feature) with the formula $(X_i = v, Y_i = v)$. This rule can be represented by the LCP rule: `d(o(`$X_1, \ldots, X_n$`), o(`$Y_1, \ldots, Y_n$`)) :-` $X_i = l, Y_i = l', [p]$.

Similarly, let $R \equiv u : l \succ l'[W]$ be a rule in a conditional preference theory (Section 2.2). Recall that this means that u is an assignment to a set of features $U \subseteq$ `Var`, l, l' are values for a feature i, and W is a set of features from `Var` s.t. the sets $U, \{i\}, W$ are mutually disjoint. As before let $[u]$ stand for the constraint obtained from u by conjoining $X_j = Y_j, X_j = w_j$ for every feature $j \in U$ (where u assigns the value w_j to feature j). Let c stand for the conjunction of constraints $X_i = Y_i$ for every feature i not in $U \cup \{i\} \cup W$. Then R can be represented by the LCP rule: `d(o(`$X_1, \ldots, X_n$`), o(`$Y_1, \ldots, Y_n$`)) :-` $[u], X_i = l, Y_i = l', c$.

The following theorems establish that LCP-theories conservatively extend these sub-languages. Proofs are straightforward, we have essentially just used standard logical notions to formalize the sub-languages:

Theorem 1 (Logical characterization for GCP-nets)**.** *Given a GCP-net $\mathcal{R}$, consider the set P of flat LCP rules representing the rules of $\mathcal{R}$ as described above. Given two outcomes s and t, $\mathcal{R} \vdash s \succ t$ iff $\mathcal{L}(P) \vdash$ `dom(s,t)`.*

Theorem 2 (Logical characterization of *CP-theories*)**.** *Given a CP-theory Γ, consider the set P of flat LCP rules representing the rules of Γ as described above. Given two outcomes s and t, $\Gamma \vdash s \succ t$ iff $\mathcal{L}(P) \vdash$ `dom(s,t)`.*

3.1 Recursive LCP-theories

Recursive or dependent rules are particularly useful in multi-agent contexts, where different agents may influence each other by stating their preferences depending on the preferences of some other agent [Maran *et al.*, 2013]. Here we illustrate with an extension to the Holiday planning example:

Example 3. *Let us consider two agents ranking features "appetizer" (rolls or bread), "main dish" (pasta or fish) and dessert (tiramisu or bread-pudding). We can formulate "If Alice doesn't prefer pasta, I would like to take pasta" as:*

```
d(o(AA, AM, AD, MA, pasta, MD),
  o(AA, AM, AD, MA, fish, MD)) :-
      dom(o(_, fish,_,_,_,_),
      o(_, pasta,_,_,_,_)).
```

Note we do not assume acyclicity in variable ordering. Recursive LCP theories have the full power of Datalog (i.e. any Datalog program can be expressed as a recursive LCP theory).

4 Algorithmic properties

The main algorithmic tasks regarding a preference theory are *dominance queries*, *consistency checking*, and *outcome optimization* Below we fix a set of features `Var` with cardinality N and an LCP-theory P over `Var`.

Checking the dominance over a pair of outcomes corresponds to finding a swapping sequence in CP-theories or a flipping sequence in CP-nets. For LCP-theories this is determined by first-order derivability: the dominance query `(s,t)` succeeds iff $\mathcal{L}(P) \vdash$ `dom(s,t)`.

The problem of consistency checking is checking whether there is any outcome `s` such that $\mathcal{L}(P) \vdash$ `dom(s,s)`.

The problem of outcome optimization corresponds to find the most preferred outcome given an assignment to a (possibly empty) subset of features. Formally, an *optimal outcome* is an outcome `s` such that there is no other outcome `t` such that $\mathcal{L}(P) \vdash$ `dom(s,t)`. Given a (possibly non-ground) term `s` (representing a partial outcome) an *optimal completion* of `s` is a ground term `t` instantiating `s` s.t. for no other ground term `t1` instantiating `s` is it the case that $\mathcal{L}(P) \vdash$ `dom(t1,t)`. Note that an acyclic CP-net or CP-theory (consistent and complete) has a unique optimal outcome, but an LCP-theory may have several optimal outcomes.

Our algorithmic approach is based on analyzing whether the input LCP-theory corresponds to a special case (e.g. acyclic or tree-structured CP-net). If so, optimal algorithms for the special case are used. Otherwise general Datalog procedures are used. In the following sections we analyze the algorithms for dominance, consistency and optimality from a general point of view, and then consider the special cases in which the algorithms are faster. We take the viewpoint of *combined complexity*, i.e. assuming N, the clauses of the LPC theory, and the given query are supplied at runtime. The results are summarized in Table 1.

It is important to notice that for Recursive LCP-theories optimality and consistency procedures never have a lower computational complexity than the dominance procedure, because a recursive LCP rule also contains a dominance query.

We note in passing that for Flat LCP theories, data-complexity is also of interest. Recall that data-complexity for a Datalog programs is the complexity of determining, for a fixed program P, and input database D and query q, whether $P, D \vdash q$ (as a function of the size of D and q). What is the distinction between P and D for LCP theories? For Flat LCP theories, P is simply the clauses for `dom/2`, and `consistent/0`. Once N, the number of features is fixed, this program is fixed. Thus data complexity for consistency of LCP theories corresponds to the complexity of determining for *fixed* N, whether $P, D \vdash$ `inconsistent`, as a function of the number of rules in the program. For Flat LCP (=linear Datalog) the data-complexity is NLogSpace-complete (see e.g. [Gottlob and Papadimitriou, 2003]).

4.1 Dominance

Theorem 3. *Given a flat LCP theory P over N features, deciding* `dom(s,t)` *is* PSPACE*-complete in N.*

Proof. Since flat LCP theories can encode the GCP-nets , the dominance problem is at least PSPACE-hard. That the problem is in PSPACE can be established in a form similar to the proof of Theorem 4.4 in [Gottlob and Papadimitriou, 2003]. Due to lack of space we omit the details. □

	General structure	Acyclic	Tree
Dominance			
CP-nets:	PSPACE-comp [Goldsmith *et al.*, 2008]	NP or harder [Boutilier *et al.*, 2004b]	Polynomial [Boutilier *et al.*, 2004b; Bigot *et al.*, 2013]
CP-theories:	PSPACE-comp [Wilson, 2004]	PSPACE-comp [Wilson, 2004]	**Polynomial**
Flat LCP-theories:	**PSPACE-comp**	**PSPACE-comp**	**?**
Recursive LCP-theories:	**EXPTIME-comp**	**PSPACE-comp**	**?**
Consistency/Optimality			
CP-nets:	PSPACE-complete [Goldsmith *et al.*, 2008]	Polynomial [Boutilier *et al.*, 2004b]	Polynomial [Boutilier *et al.*, 2004b]
CP-theories:	PSPACE-complete [Wilson, 2004]	Polynomial [Wilson, 2004]	Polynomial [Wilson, 2004]
Flat LCP-theories:	**PSPACE-complete/?**	**Polynomial***	**Polynomial ***
Recursive LCP-theories:	**EXPTIME-complete**	**PSPACE-complete**	**?**

Table 1: Computational complexity of Dominance, Consistency and Optimality. *Bold results are provided in this paper, *with some constraint on the form of the rule, ? means the corresponding problem is open.*

Theorem 4. *Given a recursive LCP theory P over N features, deciding* `dom(s,t)` *is* EXPTIME*-complete in N.*

Proof. The proof is as above, except that the `d/2` clauses may no longer be linear, hence the combined complexity for full Datalog comes into play. □

We note in passing that the connection with Datalog allows for a simple and direct proof of the PSPACE-hardness of dominance for CP-nets. We show that the PSPACE-HARD problem of determining whether a deterministic Turing Machine can accept the empty string without ever moving out of the first k tape cells can be reduced to checking dominance queries for CP-nets by modifying slightly the proof for Datalog in [Gottlob and Papadimitriou, 2003, Theorem 4.5]. In practice, the solutions of the dominance problem can be found using tabling on the `dom/2` predicate.

Note that in tree structured CP-nets a dominance query can be computed in time linear in N [Boutilier *et al.*, 2004b; Bigot *et al.*, 2013]. We observe that a procedure similar to [Bigot *et al.*, 2013] can be used for CP-theories.

4.2 Consistency and Outcome optimization

Consistency is determined by invoking the query `?-consistent.` This takes advantage of the tabling of the `dom/2` predicate.

The following theorem affirms that consistency remains in PSPACE even when the language for preferences is extended beyond CP-nets to flat LCP rules, and it is a generalization of Theorem 3 in [Goldsmith *et al.*, 2008].

Theorem 5. *Given a flat LCP theory P over N features, deciding consistency is* PSPACE*-complete in N.*

The proof follows directly from the proof of Theorem 3 since consistency is reduced to checking entailment.

Theorem 6. *Given a recursive LCP theory P over N features, deciding consistency is* EXPTIME*-complete in N.*

As above, noting that the combined complexity for full Datalog is EXPTIME.

For optimality, the user invokes the query `?-optimal(s).` Note that `optimal/1` uses partial order answer subsumption (see Section 2.3). In theory this may result in an exponential number of calls to `dom/2` atoms; each check takes exponential time. This leads to:

Theorem 7. *Given a recursive LCP theory P over N features, deciding optimality is in* EXPTIME *over N.*

For now we leave as open the corresponding problem for flat recursive theories (note that this problem is PSPACE-complete for CP-nets).

Optimization and Consistency for acyclic dependency graphs

In acyclic CP-nets the sweep-forward procedure [Boutilier *et al.*, 2004b] finds the unique optimal outcome (or completion) in polynomial time. A similar result holds for [Wilson, 2004]. Under the assumptions of consistency and acyclicity, an optimal outcome (and completion) can also be found for Flat LCP theories in polynomial time, using the following algorithm (*Acyclic-LCP-Opt*) generalizing sweep-forward (see Section 5): (1) Given a set of LCP-rules we compute the acyclic dependency graph G. (2) We compute a linearization of the topological order defined by G over `Var`: $\mathcal{O} = \{X_1, \ldots, X_N\}$. (3) For each variable X_i, chosen following $\mathcal{O}$, we consider a set $W_i \subseteq$ `Var` such that it contains all the variables that change the value jointly with X_i in at least one rule (W_i could contain only X_i). Using the rules in the LCP-theory that involve the variables in W_i, and given the assignments for the variables $X_1, \ldots, X_{i-1}$, we generate an ordering over the partial outcomes defined on the variables in W_i. We assign to X_i the X_i value of a top element (following a certain tie breaking rule) of the ordering that satisfies `outcome/1` for at least one completion (the completion has the given assignment to $X_1, \ldots, X_{i-1}$ and an arbitrary assignment to $\{X_{i+1}, \ldots, X_N\} \setminus W_i$). (4) We repeat the previous step for all the features in `Var` (following $\mathcal{O}$).

In LCP-theories it is possible to have many different optimal outcomes: we can obtain the whole set of optimal outcomes using the *Acyclic-LCP-Opt* procedure. If there is more than one optimal outcome this means that there exists some variable X that in the third step of the algorithm has more than one top element. Running in parallel all these possible assignments we obtain the whole set of optimal outcomes.

The complexity of the procedure is $O(d^w * N)$ where $w = \max_i |W_i|$: the third step of the procedure could involve all the partial outcomes defined on W_i. If the program bounds w, the procedure becomes linear in N. Note that if the LCP-theory corresponds to a CP-net or a CP-theory then

$|W_i| = 1\ \forall i$ and the algorithm coincides with the sweep-forward procedure for CP-nets, and the procedure introduced in [Wilson, 2004] for CP-theories.

We can use this procedure also to compute the optimal completion, considering the input partial outcome as pre-assigned values for a subset of features. Recursive LCP-theories may require m dominance queries, where m is the number of dominance goals in the body of the input preference rules. Since we use tabling, the time cost is amortized over all calls from the problem-solver (in lie of space). With this change, the procedure described above can be used. Because of the dominance queries, the optimality procedure is PSpace-complete. We note that if evidence for some variables is given, the resulting simplified LCP theory may have the structure of one of the special cases discussed above, and hence optimality and dominance queries may be answered using specialized, polynomial procedures. To this end, the implementation needs to maintain a dynamic dependency graph that ignores features for which values have been provided.

5 Experimental evaluation

We have developed a compiler for LCP theories, also called LCP. The compiler and associated tooling will be made available on Github as an open source project under the Eclipse Public Licence. The compiler reads a LCP-theory, builds the dependency graph and checks whether it represents an acyclic CP-net. If so, it performs a linearization of the dependency graph, and produces a pre-digested representation of the theory. Otherwise it emits the clauses unchanged so that the standard default (tabled) algorithms optimality, consistency and dominance can be used. In more detail, the compiler captures (a linearization of) the dependency order in a clause `dependency(`$[a_1, a_2, \ldots, a_N]$`).` where $a_i \in 1 \ldots N$ (features are implemented as Prolog integers), and if a_j depends on a_i, then $i < j$. Suppose a_p depends on $a_{i_1}, \ldots, a_{i_k}$. If the input LCP program specifies that if each of the features $a_{i_1}, \ldots, a_{i_k}$ had values $x_{i_1}, \ldots, x_{i_k}$ respectively, then the known order of values of a_p is given by (best) $w_1, \ldots, w_r$ (worst), then the compiler emits the fact `preference(`$[x_{p-1}, \ldots, x_1]$`,` $[w_1, \ldots, w_r]$`).` with $x_{i_1}, \ldots, x_{i_k}$ as constant, and the remaining x_i as unique variables (occur only once in the clause). Thus we use Prolog unification to select the correct preference clause to use, given the current partial outcome `[v_p, ..., v_1]` specifying values for the first `p` attributes (in reverse dependency order). We have also implemented a CP-net generator. Generating CP-nets i.i.d. is non-trivial [Allen *et al.*, 2014] and therefore we use an approximation method that randomly generates acyclic CP-nets with N features, given a maximum in-degree k for each feature. We consider a fixed ordering $X_1, \ldots, X_N$ of features. We first generate the acyclic dependency graph: for each feature X_i, we randomly choose its in-degree $d \in 0..\min\{k, i-1\}$. Next, we randomly choose d parents from the features $\{X_1, \ldots, X_{i-1}\}$. When the graph is built, we fill in the CP-tables choosing randomly one element of the domain (since the domain is binary). The resulting CP-net is written out as an LCP theory, using XSB Version 3.5.0 syntax [Swift and Warren, 2012b].

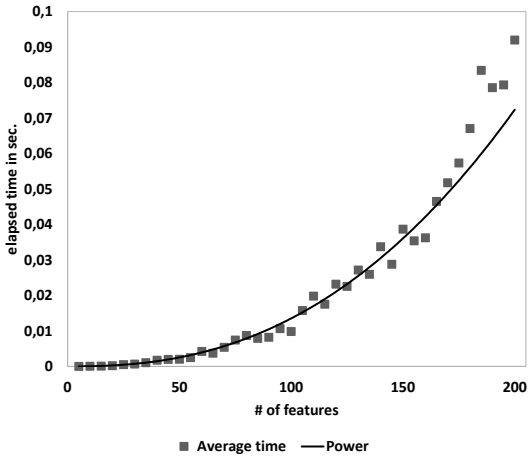

Figure 1: Optimal outcome performances.

We have run two different kinds of experiment. In the first we varied the number of features from 5 to 200 fixing the maximal number of dependencies for each feature and measure the running time for optimality queries, In the second experiment, fixing the number of features, we varied the upper bound of dependencies from 1 to 10. In both experiments we used the CP-net generator that we implemented to generate the CP-nets and we asked for the optimal outcome of the CP-nets 100 times and then we computed the average elapsed time to output the result. Figure 1 shows the results for the experiment where the upper bound for the number of dependencies is fixed to 6. The elapsed time to compute the optimal outcome grows quadratically in the number of features. This is in line with our results as summarized in Table 1. (The runtime is not linear because each step involves checking for the value of parents using unification on $O(N)$ terms.) We obtained the same result varying the upper bound of dependencies, for a fixed number of features.

6 Conclusion and future work

We have presented a new framework for conditional preferences, based on expressing preferences using Datalog. We have shown how dominance, consistency and optimality queries can be formulated directly in Datalog and implemented in modern tabled Prolog systems (XSB Prolog). We have also analyzed the complexity of the different algorithms, developed efficient procedures for some common use cases, and implemented a translator that exploits these algorithms.

Much work lies ahead. Table 1 contains some open complexity questions. We hope to exploit Datalog theory to develop more efficient special cases. We hope to use the implemented LCP system in real-life applications to determine the adequacy of the LCP system.

Acknowledgements. We gratefully acknowledge conversations with David S. Warren and Terrance Swift. These conversations led to a small code-change in the XSB compiler to better support LCP theories. We also acknowledge conversations with Francesca Rossi.

References

[Allen *et al.*, 2014] T.E. Allen, J. Goldsmith, and N. Mattei. Counting, ranking, and randomly generating CP-nets. In *In Proceedings of the 8th Multidisciplinary Workshop on Advances in Preference Handling (MPREF)*, 2014.

[Bigot *et al.*, 2013] D. Bigot, H. Fargier, J. Mengin, and B. Zanuttini. Probabilistic conditional preference networks. In *Proc. of the 29th International Conference on Uncertainty in Artificial Intelligence (UAI)*, 2013.

[Boutilier *et al.*, 2004a] C. Boutilier, I. Brafman, C. Domshlak, H. Hoos, and D Poole. Preference-based Constrained Optimization with CP-nets. *Computational Intelligence*, 20(2):137–157, 2004.

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

[Ceri *et al.*, 1989] S. Ceri, G. Gottlob, and L. Tanca. What you always wanted to know about Datalog (and never dared to ask). *IEEE Trans. on Knowl. and Data Eng.*, 1(1):146–166, March 1989.

[Cornelio *et al.*, 2015] Cristina Cornelio, Andrea Loreggia, and Vijay Saraswat. Logical conditional preference theories. *arXiv preprint arXiv:1504.06374*, 2015.

[Dantsin *et al.*, 2001] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425, September 2001.

[Domshlak and Brafman, 2002] C. Domshlak and R.I. Brafman. CP-nets: Reasoning and consistency testing. In *Proc. 8th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2002.

[Domshlak *et al.*, 2003] C. Domshlak, F. Rossi, K.B. Venable, and T. Walsh. Reasoning about soft constraints and conditional preferences: complexity results and approximation techniques. In *Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

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

[Feder and Vardi, 1999] T. Feder and M. Vardi. The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study Through Datalog and Group Theory. *SIAM J. Comput.*, 28(1):57–104, February 1999.

[Goldsmith *et al.*, 2008] J. Goldsmith, J. Lang, M. Truszczynski, and N. Wilson. The Computational Complexity of Dominance and Consistency in CP-nets. *Journal of Artificial Intelligence Research*, 33(1):403–432, 2008.

[Gottlob and Papadimitriou, 2003] G. Gottlob and C. Papadimitriou. On the complexity of single-rule datalog queries. *Inf. Comput.*, 183(1):104–122, May 2003.

[Kanellakis *et al.*, 1995] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51(1):26 – 52, 1995.

[Maran *et al.*, 2013] A. Maran, N. Maudet, M. S. Pini, F. Rossi, and K. B. Venable. A Framework for Aggregating Influenced CP-nets and Its Resistance to Bribery. In *Proceedings of AAAI-27*, pages 668–674, 2013.

[Prestwich *et al.*, 2004] S. Prestwich, F. Rossi, K. B. Venable, and T. Walsh. Constrained CP-nets. In *in Proceedings of CSCLP04*, 2004.

[Rossi *et al.*, 2004] F. Rossi, K.B. Venable, and T. Walsh. mCP nets: representing and reasoning with preferences of multiple agents. In *Proc. of the 19th AAAI Conference on Artificial Intelligence (AAAI)*, 2004.

[Swift and Warren, 2010] T. Swift and D. S. Warren. Tabling with answer subsumption: Implementation, applications and performance. In *Proceedings of the 12th European Conference on Logics in Artificial Intelligence*, JELIA'10, pages 300–312, Berlin, Heidelberg, 2010. Springer-Verlag.

[Swift and Warren, 2012a] T. Swift and D. S. Warren. XSB: Extending Prolog with Tabled Logic Programming. *Theory Pract. Log. Program.*, 12(1-2):157–187, January 2012.

[Swift and Warren, 2012b] T. Swift and D. S. Warren. XSB home page. `http://http://xsb.sourceforge.net/`, 2012.

[Toman, 1998] D. Toman. Memoing Evaluation for Constraint Extensions of Datalog. *Constraints*, 2(3/4):337–359, January 1998.

[Vardi, 1982] M. Vardi. The complexity of relational query languages (extended abstract. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing (STOC 82*, pages 137–146, 1982.

[Wilson, 2004] N. Wilson. Extending CP-Nets with Stronger Conditional Preference Statements. In *Proceedings of AAAI-04*, pages 735–741, 2004.

[Wilson, 2009] N. Wilson. Efficient Inference for Expressive Comparative Preference Languages. In *Proceedings of IJCAI-09*, 2009.