A New Look at the Semantics and Optimization Methods of CP-Networks

Ronen I. Brafman
Dept. of Computer Science
Ben-Gurion University
Beer-Sheva 84105, Israel
brafman@cs.bgu.ac.il

Yannis Dimopoulos
Dept. of Computer Science
University of Cyprus
Nicosia 1678, Cyprus
yannis@cs.ucy.ac.cy

Abstract

Preference elicitation is a serious bottleneck in many decision support applications and agent specification tasks. CP-nets were designed to make the preference elicitation process simpler and more intuitive for lay users by graphically structuring a set of *Ceteris Paribus* (CP) preference statements - preference statements most people find natural and intuitive. In various contexts, CP-nets with an underlying cyclic structure emerge naturally. Often, they are inconsistent according to the current semantics, and the user is required to revise them. In this paper we show how optimization queries can be meaningfully answered in many "inconsistent" networks without troubling the user with requests for revisions. We also describe a method for focusing users' revision process when revisions are *truly* needed. In the process, we provide a formal semantics that justifies our approach and we introduce new techniques for computing optimal outcomes.

1 Introduction

Ceteris Paribus (CP) preference statements are among the most natural and most intuitive preference statements that people make. Thus, it is not surprising that they have drawn the attention of many researchers in philosophy and AI (e.g., [Doyle and Wellman, 1994; Hanson, 1996]). CP statements indicate a preference for one value over another in the context of a fixed background. For example, the statement "I prefer an apple pie to a chocolate cake as a dessert, *ceteris paribus"* expresses the fact that given two identical contexts - i.e., meals - that differ only in their dessert, the one containing an apple pie is preferred to the one containing a chocolate cake. Finer distinctions can be made using conditional CP statements. For example: "I prefer red wine over white wine if the main course is beef." In this case, the preference for red wine to white wine is restricted to comparisons between identical meals in which the main course is beef.

CP-nets iBoutilier *et al,* 1999] are a graphical tool for representing and for structuring a set of CP statements. A CP-network consists of a graph describing the preferential dependency relationship between different domain variables. Each node is annotated by a conditional preference table (CPT) that describes how the user's preference over the different values of the variable associated with this node depends on the variables associated with the parents of this node.

Cyclic CP-nets emerge naturally when there is a set of interdependent variables, none of which is more important than the other. For example, [Domshlak *et al,* 2001] note that such dependency can emerge naturally among web-page components in their web-personalization tool. Cyclic CP-network raise some conceptual and computational problems to which we still do not have satisfactory answers. Even worse, according to the standard semantics of CP-nets, most cyclic CP-nets are inconsistent. For example, in [Domshlak and Brafman, 2002], it was shown that the preference ordering induced by any simple cycle (i.e, a cycle that does not contain smaller cycles) with more than two nodes is inconsistent. That is, there will be at least two outcomes, *o*\ and *o2,* such that one can show that *o1* is strictly preferred to o_2 and o_2 is strictly preferred to *o*\. In addition, whereas for acyclic networks one can determine the most preferred outcome in linear time even in the face of evidential constraints (i.e., constraints that fix the value of certain variables), no corresponding algorithm is known for networks that contain cycles.

The fact that many cyclic networks arc inconsistent raises a serious practical concern. When a user specifies an inconsistent cyclic network, we must ask him to revise his network. To do so, we need to provide information that will help him obtain a consistent network. In this paper we make two practical contributions aimed at improving this process. First, we show that various optimization queries can be answered naturally even when the network is "inconsistent" according to the standard semantics. In such cases, no additional burden is placed on the user. Second, when revision is required, we show how the notion of a partial model, studied in the area of logic programming, can be used to identify those aspects of a model that require revision.

While pursuing our more practical goals we make a number of technical and semantic contributions. First, we provide a more flexible semantics for CP-nets that is identical with the current semantics on those networks the latter considers consistent. This semantics justifies our approach for generating optimal outcomes given networks that are inconsistent under the standard semantics. Second, we show how to answer optimization queries using various approaches that include reduction to SAT, and reduction to logic programs. This extends current methods which are restricted to acyclic nets, and helps us define the notion of a partial outcome.

The paper is structured as follows: In Section 2 we provide the necessary background on ceteris paribus preference statements and CP-nets. In Sections 3 and 4 we take a closer look at the notion of consistency in CP-nets and the problem that cyclic networks introduce, we suggest a natural definition of a preferred model as one that cannot be improved, and explain how the problem of the existence of such models can be reduced into a CSP problem. In Section 5 we discuss an alternative translation into nonmonotonic logic programs and some of its implications, and in Section 6 we conclude. The full version of this paper is available from the authors' web site. It contains proofs, an optimization method based on cutset conditioning, a discussion of constrained optimization and other inference tasks, and a comparison with logic programs with ordered disjunction fBrewka *et ai,* 2002].1

2 CP-Nets

We start with a review of *ceteris paribus* preference statements and preferential independence, followed by the definition of CP-nets.

2.1 Ceteris Paribus Preference Statements

A *preference relation* is a total pre-order (a *ranking)* over a set of outcomes. Given two outcomes o, o', we write $o \succeq o'$ if o is at least as preferred as o' and we write $o \succ o'$ if o is strictly more preferred than o'.

The types of outcomes we are concerned with consist of possible assignments to some set of variables. More formally, we assume some given set $\mathbf{V} = \{X_1, \ldots, X_n\}$ of variables with corresponding domains $\mathcal{D}(X_1), \ldots, \mathcal{D}(X_n)$. The set of possible outcomes is then $\mathcal{D}(X_1) \times \cdots \times \mathcal{D}(X_n)$. For example, in the context of the problem of configuring a personal computer (PC), the variables may be *processor type, screen size, operating system* etc., where *screen size* has the domain $\{17in, 19in, 21in\}$, *operating system* has the domain $\{LINUX, Windows98, WindowsXP\}$, etc. Each assignment to the set of variables specifies an outcome - a particular PC configuration. Thus, a preference ordering over these outcomes specifies a ranking over possible PC configurations.

The number of possible outcomes is exponential in n, while the set of possible total orders on them is doubly exponential in n. Therefore, explicit specification and representation of a ranking are not realistic. We must find implicit means of describing this preference relation. The notion of conditional preferential independence plays a key role in such representations. Intuitively, X and Y are *conditionally preferentially independent* given Z (where X, Y, Z are disjoint subsets of V) if for every fixed assignment to Z, the ranking of X values is independent of the value of Y. Formally, let X, Y and Z be a partition of V and let z ∈ P(Z). X and Y are *conditionally preferentially independent* given z iff, for all $\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}_1, \mathbf{y}_2$ we have that

$$\mathbf{x}_1\mathbf{y}_1\mathbf{z} \succeq \mathbf{x}_2\mathbf{y}_1\mathbf{z} \text{ iff } \mathbf{x}_1\mathbf{y}_2\mathbf{z} \succeq \mathbf{x}_2\mathbf{y}_2\mathbf{z}$$

X and Y are conditionally preferentially independent given Z if they are conditionally preferentially independent given any assignment $\mathbf{z} \in \mathcal{D}(\mathbf{Z})$. In our PC example, the user may assess *operating system* to be independent of other features given *processor type.* That is, it always prefers LINUX given an AMD processor and Windows98 given an Intel processor (e.g., because he believes Windows98 is optimized for the Intel processor, whereas LINUX is otherwise better). Conditional preferential independence is a standard notion in multiattribute utility theory [Keeney and Raiffa, 1976].

2.2 CP-nets

CP-nets were introduced in [Boutilier *et al.,* 1999J as a tool for compactly and intuitively representing qualitative preference relations. This graphical model exploits conditional preferential independence in structuring a decision maker's preferences under *ceteris paribus* (all else being equal) semantics. CP-net is the first model based on the notions of purely qualitative preferential independence, and bears a surface similarity to Bayesian nets [Pearl, 1988]. However, the nature of the relationship between nodes within a CP-net is generally quite weak compared with the probabilistic relations in Bayes nets.

During preference elicitation, for each variable A' in the variable set V, the decision maker is asked to specify a set of *parent* variables Pa(A') that can affect her preferences over the values of A'. That is, given a particular value assignment to Pa(A"), the decision maker should be able to determine a preference order over the domain of X (denoted as P(A")), all other things being equal.

The above information is used to create the graph of the CP-net in which each node X has $Pa(X)$ as its immediate predecessors. Given this structural information, the decision maker is asked to explicitly specify her preferences over the values of X for each instantiation of Pa(A"). This conditional preference over the values of X is captured by a *conditional preference table* (CPT) which is annotated with the node X in the CP-net. That is, for each assignment to $Pa(X)$, $CPT(X)$ specifies a total order (denoted $\succ$) over $\mathcal{D}(X)$, such that for any two values $x_i, x_j \in \mathcal{D}(X)$, either $x_i \succ x_j$, or $x_j \succ x_i$.

Formally, a *CP-net* $\mathcal{N}$ over variables $\mathbf{V} = \{X_1, \ldots, X_n\}$ is a directed graph G over $X_1, \ldots, X_n$ and a set of conditional preference tables $CPT(X_i)$ for each $X_i \in \mathbf{V}$. Each conditional preference table $CPT(X_i)$ associates a total order $\succ_j$ ($\mathbf{u}_j$) over the domain of X_i with each instantiation $\mathbf{u}_j$ of X_i's parents $Pa(X_i) = \mathbf{U}$.

The semantics of CP-nets is defined as follows. A *model* for a CP-net is a total-order over the set of possible outcomes that satisfies every CPT within the CP-net. A total order satisfies CPT(X) if for every pair of X-values x_i, x_j and every assignment c to $Pa(X)$ such that $x_i \succ x_j$ given c according to CPT(X), the following holds: Let o, o' be two outcomes that differ only in their assignment to X and satisfy c, and suppose that o assigns x_i to X and o' assigns x_j to X, then, $o \succ o'$ holds in the model. We say that $o \succ o'$ is *valid* according to the CP-network if $o \succ o'$ holds in all models of the CP-net. A CP-net is said to be *consistent* if it has a model.

As an example, consider the network in Figure 1(a), in which all variables are boolean. This network has three variables, *A,B,* and *C.* The preferences over *A* and *B* are unconditional, whereas the preferences for C's values depend on the values of *A* and *B.* In Figure 1(b), we see the preference order induced by this CP-network. For every pair of

[1]A note on notation. Due to the difficulty of using the logical negation operator, ¬, in drawing packages, we use two alternative notations: $\bar{a}$ instead of $\neg a$, and $a1, a2$ instead of $a, \neg a$.

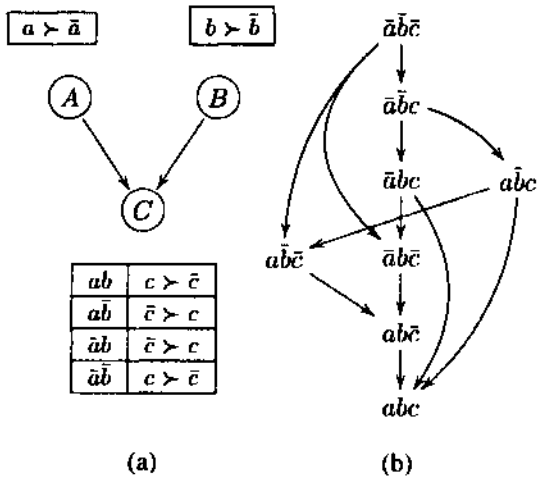

Figure 1: A CP-net and its corresponding preferential order.

outcomes o, o', we have that $o \succ o'$ is valid iff there is a path from o' to o in this graph.

3 The Weak Preference Semantics

According to the standard semantics, the preferences expressed in the CPT of a CP-net are *strict* preferences. We suggest viewing them as *weak* preferences using pre-orders instead of strict orders. As we show, this leads to a semantics that is only slightly weaker than the standard semantics. In fact, it leads to identical orderings on CP-nets that are consistent according to the standard semantics. More formally, a *model* for a CP-network is a total *pre*-order over the set of outcomes that satisfies every CPT within the CP-net. A total pre-order satisfies CPT(X) if for every pair of X-values x_i, x_j and every assignment c to $Pa(X)$ such that $x_i \succeq x_j$ given c according to CPT(X)[2] the following holds: Let o, o' be two outcomes that satisfy c and differ only in their assignment to X, such that o assigns x_i to X and o' assigns x_j to X. Then, $o \succeq o'$ holds in the ordering. We say that $o \succeq o'$ is *valid* according to the CP-network if $o \succeq o'$ holds in all models of the CP-net. Finally, we say that $o \succ o'$ is *valid* according to the CP-network if $o \succeq o'$ is valid but $o' \succeq o$ is not valid. Thus, if $o \succeq o'$ holds in all models and there is a model in which $o \succ o'$ holds, we say that $o \succ o'$ is valid. Had we required $o \succ o'$ to hold in all models for $o \succ o'$ to be valid, no strict preference would be valid. This is because the preference relation in which all outcomes are equally desirable is a model for every CP-network according to the above semantics. Thus, *all* CP-nets are "consistent" according to our semantics.

The above semantics can be related to a more syntactic notion of a proof of preference, or a flipping sequence [Boutilier *et al.,* 1999]. Let *o* be an outcome, and suppose that *o* assigns to *X* some value x_j that the user ranks lower than x_i in parental context *c* satisfied in o. Then, we can improve *o* by flipping the value of *X* from x_j to x_i. An improving *flipping sequence* $o_1, o_2, \ldots, o_k$ is a sequence of outcomes such that

[2] In the standard semantics we have $x_i \succ x_j$ instead of $x_i \succeq x_j$.

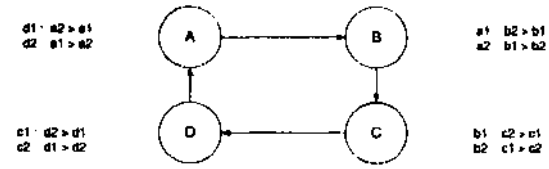

Figure 2: A Simple 4-Cycle CP-network.

o_i is obtained from o_{i-1} via a single improving flip. Naturally, these definitions are with respect to some CP-network, which we take to be fixed by the appropriate context.

Theorem 1 *$o \succeq o'$ is valid iff there is an improving flipping sequence from o' to o.*

An immediate consequence is the following:

Consequence 1 *$o \succ o'$ is valid iff there is an improving flipping sequence from o' to o but no improving flipping sequence from o to o'.*

Another interesting theorem relates the standard, stronger semantics, and our semantics.

Theorem *2 Let G be a CP-network that is consistent according to the standard semantics. Then, G satisfies $o \succ o'$ according to our semantics iff G satisfies $o \succ o'$ according to the standard semantics.*

This theorem shows that our semantics is a direct extension of the standard semantics. To understand it better, we will soon take a look at networks that are inconsistent according to the standard semantics.

Finally, we define the notion of optimality in the context of a CP-network. There are two possible definitions: An outcome o is said to be *strongly optimal* iff there is no other outcome o' such that $o' \succeq o$ holds. An outcome *o* is said to be *weakly optimal* iff there is no other outcome *o'* such that $o' \succ o$ holds. Thus, in the first case, *o* is either strictly better than any other outcome or incomparable. In the second case, there may be other outcomes that are as preferred as o, but no outcome that is strictly preferred over *o.* In networks that are consistent according to the standard semantics, there is a unique best outcome that is strictly better than any other outcome. However, when we move beyond this class of networks, we can have more than one optimal outcome, and it can be either strongly or weakly optimal. The latter class is computationally more challenging to identify, and so we concentrate on strongly optimal outcomes.

4 Optimality in Cyclic Networks

The semantics of CP-nets allows for cycles. But only the consistency of acyclic CP-nets is guaranteed according to the standard semantics [Boutilier *et al,* 1999]. Moreover, such networks have a unique optimal outcome. This remains true even if we introduce evidential constraints, i.e., constraints that *fix* the value of some variables.

Cyclic networks often induce an inconsistent (i.e., cyclic) preference order according to the standard semantics. For example, consider the network in Figure 2 which contains a cycle of size 4, called a simple cycle in [Domshlak and Brafman, 2002]. The web-personalization applications described in [Domshlak *et al.,* 2001] naturally gives rise to such cyclic structures. There, variables correspond to articles, ads, and other content element in an online newspaper. The value of each variable indicates whether it is currently displayed or not. The CP-net captures the preferences of the editor regarding the presentation of different elements on the user's current screen. These preferences together with user-generated content constraints lead to a personalized view that takes into account the user's interests and the editor's expertise on preferred combinations of news items. For example, in the CP-net in Figure 2, *A* could be a review of a new Toyota 4x4 vehicle, *B* a test-drive of a new BMW series 7 car, *C* a story about a recent Man. United match, and *D* a story about Man. City's recent success. Suppose that Man. United is sponsored by Toyota, and Man. City by BMW, and the editor would prefer not to display stories about competing teams and/or companies in the same screen. This is expressed in the CP-net in Figure 2 by stipulating that if *A* is present then *B* should not, if *A* is not present then *B* should be present, etc.

In [Domshlak and Brafman, 2002] it is shown that such a network is not consistent, i.e., there is no total order of the set of outcomes that satisfies all the preferences embodied in this network. It is easy to see why this is so when we examine Figure 3 which describes the relationships among different possible outcomes. The nodes in that figure correspond to outcomes and the edges correspond to legal improving (single) flips. For example, consider the outcome *a2b2c\d2* in the lower left-hand side of Figure 3. Given that *A* is assigned a2, we can see from the B's CPT that 61 is a more preferred value for *B.* Thus, there is an edge from *a2b2cld2* to *a2blcld2.*

We can see that Figure 3 contains cycles, making it impossible to totally order its elements consistently. However, it is also apparent that it contains two elements, *alb2cld2* and *a2blc2dl* that, in some sense, can be viewed as optimal elements, as well as two elements, *alblcldl* and *a2b2c2d2,* that in some sense can be viewed as the worst elements.

Indeed, according to our semantics, this network induces three classes of outcomes described in Figure 4. The outcomes within each class are either equally preferred or incomparable, and there is a strict preference between outcomes belonging to different classes. An important consequence of this is that there are clear candidates for the set of optimal outcomes: the two outcomes in the top class.

We can see that our semantics is more lenient, allowing for specifications that, in some sense, are cyclic. If the cycle contains all outcomes, then all outcomes are equally preferred. This specification is not informative, and the user needs to be informed of this fact. However, the example above shows that the cycle may contain some, but not all of the outcomes. Whereas the standard semantics will dismiss this CP-network as inconsistent, our semantics is more tolerant, and can use this CP-net to determine an optimal outcome.

We now turn to the computation of an optimal outcome via a simple reduction to a CSP problem (or a SAT problem when the variables are binary). The variables in our reduction

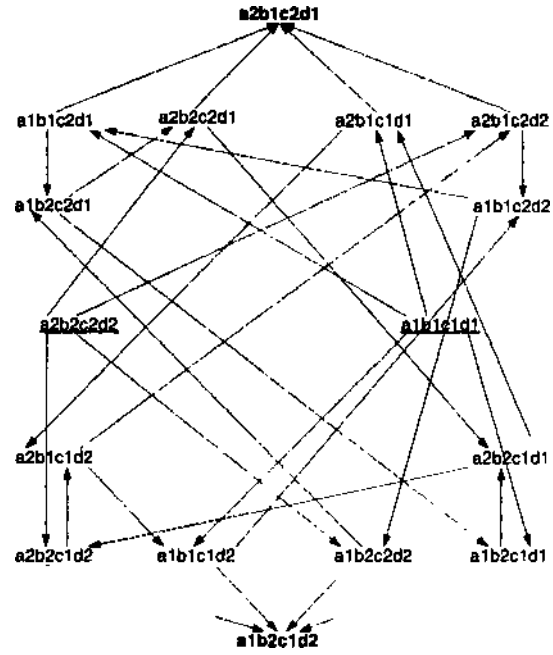

Figure 3: Outcome Space for the 4-Cycle Network.

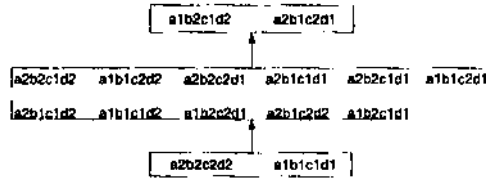

Figure 4: Ordered Outcome Classes for 4-Cycle Network

consist of the variables in the CP-network. hor every entry in the CP Table of every variable v, we add the constraint $\phi \rightarrow v_1$, where ϕ denotes the context (i.e., the assignment to the parents of v) and v_1 is the preferred value of v.

As an example, for the CP-network in Figure 1 we obtain the following propositional formula.

$$a \wedge b \wedge (a \wedge b \rightarrow c) \wedge (a \wedge \bar{b} \rightarrow \bar{c}) \wedge (\bar{a} \wedge a \rightarrow \bar{c}) \wedge (\bar{a} \wedge \bar{b} \rightarrow c).$$

In the case of Figure 2, we get the formula

$$(d1 \rightarrow a2) \wedge (d2 \rightarrow a1) \wedge (a1 \rightarrow b2) \wedge (a2 \rightarrow b1) \wedge$$
$$(b1 \rightarrow c2) \wedge (b2 \rightarrow c1) \wedge (c1 \rightarrow d2) \wedge (c2 \rightarrow d2).$$

Lemma 1 *o is a strongly optimal outcome for a CP-net iff it satisfies the above CSP*

The above algorithm does not work when we have weakly optimal outcomes, i.e., when we have a set of outcomes that are equally preferred, but cannot be strictly improved. Figures 5 and 6 show an example of a network giving rise to such a case and the relation over the outcome space it induces. From now on, our treatment centers on *strongly optimal* outcomes only, to which we simply refer as *optimal outcomes.*

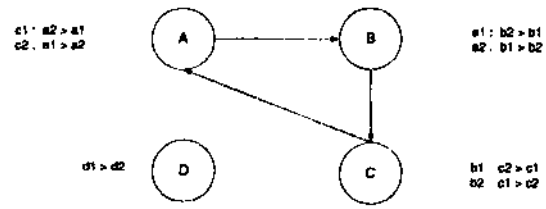

Figure 5: A Network Without a Strongly Optimal Outcome

[a1b1dd1	a1b1c2d1	a1b2c1d1	a1b2c2d1
a2b1c1d1	a2b1c2d1	a2b2c1d1	a2b2c2d1

t

la1b1c1d2	a1b1c2d2	a1b2c1d2	a1b2c2d2
a2b1c1d2	a2b1c2d2	a2b2c1d2	a2b2c2d2 J

Figure 6: Outcome Classes for Network in Figure 5

Lemma 1 implies that the problem of finding a single optimal outcome is in NP. When the CP-network is acyclic, an optimal outcome can be found in linear time [Boutilier *et* a/., 19991. Our result indicates a linear time solution for simple-cycle nets, too, if the variables are binary, because the resulting CSP is an instance of 2-SAT LAspvall *et* a/., 1979].

5 Logic Programs and Partial Consistency

We start with some background on the semantics of logic programs, followed by a reduction of CP-net optimization to logic programs. Then, using the notion of a partial stable model of a logic program, we can define a corresponding notion for a CP-net that captures those variables for which we have some reasonable candidate for an optimal value.

5.1 The Semantics of Logic Programs

A propositional *normal logic program* P is a set of *normal rules*. A normal rule is a rule of the form $h \leftarrow a_1, a_2, ..., a_n, not\ b_1, not\ b_2, ..., not\ b_m$ where $h, a_1, a_2, ..., a_n, b_1, b_2, ..., b_m$ are propositional atoms. The set of atoms of P is denoted by $Atoms(P)$. Let S be a truth or value assignment on the atoms of *Atorns(P).* We denote by S^+ the set of atoms of 5 that are assigned the value true, and *S~* the set of atoms that are assigned the value false. The *reduct* P^S of a logic program *P* w.r.t. an assignment *S* is the logic program obtained from *P* after deleting (i) every rule of *P* that has a negated atom $not\ b_i$, with $b_i \in S^+$ (ii) every negated atom from the body of the remaining rules.

The resulting program does not contain negated atoms, and is called a definite logic program. Let $cl(P)$ denote the deductive closure of a definite logic program P, which coincides with its minimal model. A *stable model* LGelfond and Lifschitz, 1988] of a logic program *P* is an assignment 5, such that $S^+ = cl(P^S)$. A normal logic program may have none, one or several stable models. The problem of deciding whether a normal logic program has a stable model is NP-complete, but many problems can be solved efficiently in practice by state-of-the art systems such as smodels iSurjanen and Niemela, 2001] and DLV iDell'Armi *etal,* 2001].

An alternative semantics for normal logic programs is the *partial stable model* semantics [Sacca and Zaniolo, 1990]. This is a three-valued semantics, where each atom may assume the values: true, false or undefined. Given a three-valued assignment *S of Atoms(P).,* we say that *S is a partial model* of *P* if for each $\neg A \in S$\ every rule with head *A* contains at least one literal *B* in its body, such that $\neg B \in S$. The reduct P^S of *P* w.r.t. a partial model *S* is defined as the program that is obtained from *P* after deleting (i) every rule of *P* that has a negated atom *not* b_i, with $b_i \in S^+$ (ii) every rule where an undefined atom occurs (iii) every negated atom from the body of the remaining rules. A partial model *S* such that $S^+ = cl(P^S)$ is called a *founded model* of P. A maximal founded model is called *partial stable model of P.* It is not hard to see that every program has a partial stable model.

Example 1 *Consider the program*
$a \leftarrow not\ d \quad b \leftarrow not\ a \quad c \leftarrow not\ b \quad d \leftarrow not\ c$
This program has two stable models, namely $M_1 = \{a, c\}$ *and* $M_2 = \{b, d\}$. *Consider now the program*
$a \leftarrow not\ c \quad b \leftarrow not\ a \quad c \leftarrow not\ b$
$d \leftarrow not\ e \quad e \leftarrow not\ d$
This program has two partial stable stable models, namely $M_1 = \{d\}$ *and* $M_2 = \{e\}$, *but no stable model.*

The intuition behind a founded model is similar to that behind a stable model, except that we are willing to ignore certain variables and the rules that they participate in. A partial stable model is one where we try to ignore as few variables (w.r.t. set inclusion) as we can.

5.2 Translating CP-nets into Logic Programs

In this section we translate CP-nets into nonmonotonic logic programs, showing a one-to-one correspondence between the net's optimal outcomes and the stable models of the corresponding logic program.

A preference $p_1 \vee p_2 \vee \ldots \vee p_k : q_1 > q_2 > \ldots > q_n$, where each p_i is a conjunction of atoms, translates into a set of rules of the following form $q_1 \leftarrow p_j$ for $1 \leq j \leq k$. Additionally, for every variable X of the network with $dom(X)=\{v_1, v_2, \ldots, v_n\}$ a set of rules of the form $v_i \leftarrow not\ v_{i_1}, not\ v_{i_2}, \ldots, not\ v_{i_{n-1}}$, for $i_j \neq i$, is added to the program. We call this set *choice rules set.* Finally for each variable X, with $dom(X)=\{v_1, v_2, \ldots, v_n\}$ we add the set of *uniqueness constraints* $\leftarrow v_i, v_j$, for every pair of values such that $i \neq j$.

Theorem 3 *o is a strongly optimal outcome for a CP-net iff it is a stable model of its corresponding logic program.*

Note that only the most preferred values of each preference statement participate in the translation of a CP-network into a logic program. Furthermore, the *not* operator appears only in the choice rules set, a feature that seems to indicate a rather weak link between CP-networks and nonmonotonic logic programs. However, in the long version of the paper we show that in the case of CP-networks with constraints, the *not* operator plays a more central role in the translation and the relation between CP-networks and logic programs is stronger than it may appear here.

Example 2 *Consider the network:*
$a_1 : b_1 > b_2 \quad a_2 : b_2 > b_1 \quad b_1 : a_1 > a_2 \quad b_2 : a_2 > a_1$
The corresponding logic program is
$b_1 \leftarrow a_1 \quad b_2 \leftarrow a_2$
$a_1 \leftarrow b_1 \quad a_2 \leftarrow b_2$
$a_1 \leftarrow not\ a_2 \quad a_2 \leftarrow not\ a_1 \quad b_1 \leftarrow not\ b_2$
$b_2 \leftarrow not\ b_1 \quad \leftarrow a_1, a_2 \quad \leftarrow b_1, b_2.$
The above program has exactly two stable models, namely $M_1 = \{a_1, b_1\}$ *and* $M_2 = \{a_2, b_2\}$.

5.3 Partial Stable Models and Partial Outcomes

We showed a direct correspondence between the optimal outcomes of a CP-network and the stable models of its associated logic program. However, some logic programs have *no* stable model, much like some CP-nets have no strongly optimal outcome. We can interpret the non-existence of an optimal outcome as an indication that some of the preferences are not well-defined. There are two ways to handle such situations: we can isolate the ill-defined preferences and reason with the rest of the network, or we can try to revise the ill-defined preferences. In both case, we require the ability to identify the problematic parts of the network. Here we show how partial stable models can provide valuable assistance in this task.

Example 3 *Consider the network defined as follows.*
$a_1 : b_1 > b_2 \quad a_2 : b_2 > b_1 \quad b_1 : a_2 > a_1 \quad b_2 : a_1 > a_2$
$c_1 : d_1 > d_2 \quad c_2 : d_2 > d_1 \quad d_1 : c_1 > c_2 \quad d_2 : c_2 > c_1$
Its corresponding logic program is
$b_1 \leftarrow a_1 \quad b_2 \leftarrow a_2 \quad a_2 \leftarrow b_1 \quad a_1 \leftarrow b_2$
$c_1 \leftarrow d_1 \quad c_2 \leftarrow d_2 \quad d_1 \leftarrow c_1 \quad d_2 \leftarrow c_2$
together with the corresponding choice rules set and uniqueness constraints. This program has no stable model, but it has two partial stable models, $M_1 = \{c_1, d_1\}$ *and* $M_2 = \{c_2, d_2\}$. *These partial stable models do not assign a value to variables* a_1, a_2, b_1, b_2 *because the corresponding subnetwork does not possess an optimal outcome.*

We call the outcomes of a CP-network that correspond to the partial stable models of its associated logic program, *partial optimal outcomes.* Intuitively, a partial stable model corresponds to an optimal assignment to a coherent sub-net of the original CP-net that contains the same preferences as the CP-net. Thus, we attempt to remove certain nodes, until we get a coherent specification. Those nodes that were removed are the ones we should point to the user as problematic.

Existing knowledge on partial stable models (e.g., [Dimopoulos *et al,* 2002]) implies that a total partial optimal outcome (i.e., one that assigns a value to every variable) is an optimal outcome, and that deciding whether a logic program has a partial stable model other than the empty set is NP-complete. Discovering whether a variable appears in *some* partial optimal outcome is NP-complete, whereas deciding whether a variable appears in *all* partial optimal outcomes is Π^p_2-complete.

6 Conclusion

We showed that the current semantics of CP-network is too conservative in its interpretation of user specification. This is contrary to the aim of research on preference elicitation techniques, i.e., reducing the burden on users. Consequently, we suggested a different semantics under which optimization can be meaningfully applied to CP-nets that are currently considered inconsistent. We showed how such an optimization process can be practically implemented, extending current optimization techniques to cyclic network. Moreover, we described a computational method for identifying sub-networks that are meaningful, providing the first tool that can point out to a user what part of her specification requires revision.

References

[Aspvall *et al.,* 1979] B. Aspvall, M. Plass, and R. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Proc. Letters,* 1979.

[Boutiliere/fl/., 1999] C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole. Reasoning with conditional ceteris paribus preference statements. In *VAV99,* 1999.

[Brewka *etai,* 2002] G. Brewka, 1. Niemela, and T. Syrjanen. Implementing ordered disjunction using answer set solvers for normal programs. In *J EUA'02,2002.*

[DeirArmi*/fl/.,2001] T. DeH'Armi, W. Faber, G. Jelpa, C. Koch, N. Leone, S. Perri, and G. Pfeifer. System description: DLV. In *Proceedings of LPNMR-01,* 2001.

[Dimopoulos*etal,* 2002] Y. Dimopoulos, B. Nebel, and F. Toni. On the computational complexity of assumption-based argumentation for default reasoning. *Artificial Intelligence,* 141,2002.

[Domshlak and Brafman, 2002] C. Domshlak and R. 1. Brafman. CP-Ncts - reasoning and consistency testing. In *Proc. KR'02, 2002.*

iDomshlakera/., 20011 C. Domshlak, R. I. Brafman, and E. S. Shimony. Preference-based configuration of web page content. In *Proc. I J CAPO I,* 2001.

[Doyle and Wellman, 1994] J. Doyle and M. Wellman. Representing preferences as *ceteris paribus* comparatives. In *AAAI Spring Sym. on Decision-Theoretic Planning,* 1994.

iGelfond and Lifschitz, 1988] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. ICSLP-88,* 1988.

[Hanson, 1996] S. O. Hanson. What is a ceteris paribus preference. *J. of Philosophical Logic,* 25:307-332,1996.

[Keeney and Raiffa, 1976] R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-offs.* Wiley, New York, 1976.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[Sacca and Zaniolo, 1990] D. Sacca and C. Zaniolo. Stable models and non-determinism in logic programs with negation. In *Proc. PODS'90,*1990.

[Surjanen and Niemela, 2001] T. Surjanen and I. Niemela. The Smodels system. In *Proceedings ofLPNMR-01,*2001.