Preferences over Objects, Sets and Sequences

Sandra de Amo and Arnaud Giacometti
Universidade Federal de Uberlândia
Université de Tours
Brazil
France

1. Introduction

Recently, a lot of interest arose in the artificial intelligence and database communities concerning the topic of preference elicitation, modelling and reasoning. In fact, due to the huge amount of information users are faced up to daily, the development of formalisms allowing preference specification and reasoning turns out to be an essential task. A lot of work has been done in this area so far (Boulilier et al., 2004); (Brafman et al., 2006a); (Chomicki, 2003); (Kießling, 2002); (Wilson, 2004). Most of this work focus on specifying and reasoning with preferences over objects in some universe *U*. In most applications, mainly those related to the database field, one deals with huge set of objects, which makes unfeasible for the users to specify their preferences in a *quantitative* way, that is, by explicitly associating to each object (or tuple) *o* a number *pref(o)* standing for her *degree of preference* concerning this object. A *qualitative* framework for expressing preferences over objects is more suitable in this case. The user is asked to provide a set of statements or rules which express her generic preferences over the attribute values of the objects. For instance, the user can express her preferences about films by stating that (1) concerning comedies, she prefers those from Woody Allen to those from Nanni Moretti (2) concerning Nanni Moretti's movies, she prefers comedies to dramas. Such frameworks, besides providing a compact way for expressing preferences, are also supposed to derive an explicit preference ordering over the objects, given the compact specification provided by the user, and produce an algorithm to determine the most preferred objects according to this ordering.
Some recent research on preference elicitation and reasoning has focused on preference over more complex entities, like *sets* of objects (Brafman et al., 2006b); (des Jardin & Wagstaff, 2005). Indeed, in many situations, instead of selecting a most preferred object, one may be interested in selecting a best *set* of objects whose components satisfy certain criteria of diversity and mutual compatibility. For instance, in the creation of a film festival program, a criteria for a "good" program could be *"a program including a comedy is better than a program which doesn't include one"*.
However, more complex entities other than simple sets of objects have been appearing in recent applications. For instance, in the design of a web page, the developer can take into account user preferences about hyperlink structures (trees). In our example of the film festival program, an optimal program should not only be characterized by the quality, diversity and compatibility of its components but also by the *ordering* in which each film is

presented in the program. So, it is natural to think about preference elicitation and reasoning over *structures* rather than merely over simple objects or non-structured sets of objects.

In this chapter, we cover with some details some classical and important formalisms to specify preferences over objects and sets of objects and we address the problem of specifying and reasoning with preferences over *sequences* of objects. The material we present here addressing this topic is an extension of our previous work (de Amo & Giacometti, 2007). Preferences over sequences of objects naturally appear when a decision maker is faced to the problem of producing an optimal sequence of objects. The following example illustrates the kind of preference statements we will deal with.

Example 1 Let us suppose a decision maker who works on the creation of a program for a film festival. Based on his past experiences on film festivals, there are some rules he thinks are crucial to the success of such an event.

1. For comedies, it is better to choose those by Woody Allen than those by Nanni Moretti. Concerning Nanni Moretti's movies, comedies are better than dramas.
2. It is better to start the festival by presenting a comedy.
3. If the previous film was a comedy, then it is better to follow it by a drama. However, if the previous film was a drama, then it is better to follow it by a comedy, unless it is a film by Nanni Moretti, in which case, it is better to follow it by another drama.
4. If there is a drama in the program, then it is better to present a comedy sometime before it.

We introduce the logic framework **TPref** allowing preference elicitation and reasoning over sequences of objects as well as an algorithm to yield the most preferred sequences satisfying a given set of *temporal* constraints. Our elicitation procedure consists in obtaining from the user (1) a set of *temporal conditions* which affects her preferences over sequences of objects and (2) a set of statements or rules involving these temporal conditions, which express her preferences. The four statements illustrated in Example 1 are preference statements we treat in this paper.

After preference elicitation, the statements provided by the user are translated into formulae of the logic **TPref**. Our formalism, which is based on Propositional Temporal Logic (PTL), generalizes the language introduced in (Wilson, 2004) for expressing preference over single objects. We show a procedure to decide the consistency of a set of statements in the *past* fragment of the logic **TPref**, that is, if a set of statements Φ (a compact preference representation) derives an explicit preference ordering $>_{\phi}$ over sequences of objects. We discuss the difficulties for using this same idea in proving consistency in the general case of preference statements involving past and future conditions. Finally, we provide an algorithm for producing the best sequences of objects given a set of temporal preference statements Φ.

1.1 Related work

The research literature on preference reasoning and elicitation over objects is extensive. The approach of CP-Nets (Boutilier et al., 2004) uses a very simple graphical model which captures users qualitative conditional preference over objects, under a *ceteris paribus* semantics. The order on objects induced by a CP-Net is rather restrictive, due mainly to the ceteris paribus semantics and also by the fact that all attributes are equally important where comparing two objects. The approach of TCP-Nets (Brafman et al., 2006a) generalizes the

CP-Nets by introducing the ability of expressing a relative importance and conditional relative importance of object attributes. Thus, a TCP-Net is a more refined tool for comparing objects than CP-Nets. The approach introduced in (Wilson, 2004) uses a logical framework for expressing conditional preference statements. It consists of a formalism in the same lines of CP-Nets but with a richer language allowing to express not only the usual CP-Nets ceteris paribus statements but also TCP-Nets statements and more general conditional statements (called *stronger conditional statements*). The temporal conditional preference statements we introduce in Section 4 for specifying preferences over sequences of objects is a generalization, in the temporal context, of the stronger conditional statements of (Wilson, 2004). The conditions in a stronger conditional statement can be viewed as a propositional logic formula. In our approach for specifying preferences over sequences of objects, the conditions are propositional *temporal* logic formulae.

In the database area, the problem of enhancing well-known query languages with preference features has been tackled in several recent and important works in the area. In (Chomicki, 2003), a simple logical framework is proposed for expressing preferences. Preferences are expressed by *preference formulae*. These formulae are incorporated into relational algebra and into SQL, through the operator *winnow* parameterized by a preference formula. (Kießling, 2002) introduced Preference SQL which extends SQL by a preference model based on strict partial orders. Several built-in base preference constructors are proposed. The optimizer uses an efficient rewriting procedure which transforms preference queries into standard SQL queries.

Recent work on preference modelling in AI has focused on *sets* of objects instead of single objects. In (Brafman et al., 2006b), a language for specifying *qualitative* preferences over sets is introduced. The language allows users to express preferences over sets of objects taking into account a class of basic properties which affect their choice. It is shown that a set-preference statement specified in this language can be transformed into a conditional preference statement over attributed objects. The language introduced in (des Jardins&Wagstaff, 2005) allows *quantitative* preference specification over sets of objects. It supports two important preference notions: *diversity* and *depth*. *Diversity* specifies the amount of variability among objects in a set, and *depth* specifies preferred feature values.

Most work on *temporal reasoning with preferences* is related to automated plannning. Preferences concerns the relative execution times of a set of events $\{e_1, \ldots, e_n\}$ (Khatib, 2001); (Kumar, 2007). A preference statement in such an *explicit* temporal framework may establish for instance that event e_i must be scheduled between x_i and y_i seconds before event e_j.

Propositional Temporal Logic (PTL) was introduced in (Prior, 1997) as a formal system for specifying and reasoning with paralell programs. Recently, PTL has been used in the automated planning context, as a formalism to specify "good" executing plans (which can be viewed as sequences of state transitions). In (Bacchus et al., 1996), PTL has been used in the automated planning context where actions depend on past and current states. The problem considered there is of rewarding temporally extended behaviors, that is, rewarding sequence of actions (or state transitions) achieving a predefined goal. Rewards are associated to properties that sequences must satisfy. Such properties are expressed by PTL formulae. In (Bienvenue et al., 2006), a formalism based on temporal logic and situation calculus was introduced in order to express *qualitative preferences* about executing plans. Such formalism allows to specify, to reason and to generated preferred plans. This approach generalizes the one proposed in (Son & Pontelli, 2006), which also uses temporal logic for expressing preferences over executing plans with an implementation using answer-set programming. At the best of our knowledge, there are no work treating qualitative conditional preferences elicitation and reasoning over *sequence of attributed objects* in the lines of the CP-Net formalism. The approach we propose in this chapter is a first step towards incorporating a formalism for *reasonning* with preferences over sequences of objects into a temporal relational query language, and so, building a bridge between the two disciplines (AI and Temporal Databases), in the lines of which has been done in (Endres & Kießling, 2006), where a method for transforming TCP-Nets queries into database preference queries has been proposed.

Chapter Organization. This chapter is organized as follows. In Section 2, we present three classical approaches for preference specification over *objects*, the CP-Nets, the TCP-Nets and the strong conditional statements of (Wilson, 2004). We discuss important problems related to this topic, such as finding the most preferred objects, comparing objects (dominance queries) and ordering objects (ordering queries). We describe the third approach (Wilson, 2004) with more details since it constitutes a necessary background for our work on sequences of objects. In Section 3, we present a simple approach for specifying preferences over *sets of objects*. In Section 4, we present our approach for eliciting and reasoning with preference over *sequences of objects*. In this Section, we introduce the syntax and semantics of the language **TPref** allowing to express preferences over sequence of objects, we show how to test the consistency of a set of statements Φ in **TPref**, and discuss its complexity. Besides, we present an algorithm to produce the optimal sequences satisfying a set of simple temporal constraints.

2. Preferences over objects

In most AI applications involving the ability of making decisions, users are required to compare different alternatives and must be able to choose those which better conform to their needs or personal preferences. Thus, such applications must support the ability of automate the preference elicitation process. In this section we will present three important approaches for representing and reasoning with preferences over objects. The first approach is based on a graphical model focusing on the notion of conditional preferential independence. The second approach is also based on a graphical model and generalizes the first one. The third approach is quite general and is based on a logical framework allowing users to express their preferences through a set of rules.

2.1 Conditional preference networks (CP-Nets)

The graphical model we describe in this section was introduced in (Boutilier et al., 2004) and is similar to a Baysesian Network (Pearl, 1988) from a syntactical point of view. Nevertheless both models differ with respect to their semantics. The model we present here, called *Conditional Preference Networks (CP-Nets)* uses a graph in order to capture statements of *qualitative* conditional preference independence. The semantics of the model is based on the *ceteris paribus* semantics which has been largely exploited in the AI field in the past (Doyle et al., 1991). Other approaches for representing and reasoning with preferences have employed graphical representations of preference relations such as (Bacchus & Grove, 1995) and (Bacchus and Grove, 1996), but with a different semantics.

In the CP-Net preference model, the user is required to specify, for any specific attribut A of interest, which other attributes can influence her preferences for values of A. For each instantiation of the relevant attributes for A (the parents of A in the graphical representation) the user must specify her preference ordering over values of A according to the values of its parents. For instance, let us consider a set of objects with attributes A,B,C,D and let us suppose that preference over attribute A depends on attributes B and C. So, the user may specify that *if the value of B is b_1 and the value of C is c_2, and everything else is equal then she prefers a_2 to a_1 as a value for attribute A*. Based on this preference rule, the user can decide that between two objects $o_1 = (a_1, b_1, c_2, d_1)$ and $o_2 = (a_2, b_1, c_2, d_1)$ she prefers object o_2 to object o_1. On the other hand, this rule cannot allow her to decide that object o_2 is preferred to object $o_3 = (a_1, b_1, c_2, d_2)$, since the values of the attribute D in both objects are different. The *ceteris paribus semantics* (*everything else being equal*) imposes that we can only compare objects according to a given preference rule r if the objects have the same values on the attributes not appearing in r.

Notation. We suppose a set $V = \{X_1, X_2, ..., X_n\}$ of attributes. For each attribute $X \in V$, we denote by $\mathbf{dom}(X)$ the finite set of values of X (the domain of X). For $Z = \{Z_1, ..., Z_m\} \subseteq V$ we denote by $\mathbf{dom}(Z)$ the set $\mathbf{dom}(Z_1) \times \mathbf{dom}(Z_2) \times ... \times \mathbf{dom}(Z_m)$. If $Z = V$, we denote by $\mathcal{O}$ the set $\mathbf{dom}(Z)$. The elements of $\mathcal{O}$ are called *objects, tuples or outcomes*. If $o = (x_1, ..., x_n)$ is an object, we denote by $o[X_i]$ the element $x_i \in \mathbf{dom}(X_i)$. If $Z = \{Z_1, ..., Z_m\} \subseteq V$, we denote by $o[Z]$ the tuple of elements $(o[Z_1], ..., o[Z_m])$. Sometimes we abbreviate this tuple by $\mathbf{z}$.

Definition 1 (CP-Net Preference Model) A *CP-Net* over a set of attributes V is a directed graph $\mathcal{N} = (V, E)$ where each node $X \in V$ is annotated with a conditional preference table (CP table) CPT(X). Each CP table CPT(X) associates a total order $\succeq_{\mathbf{u}}$ with each instantiation $\mathbf{u}$ of the attributes which are parents of X in the graph.

The following example illustrates the concept of CP-Net as a formalism for specifying user's preferences.

Example 2 Let V = {Director (D), Genre (G) }, $\mathbf{dom}(D)$ = {Woody Allen (w), Nanni Moretti (n), Hitchcock (h)}, $\mathbf{dom}(G)$ = {comedy (c), drama (d), thriller (t)}. Let us suppose that I strictly prefer comedies to dramas and thrillers to comedies but my preference about film directors is conditioned to the film genre: I prefer Nanni Moretti's dramas toWoody Allen's dramas, and Woody Allen's dramas to Hitchcock's dramas. However, I prefer Woody Allen's comedies to Nanni Moretti's comedies and Nanni Moretti's comedies to Hitchcock's comedies. On the other hand, for thrillers I largely prefer Hitchcock's ones than Woody Allen's. But if I had to choose between a Woody Allen's thriller and a Nanni Moretti's thriller I would choose a Woody Allen's thriller. These preference rules can be expressed by the CP-Net depicted in Figure 1(a).

A CP-Net aims at capturing a *preference ordering* (a total order) over the objects in $\mathcal{O}$. Thus, the semantics of a CP-Net is defined as the set of preference orderings which are consistent with the preference constraints imposed by the given CP-Net. In Figure 1(b) one represents by thin arrows the relationships between objects which are entailed by the CP-Net of Figure 1(a). An arrow from object o to object o' means that $o \succeq o'$. The arrows resulting from transitivity (e.g. from (t, h) to (d, n)) are not showed in the figure. The thick arrows are not entailed by the CP-Net of Figure 1(a) but are consistent with it (see the discussion following Theorem 1 below).

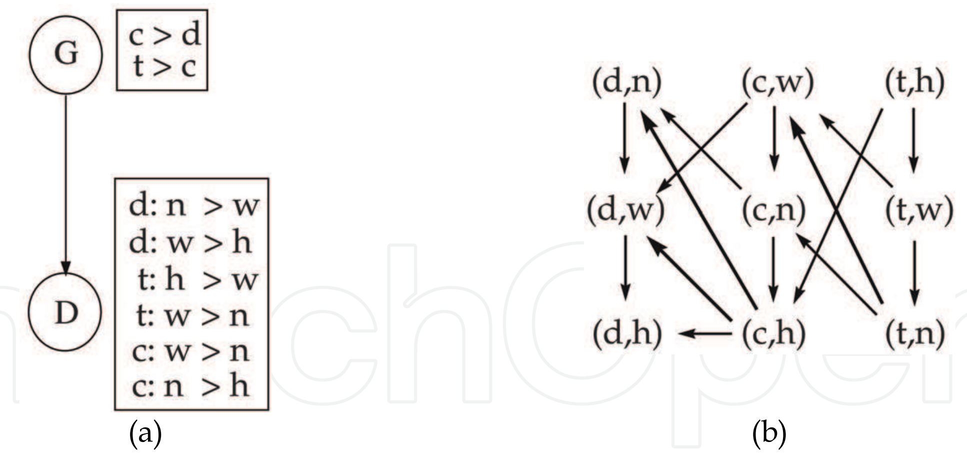


Fig. 1. (a) A CP-Net $\mathcal{N}$ (b) A preference ordering satisfying $\mathcal{N}$

Definition 2 (Satisfiability of a CP-Net) Let $\mathcal{N}$ be a CP-Net over the set of attributes V, $X \in V$ and $U \subset V$ the set of parents of X in $\mathcal{N}$. Let Y be the set of attributes other than X and its parents. Let $\succeq_{\mathbf{u}}$ be the ordering over **dom**(X) imposed by the CPT(X) for a given instantiation **u** of the attributes in U. We say that a preference ordering $\succeq$ over $\mathcal{O}$ *is compatible with* $\succeq_{\mathbf{u}}$ iff for all instantiations **y** of the attributes in Y we have $\mathbf{y}x\mathbf{u} \succeq \mathbf{y}x'\mathbf{u}$ iff $x \succeq_{\mathbf{u}} x'$. A preference ordering $\succeq$ *satisfies* the CP table CPT(X) iff it is compatible with $\succeq_{\mathbf{u}}$ for any instantiation **u** of the attributes in U. We say that the preference ordering $\succeq$ *satisfies* the CP-Net $\mathcal{N}$ iff it satisfies all the CP tables of $\mathcal{N}$. A CP-Net $\mathcal{N}$ is *satisfiable* iff there exists some preference ordering $\succeq$ satifying it.

In Figure 1(b) it is depicted a preference ordering satisfying the CP-Net of Figure 1(a). The following theorem guarantees that for acyclic CP-Nets it is possible to build an ordering satisfying it.

Theorem 1 Every acyclic CP-Net is satisfiable.

The detailed proof of Theorem 1 can be found in (Boutilier et al., 2004). The ordering given by this theorem is built by induction on the number of attributes in the CP-Net and uses the topological ordering on these attributes induced by the acyclic graph. The preference ordering depicted in Figure 1(b) is obtained by using the construction of Theorem 1. The thick arrows are specific to this particular ordering. They are built respecting the ordering given in the CP table CPT(G). A preference ordering satisfying an acyclic CP-Net is not unique in general. For instance, if we consider an arrow going from (d, n) to (c, h) in Figure 1(a) instead of the opposite arrow depicted in this figure, and we keep the other arrows, we obtain another ordering satisfying $\mathcal{N}$.

Best Outcomes. Given an acyclic CP-Net $\mathcal{N}$, the task of determining the best outcomes with respect to the preference orderings satisfying $\mathcal{N}$ is very simple. Even if the preference ordering satisfying $\mathcal{N}$ is not unique, surprisingly, the best outcome is unique and independs on the particular preference ordering satisfying $\mathcal{N}$. The algorithm for building this unique best outcome consists in sweeping through the graph from ancestors to descendents instantiating each attribute to its most preferred value given the instantiation of its parents. We describe the process of determining the best outcome in the following example.

Example 3 (Producing the best outcome determined by a CP-Net) Let us consider the CP-Net of Example 2. We begin by choosing the best value for attribute G (the attribute with no ancestors). This best value is t. Next, we take the children of attribute G. In our case, we have only one child, the attribute D. For G instantiated as t, the best value for the attribute D is h. Then, the best outcome is the object (t, h), that is, the most preferred movie is a Hitchcock's thriller.

This process of sweeping through the graph from ancestors to descendents and instantiating the attributes with the most preferred values given the instantiation of their parents is called *forward sweep*. The following theorem guarantees that this procedure produces the best outcome. The proof can be found in (Boutilier et al., 2004).

Theorem 2 Let $\mathcal{N}$ be an acyclic CP-Net. The best outcome with respect to any preference ordering satisfying $\mathcal{N}$ is unique and is produced by the forward sweep procedure.

Discussion. The CP-Net model for preference reasoning is not restricted to acyclic graphs. The advantage of considering acyclic CP-Nets is that the acyclicity of the graph implies that the model is consistent, that is, the CP-Net induces a preference ordering over the objects. If the graph is cyclic, the existence of such preference ordering is not guaranteed. In (Domshlak & Brafman, 2002) some initial results on consistency testing for cyclic CP-Nets were presented. More recently (Prestwich et al., 2005) showed that the optimal outcomes of an unconstrained (and possibly cyclic) CP-Net are the solutions of a set of hard constraints. They proposed a new algorithm for finding optimal outcomes which makes use of hard constraint solving. This new algorithm works even for cyclic CP-Nets. Besides, it works also with any preference formalism which produces a preorder over the outcomes. Another aspect which has to be considered is the constraint enforcing that in each CP table CPT(X), the domain **dom**(X) is totally ordered. The general definition of a CP-Net allows an arbitrary total *preorder* over **dom**(X), that is, the antisymmetric property is not required to be satisfied ($a \succeq b$ and $b \succeq a$ do not imply $a = b$). The difficulty with such general CP-Nets is that consistency is not verified in general. In (Boutilier et al., 2004) it is proved that consistency can be guaranteed if some special conditions are verified by the acyclic CP-Net.

Besides the problem of finding the best outcomes determined by a CP-Net $\mathcal{N}$, two other problems are particularly important: the dominance problem and the ordering problem. Both problems involve the task of comparing two objects o and o'. The first problem asks if $\mathcal{N}$ can deduce $o \succeq o'$(denoted by $\mathcal{N} \models o \succeq o'$). That is, it asks if *for all preference orderings* $\succeq$ *consistent with* $\mathcal{N}$ *it is true that* $o \succeq o'$. The second problem asks if the CP-Net is incapable of deducing $o' \succeq o$ (denoted by $\mathcal{N} \not\models o' \succeq o$). That is, it asks if *there exists a preference ordering* $\succeq$ *consistent with* $\mathcal{N}$ *such that* $o \succeq o'$. The second problem is easier than the first one. It can be proven that for acyclic CP-Nets, the complexity of determining the truth of at least one of the orderings queries $\mathcal{N} \not\models o' \succeq o$ or $\mathcal{N} \not\models o \succeq o'$ is $O(n)$ over the number n of attributes involved in the CP-Net $\mathcal{N}$ (Boutilier et al., 2004). On the other hand, the dominance problem is polynomial (when the graph verifies some conditions) and NP-complete in general. For a deeper discussion on these topics, see (Boutilier et al, 2004). In (Boutilier et al. 1997) it was shown that the dominance problem is intrinsically related to the problem of finding optimal outcomes satisfying a set of given constraints (the constraint-based preferential optimization problem in CP-Nets).

2.2 Tradeoffs-enhanced conditional preference networks (TCP-Nets)

In preference elicitation with CP-Nets the user describes how her preference over the values of an attribute depends on the values of other attributes. CP-Nets are able to specify a class of intuitive and useful preference statements of the form: "*I prefer the value a_0 for attribute A given that B = b and C = c*". However, there are other intuitive and important preference statements which cannot be represented by a CP-Net. These statements have the form: "*It is more important to me that the value of attribute A be better than the value of attribute B be better*". For instance, I could say that when choosing a movie, a most preferred genre is more important than a most preferred director. So, when comparing the films $f_1 = (c, w)$ and $f_2 = (t, n)$ in Example 2, I would prefer f_2 to f_1. Notice that the CP-Net $\mathcal{N}$ given in Example 2 is not able to infer $f_2 \succeq f_1$ nor $f_1 \succeq f_2$. Another kind of intuitive statements which cannot be represented by a CP-Net has the form: "*Given that C = c, a better assignement of attribute A is more important to me than a better assignement of attribute B*". For instance, I could say that when choosing a movie produced in the 50's, a most preferred genre is more important than a most preferred director. However, for movies produced during the 60's, directors play a more important role in my decision than the movie genre.

A CP-Net is able to specify only one kind of relationship between attributes, the *conditional preference dependence* relationship. In this section we consider an extension of the CP-Net formalism allowing two other kind of relationships between attributes: *relative importance* (atribute A is more important than attribute B in my decision) and *conditional relative importance* (attribute A is more important than attribute B in my decision given that the value for attribute C is c_0). This enhanced model, introduced in (Brafman et al., 2006a), is called *Tradeoffs-enhanced Conditional Preference Network (TCP-Nets)*.

Like CP-Nets, TCP-Nets are annotaded graphs where nodes are attributes. Unlike CP-Nets, TCP-Nets have three types of edges. The first one corresponds to CP-Nets edges, indicating conditional preference between attributes. The second edge type (directed) capture *relative importance* of attribute X over attribute Y . More precisely, let X and Y be two attributes mutually preferenctially independent given $Z = V - \{X, Y\}$, that is, for every fixed instantiation of the attributes in Z, the ranking of X values is independent of the value of Y . We say that X is more important than Y , denoted $X \rhd Y$, if for every instantiation $\mathbf{z}$ of the attributes in Z and for every $x, x' \in \mathbf{dom}(X)$ such that $x \succ x'$ given $\mathbf{z}$, we have that $xy\mathbf{z} \succ x'y'\mathbf{z}$.

The third edge type (undirected) captures *conditional relative importance*. More precisely, let X and Y be a pair of attributes in V and let $Z \subseteq V - \{X, Y\}$. We say that X is more important than Y given $\mathbf{z} \in \mathbf{dom}(Z)$ iff for every $\mathbf{w} \in \mathbf{dom}(V - (\{X, Y\} \cup Z))$ we have: $xy\mathbf{zw} \succ x'y'\mathbf{zw}$ whenever $x \succ x'$ given $\mathbf{zw}$. We denote this relation by $X \rhd_{\mathbf{z}} Y$. Thus, an undirected edge of the third type between attributes X and Y , labelled with the set of attributes Z, means that $X \rhd_{\mathbf{z}} Y$ or $Y \rhd_{\mathbf{z}} X$, depending on the values of the attributes in Z. As in CP-Nets, each node X in a TCP-Net is annotaded with a CP table CPT(X). In addition, in TCP-Nets, each undirected edge labelled with Z between attributes X and Y is annotaded with a *conditional importance table (or CI table)* CIT(X, Y, Z), describing the relative importance of X and Y given the value of the corresponding importance-conditioning attributes Z.

Definition 3 (TCP-Net Preference Model) A TCP-Net $\mathcal{N}$ is a tuple (V, cp, i, ci, cpt, cit) where: (1) V is a set of attributes (the nodes of $\mathcal{N}$); (2) cp (conditional preference arcs) is a set of

directed arcs (X, Y), for $X, Y \in V$; (3) i (importance arcs) is a set of direct arcs (X, Y), for $X, Y \in V$ such that $X \triangleright Y$; ci (conditional importance) is a set of undirected arcs $\{X, Y\}$ labelled with a set of attributes Z such that $X \triangleright_{\mathbf{z}} Y$ or $Y \triangleright_{\mathbf{z}} X$ depending on the assignement $\mathbf{z}$ of attributes in Z ; *cpt* associates a CP table CPT(X) to each node X of $\mathcal{N}$, where CPT(X) is a mapping from **dom**(Parents(X)) to strict partial orders over **dom**(X); *cit* associates a CI table CTI(X, Y, Z) indicating, for each instantiation $\mathbf{z} \in$ **dom**(Z), the relative importance of X and Y.
The following example illustrates the concept of TCP-Net as a formalism for specifying preferences.
Example 4 Let V = {Director (D), Genre (G), Year (Y) }, **dom**(D) = {Woody Allen (w), Nanni Moretti (n)}, **dom**(G) = {comedy (c), drama (d)}, **dom**(Y) = {80,90}. Let us suppose that I strictly prefer comedies to dramas but my preference about directors is conditioned to the film genre: When choosing dramas, I prefer Nanni Moretti's to Woody Allen's. However, for comedies, I prefer Woody Allen's to Nanni Moretti's. When choosing a film, the year of production is more important for my decision than the director. When choosing a Woody Allen's film, its genre is more important to me than its year of production. But for Nanni Moretti's films, the year of production is more important than the genre. These preference rules can be expressed by the TCP-Net depicted in Figure 2(a).

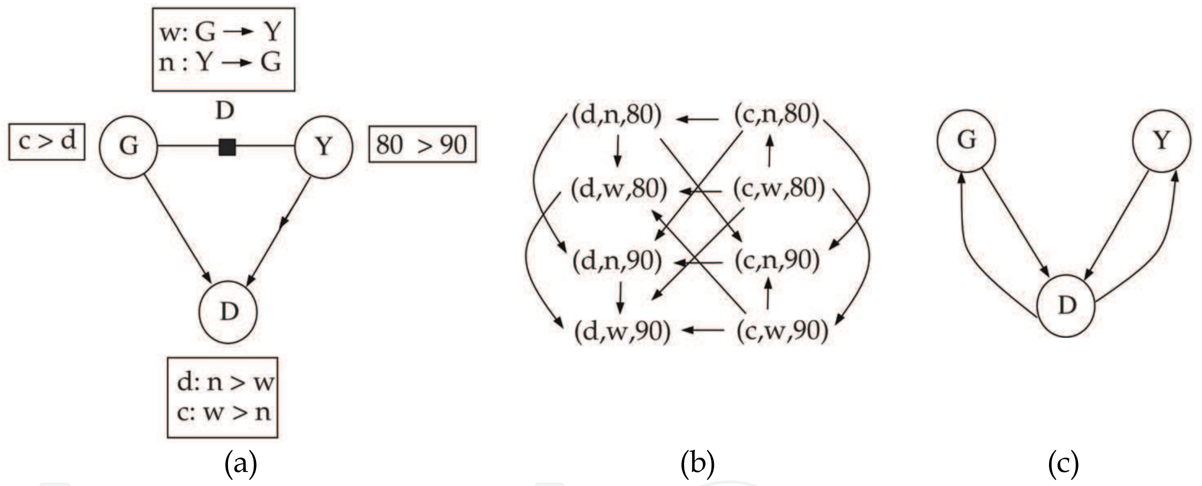


Fig. 2. (a) A TCP-net $\mathcal{N}$ (b) The partial ordering induced by $\mathcal{N}$ (c) The dependence graph

The semantics of a TCP-Net is defined in terms of the set of *strict partial orders* consistent with the constraints imposed by the preference and importance relations expressed by the graph edges, the CP and CI tables. As for CP-Nets, TCP-Nets semantics is based on the *ceteris paribus* semantics. We present here only the intuitive idea behind the semantics of a TCP-Net. For a more formal presentation, see (Brafman et al., 2006a). A strict partial order $\succ$ *satisfies* a TCP-Net $\mathcal{N}$ if the following intuitive conditions are verified: (1) in each CP table CPT(X), for every $\mathbf{z} \in$ **dom**(Z) (where Z = Parents(X)), two objects o and o' differing only on the attribute X and whose values on Z are given by $\mathbf{z}$, are ordered by $\succ$ consistently with the ordering on the X values given in CPT(X); (2) if $X \triangleright Y$, then any two objects o and o' differing only on the values of X and Y are ordered as $o \succ o'$ if the relationship $o[X] \succ o[X']$ appears in the CP table CPT(X) corresponding to the instantiation given by o[Parents(X)]; (3) in each CI table CPI(X, Y,Z), for every $\mathbf{z} \in \mathbf{dom}(Z)$ such that $X \rhd_{\mathbf{z}} Y$, any two objects o and o' differing only on the attributes X and Y and whose values on Z are given by $\mathbf{z}$ are ordered as $o \succ o'$ if the relationship $o[X] \succ o[X']$ appears in the CP table CPT(X) corresponding to the instantiation given by o[Parents(X)]. In Figure 2(b) one represents the relationships between objects which are entailed by the TCP-Net in Figure 2(a). The arrow from object o to object o' means that $o \succ o'$. The arrows resulting from transitivity are not showed in the figure. Notice that the arrow from film (*d, n,* 80) to film (*c, n,* 90) is inferred using the CI table CIT(G, Y,D) which imposes that, for Nanni Moretti's movies, the year of production is more important than the genre. So, as I prefer films produced in the 80's than films produced in the 90's, I prefer the first film to the second one, even if the first film is a drama and the second one is a comedy.

Definition 4 (Satisfiability of a TCP-Net) A TCP-Net $\mathcal{N}$ is satisfiable (or consistent) iff there is some strict partial order $\succ$ over $\mathcal{O}$ that satisfies it. Let $o, o' \in \mathcal{O}$. We say that $o \succ o'$ is implied (or inferred) by the TCP-Net $\mathcal{N}$ iff it is verified by all strict partial orders $\succ$ over $\mathcal{O}$ satisfying $\mathcal{N}$.

Satisfiability is a desired property for TCP-Nets since it is important to guarantee that the preference rules provided by the users do not lead to inconsistencies like "I prefer object o to object o' and object o' to object o". However, the definition of TCP-Net satisfiability does not provide a mechanism for testing TCP-Net consistency. Fortunately, for a large class of TCP-Nets consistency is guaranteed. This class of TCP-Nets is referred as *conditionally acyclic* and is defined as follows:

Definition 5 (Conditionally Acyclic TCP-Nets) Let $\mathcal{N}$ be a TCP-Net over the set of attributes V . We associate to $\mathcal{N}$ a graph $\mathcal{N}^*$, called *the dependency graph of* $\mathcal{N}$ in the following way: the nodes of $\mathcal{N}^*$ are the same as the nodes of $\mathcal{N}$. Each directed edge of $\mathcal{N}$ is a directed edge of $\mathcal{N}^*$. For each undirected edge $\{X, Y\}$ of $\mathcal{N}$, labelled by the set of attributes Z, we insert in $\mathcal{N}^*$ two directed edges (A,X) and (A, Y) for each attribute $A \in Z$. Besides, for each assignement $\mathbf{z} \in \mathbf{dom}(Z)$ of the attributes in Z, we insert a direct edge (X,Y) or (Y,X) depending on the information given in the CI table CIT(X,Y,Z) corresponding to the assignement $\mathbf{z}$. In that way, we are able to associate a set of directed graphs $\mathcal{G}(\mathcal{N})$ to the TCP-Net $\mathcal{N}$, one for each assignement of the attributes labelling the undirected edges of $\mathcal{N}$. We say that the TCP-Net is *conditionally acyclic* if each graph of $\mathcal{G}(\mathcal{N})$ is acyclic.

For instance, the dependence graph associated to the TCP-Net of Figure 2(a) is given in Figure 2(c). As we see, this TCP-Net is not conditionally acyclic, since the graph $\mathcal{N}^*$ is cyclic. Now, if we consider the TCP-Net depicted in Figure 3(a), it is easy to see that it is conditionally acyclic, since all graphs in $\mathcal{G}(\mathcal{N})$ (showed in Figure 3(b)) are acyclic.

For conditionally acyclic TCP-Nets we have the following result, whose proof can be found in (Brafman et al. 2006a).

Theorem 3 Every conditionally acyclic TCP-Net is satisfiable.

Discussion. (1) *Complexity*: Unfortunately, testing for conditionally acyclicity is not an easy task. This problem is shown to be coNP-hard in (Brafman et al. 2006a). (2) *Best Outcomes*:

One of the central properties of the CP-Net model is that, given an acyclic CP-Net $\mathcal{N}$ and a (possibly empty) partial instantiation **x** of some of its attributes, it is simple to determine a *best object* consistent with **x**. In the previous section, we presented the forward sweep procedure which produces the best object of an acyclic CP-Net. This procedure works also for conditionally acyclic TCP-Nets. The relative importance relations do not have any influence in the process of obtaining the optimal outcome. In order to obtain the best object, we simply consider the CP-Net part of the TCP-Net $\mathcal{N}$ (ignoring the *i*-edges and the *ci*-edges) and we apply the forward sweep procedure for the resulting CP-Net. This simple algorithm for finding the best outcome can be applied to all TCP-Nets for which the CP-Net part is acyclic. In particular, it is applicable for conditionally acyclic TCP-Nets. However, finding the best outcome associated to a TCP-Net $\mathcal{N}$ satisfying a set of hard constraints is not trivial. In (Brafman et al., 2006a), an algorithm (Search-TCP) is developed for producing the best outcomes associated to a conditionally acyclic TCP-Net $\mathcal{N}$ satisfying a set of hard constraints $\mathcal{C}$ on the attributes of $\mathcal{N}$.

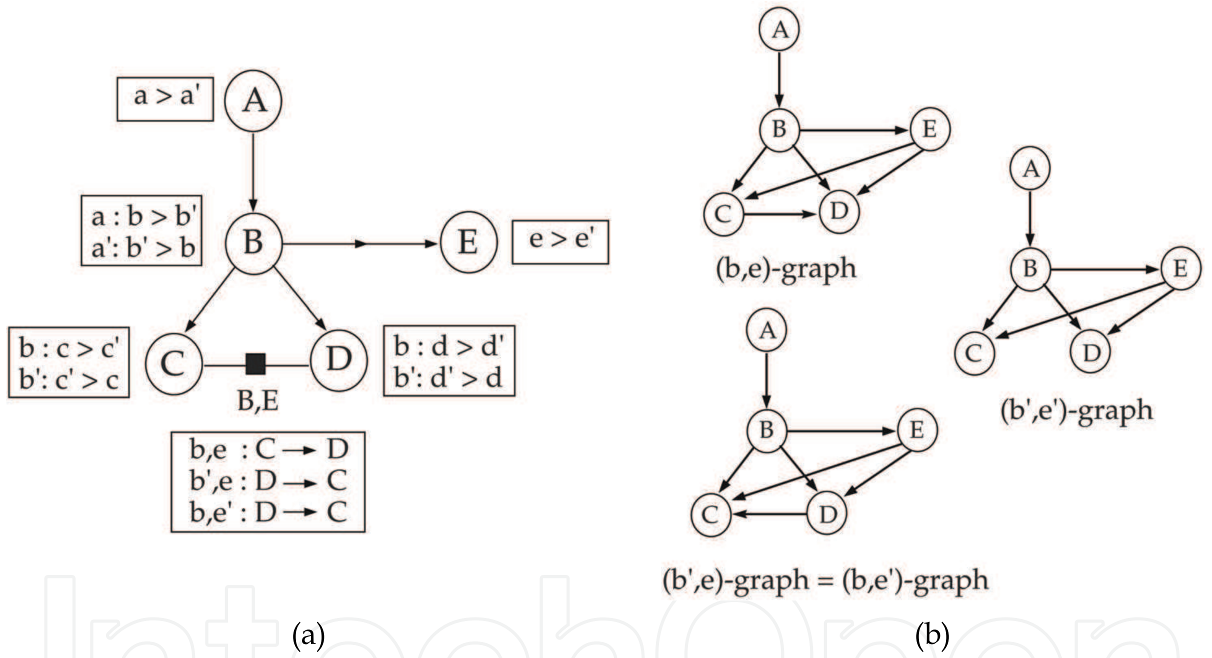


Fig. 3. (a) A conditionally acyclic TCP-net $\mathcal{N}$ (b) The set of acyclic graphs $\mathcal{G}(\mathcal{N})$

2.3 A logical framework for preferences over objects

In this section we present a third approach for preference elicitation and reasoning introduced in (Wilson, 2004). This approach is based on a logical framework and generalizes the CP-Nets and TCP-Nets approaches.

The Preference Language $\mathcal{L}$. The language $\mathcal{L}$ is constituted by statemets of the form φ: $u \rightarrow (X = x) > (X = x')$, where u is a formula of the form $(X_{i_1} = x_1) \wedge \ldots \wedge (X_{i_k} = x_k)$, with $X_{i_j} \in V - \{X\}$ and $x_j \in \mathbf{dom}(X_{i_j})$ for all $j \in \{1, \ldots, k\}$ and $x, x' \in \mathbf{dom}(X)$. We call such statements *conditional preference rules* or *cp-rules* for short. The formula u is called the *condition* of the cp-rule φ. The set of attributes appearing in u is denoted by *Attr(u)*. If φ is the statement $u \rightarrow (X$

$= x) > (X = x')$ then sometimes we denote u by u_{φ}, X by X_{φ} and x, x' by x_{φ} and x'_{φ} respectively. A *conditional preference theory* over V is a finite set of statements of $\mathcal{L}$.

Example 5 Let $V = \{G,D\}$ as in Example 2. Let φ_1 and φ_2 the following conditional preference rules:

$\varphi_1 : (G = c) \to (D = w) > (D = n)$,

$\varphi_2 : (D = n) \to (G = c) > (G = d)$.

Then $\Gamma = \{\varphi_1, \varphi_2\}$ is a conditional preference theory which expresses the first preference statement of Example 1.

A conditional preference statement $\varphi : u \to (X = x) > (X = x')$ induces a preference ordering on objects over V. Let $o = tyx$ and $o' = tyx'$ be objects over V, where y is an object over $Attr(u)$, t is an object over $V - (Attr(u) \cup \{X\})$. We say that o is *preferred* to o' according φ. The set of pairs of objects (o, o') where o is preferred to o' according to φ is denoted by φ^*. If Γ is a conditional preference theory, we denote by $>_\Gamma$ the transitive closure of the binary relation $\Gamma^* = \bigcup_{\varphi\in\Gamma} \varphi^*$.

Example 6 Let us compare the objects $o_1 = (c,w)$ and $o_2 = (d, n)$ according to Γ. We have that (c,w) is preferred to (c, n) according to φ_1. And (c, n) is preferred to (d, n) according to φ_2. Then, using transitivity, we conclude that (c,w) is preferred to (d, n), that is, $o_1 >_\Gamma o_2$.

Consistency Test. One important feature of preference conditional theories is that there is no need of eliciting a total order on the values of an attribute given each assignement to its parents, as in the CP-Net preference model. So, a conditional preference theory is a compact way of expressing preference: we can reason with any theory Γ specified by the user, provided this theory satisfies some properties which guarantee its *consistency*. Besides, the user can add new statements later on; because the logic used in the deduction system is monotonic, all previous deductions concerning preferences will hold.

Now, we present the concept of *consistency* for a preference conditional theory Γ. A *model* of Γ is a strict partial order (that is, a transitive and irreflexive relation) $>$ on objects $\mathcal{O}$ over V such that $>$ contains the induced ordering $>_\Gamma$. We say that Γ is *consistent* if there exists a model $>$ for Γ. It is easy to see that a theory Γ is consistent if and only if its induced relation $>_\Gamma$ is irreflexive, since $>_\Gamma$ is transitive by definition.

Example 7 The theory Γ presented in Example 5 is consistent. Indeed $>_\Gamma = \{(o_1, o_3), (o_3, o_4), (o_1, o_4)\}$ is a strict partial order over the set of objects $\mathcal{O} = \{o_1, o_2, o_3, o_4\}$, where $o_1 = (c,w)$, $o_2 = (d,w)$, $o_3 = (c, n)$, $o_4 = (d, n)$. Note that $(o_1, o_3) \in \varphi_1^*$, $(o_3, o_4) \in \varphi_2^*$ and (o_1, o_4) is inferred by transivity. However, the theory $\Gamma' = \Gamma \cup \{\varphi_3, \varphi_4\}$ where: $\varphi_3 : (G = d) \to (D = n) > (D = w)$ and $\varphi_4 : (D = w) \to (G = d) > (G = c)$, is not consistent. Indeed, $(o_4, o_2) \in \varphi_3^*$ and $(o_2, o_1) \in \varphi_4^*$. So, $o_1 >_{\Gamma'} o_1$, since $(o_4, o_1) \in >_{\Gamma'}$ and $(o_1, o_4) \in >_{\Gamma'}$, which proves that $>_{\Gamma'}$ is not irreflexive.

We associate to each preference conditional theory Γ a graph $G(\Gamma)$ defined as follows: the nodes of $G(\Gamma)$ are the attributes appearing in the rules of Γ and the set of edges is given by $\{(Y,X_\varphi) : Y \in U_\varphi\}$, where U_φ denotes the set of attributes appearing in the condition u_φ. The preference conditional theory Γ is *acyclic* if its graph $G(\Gamma)$ is acyclic.

As we will see in Theorem 4, in order to ensure consistency for acyclic theories it will be sufficient to ensure *local consistency*. More precisely : Let o be a fixed object over V and X be an attribute in V . Let $x, x' \in \mathbf{dom}(X)$. We say that (x, x') is *validated* by o if there exists a statement $(\varphi : u_\varphi \rightarrow X = x > X = x') \in \Gamma$ such that o satisfies the formula u_φ (the conditions of φ). We define the relation $>_o^X$ on $\mathbf{dom}(X)$ as the transitive closure of the set of all pairs (x, x') validated by o. We say that the preference theory Γ is *locally consistent* if for all objects o and all attributes X, the relation $>_o^X$ is irreflexive.

Example 8 Let us consider the situation of Example 7 except that the set of attributes V is augmented with a third attribute Y (year of production), so $V = \{G, D, Y\}$. Let us consider the preference theory $\Gamma_1 = \{\varphi_1, \varphi_5\}$, where $\varphi_5 : (Y = 1990) \rightarrow (D = n) > (D = w)$. Let $o = (c, w, 1990)$ and let us fix the attribute D. Then $w >_o^D n$ since (w, n) is validated by o, if we consider the statement φ_1. But (n,w) is also validated by o, if we consider the statement φ_5. Thus, Γ_1 is not locally consistent.

The following theorem gives necessary and sufficient conditions for ensuring consistency of a preference theory Γ.

Theorem 4 Let Γ be a conditional preference theory. Then, we have : (1) If Γ is consistent then Γ is local consistent. (2) If Γ is local consistent and acyclic then Γ is consistent. (3) If all the attributes in V are binary, local consistency can be determined in time proportional to $|\Gamma|^2 \times |V|$.

The theory Γ_1 presented in Example 8 is not locally consistent, so it is not consistent by Theorem 4. Notice that its graph $G(\Gamma_1) = \{(G,D),(Y,D)\}$ is acyclic. On the other hand, the theory Γ given in Example 5 is consistent but its graph $G(\Gamma) = \{(G,D),(D,G)\}$ is cyclic. By Theorem 4 we can conclude that it is local consistent. This is an example of a local consistent theory whose graph is cyclic.

Finding optimal outcomes. Let Γ be a preference conditional theory over a set of attributs V. Given an object o over $V' \subseteq V$, we say that a value $x_i \in \mathbf{dom}(X_i)$ is undominated given the object o if there is no statement $u \rightarrow (X_i = x) > (X_i = x_i)$ in Γ, such that o satisfies u. The algorithm for finding optimal objects with respect to a locally consistent preference theory Γ with acyclic $G(\Gamma)$ works as follows: (1) enumerate the attributes of V in such a way that the ordering $\langle X_1, ..., X_n\rangle$ is compatible with the graph $G(\Gamma)$ (that is, if $i > j$ then there is no path going from X_j to X_i in $G(\Gamma)$). (2) For each $i \in \{1, ..., n\}$ let $\alpha(X_i) = x$, where $x \in \mathbf{dom}(X_i)$ and x is undominated with respect to the object $o = (\alpha(X_1), ..., \alpha(X_{i-1}))$. Local consistency of Γ ensures that such x always exists.

Example 9 Let us consider $V = \{G, D, Y\}$ as in Example 8. Let us consider the preference theory $\Gamma_1 = \{\varphi_1, \varphi_6\}$, where $\varphi_6 : (G = d) \rightarrow (Y = 1990) > (Y = 2000)$. We have $G(\Gamma_1) = \{(G,D), (G, Y)\}$. Then, the ordering $\langle G, D, Y\rangle$ is compatible with $G(\Gamma_1)$. We choose $\alpha(G) = c$. For $\alpha(D)$, the only undominated value given (c) is w. For $\alpha(Y)$, both values 1990 and 2000 are undominated given (c,w). So, a best object is $o_1 = (c,w, 1990)$. Another one is $o_2 = (c,w, 2000)$. By choosing $\alpha(G) = d$, we also get $o_3 = (d,w, 1990)$ and $o_4 = (d, n, 1990)$ as best objects.

3. Preferences over sets of objects

For the time being, we have been interested in formalisms allowing to eliciting and reasoning with preferences over objects. We have introduced some important frameworks for specifying user's preferences in a compact way, besides discussing important issues related to this topic, such as decidability and complexity of the problems of finding the best objects, dominance and ordering queries and introduction of hard constraints. In this section we tackle these issues in a broader context, by considering a simple framework for dealing with preferences over *sets of objects*. This problem arises naturally in the context of our running example. Let us suppose the task of creating a program for a film festival. Here, the crucial task is not to obtain user's preferences about movies considered individually, but about several possible *sets* of movies. Thus, the user has to be able to specify her preferences about a group of films, taking into account aspects like genre diversity, genre adaptability (for instance, a user may not be interested in programs containing both comedies and dramas), etc. In such situation, we would like to be able to determine the preferred characteristics which must be satisfied by a *group of objects*, and then to be able to select from a set of objects the best subset satisfying these preference rules. A simple way to treat the problem of finding the best subset of objects is to produce a set containing the k best elements according to a set of preference rules on individual objects. This naive solution is not suitable since the attractiveness of particular objects does not imply that these same objects put together would constitute an atractive set. If one of the requirements for a "good" set is the diversity of its elements, putting together a set of good objects would not necessarily produce the required diversity. Several recent works on preference modelling in AI have focused on eliciting and reasoning with preference over sets of objects, from a quantitative and a qualitative perspectives. In (des Jardins & Wagstaff, 2005) for instance, it was proposed a formalism to deal with preferences over sets of objects supporting the notions of *diversity* and *depth*. These concepts allows expressing preferences in a quantitative way, by *measuring* in some sort the degree of diversity and depth of a preferred set of objects. Since in this chapter we are focusing on formalisms based on a qualitative perspective, we will describe here the very simple and elegant approach of (Brafman et al., 2006b). This approach allows the user to specify a broad class of interesting properties about sets of objects. And surprisingly, such set-preference statements can be naturally transformed into conditional preference statements over attributed objects.

The Specification Language. Most properties of sets of objects which are important for users when specifying their preferences take the forms: (1) "*at least one object in the set satisfies* $C = c$ *and* $D = d$ *or* $A = a$; (2) *the number of objects satisfying* $C = c$ *is* 2. Let $\mathcal{L}$ the propositional language where the propositions are of the form $X = x$, where X is an attribute in the set V of attributes and $x \in \mathbf{dom}(X)$. An object $o \in \mathcal{O}$ *satisfies* $X = x$ if $o[X] = x$. This notion of *satisfaction* is extended to formulae $\varphi \in \mathcal{L}$ as usually in Propositional Logic. It is denoted by $o \models \varphi$. Now we consider the following class of properties $\mathcal{C}$ over sets of objects : $\langle |\varphi| \ \theta n \rangle$, where $\varphi \in \mathcal{L}$, $\theta \in \{=, \leq, \geq, >, <\}$, $n \in \mathbb{N}$. Using such statements, the user is able to express the properties about sets of objects which may affect her preferences. These properties refers to the number of objects in the selected subset O of objects verifying some constraints. The following example illustrates these properties:

Example 10 Let us consider the situation depicted in Example 2, but with an extra attribute standing for the film title. So, $V = \{G, D, T\}$. Let us suppose the following properties which affect user's preferences about a film program: the fact the it contains at most two Woody Allen's comedies, at least two Hitchcock's thrillers and no dramas. This can be specified by the following set-preference properties:

P_1: $\langle |G = c \wedge D = w| \leq 2\rangle$

P_2: $\langle |G = t \wedge D = h| \geq 2\rangle$ P_3: $\langle |G = d| = 0\rangle$

Let us consider the following set of films: $O = \{(c,w, t_1), (c,w, t_2), (c,w, t_3), (d, n, t_4), (t, h, t_5), (t, h, t_6)\}$. For this set we have $P_1(O)$ = false, $P_2(O)$ = true and $P_3(O)$ = false.

Now, let us consider a set of properties $\mathcal{P} = \{P_1, ..., P_n\} \in \mathcal{C}$. Each property P_i can be treated as an attribute taking values in the set {true, false} ($\mathbf{dom}(P_i)$ = {true, false}). Each subset of objects $O \subset \mathcal{O}$ corresponds to an "object" in $\mathcal{V} = \mathbf{dom}(P_1) \times .... \times \mathbf{dom}(P_n)$. That is, abstractly, each subset O can be viewed as a vector of truth-values (an object). Moreover, any preference order over objects in $\mathcal{V}$ implicitly induces a preference order over sets of objects of $\mathcal{O}$. So, in order to specify preferences over sets of objects, the user must simply specify (1) which are the properties about sets that affects her preferences and (2) her specific preference rules involving the validity of these properties. After such specifications, the problem of extracting a preference ordering over sets of objects satisfying the user's requirements is reduced to the problem of extracting a preference ordering over objects. Thus, we can use one of the formalisms introduced in the previous section for reasoning with preferences over objects in order to infer a preference ranking over sets of objects. The following example illustrates this idea.

Example 11 Let us consider the situation of our Example 10. Let us suppose the user specifies the following preference statements: (1) She prefers programs containing at most two Woody Allen's comedies; (2) For programs containing more than two Woody Allen's comedies she prefers a program containing at least one drama; (2) For programs containing no dramas he prefers a program containing at least two Hitchcock's thrillers. These preference statements can be represented by the TCP-Net depicted in Figure 4.

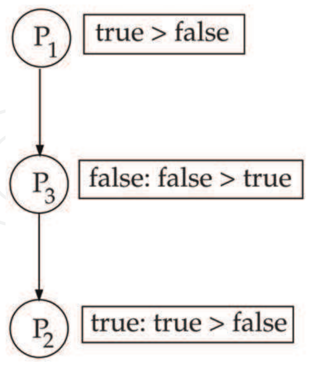


Fig. 4. A TCP-Net representing set-preference statements

4. Preferences over sequences of objects

In this section, we present our formalism allowing to specify compact preference statements provided by the users. First, we will formalize the notion of *temporal conditions* used for ranking sequences of objects. By viewing each object in a sequence σ as a state, we propose to use the formalism of Propositional Linear Temporal Logic (PTL) to capture the desired properties of sequence of objects, which we call *temporal conditions*. After formalizing our temporal conditions, we introduce the language **TPref** for expressing conditional preferences over sequences of objects. Preference statements in **TPref** use temporal conditions in their formulation.

4.1 Temporal conditions

The language we use for expressing temporal condition is basicly the Propositional Temporal Logic (PTL), adapted to our context. In PTL, the basic formulae are propositional variables $p_1, ..., p_n$. In our case, basic formulae or *propositions* are of the form $X = a$ where $X \in V$ and $a \in$ **dom(X)**. In order to emphasize the fact that our language assume a particular basic formula format, we will call it STL (for Simple Temporal Logic) instead of PTL. We stress however that both logics are essentially the same.

Definition 6 (The language STL for temporal conditions) The STL formulae are defined as follows: (1) **true** and **false** are STL formulae. (2) if P is a proposition then P is a STL formula. (3) if F and G are STL formulae then $F \wedge G$, $F \vee G$ and $\neg F$ are STL formulae. (4) if F and G are STL formulae then F **Until** G and F **Since** G are STL formulae. A *temporal condition* is a STL formula. If F is a temporal condition, we denote by Attr(F) the set of attributes appearing in F.

Next, we present the semantics of temporal conditions. Temporal conditions are evaluated over *sequences* of objects. A *sequence of objects* of $\mathcal{O}$ is a structure consisting of a set of objects $\{o_1, o_2, ..., o_k\}$ with an (temporal) ordering $o_1 < o_2 < ... < o_k$, telling us that o_i comes before o_{i+1}. We denote this structure simply by $\sigma = \langle o_1, o_2, ..., o_k\rangle$. If $\sigma = \langle o_1, ..., o_k\rangle$ then k is called the *length* of σ and is denoted by $|\sigma|$. We denote by $Seq(\mathcal{O})$ the set of sequences of objects in $\mathcal{O}$ and by $Seq_n(\mathcal{O})$ the set of sequences of length n in $Seq(\mathcal{O})$.

Definition 7 (STL Semantics) The notion of *satisfaction* of a **STL** formula by a sequence of objects $\sigma = \langle o_1, ..., o_k\rangle$ at a state $i \in \{1, ..., k\}$ (denoted by $(\sigma, i) \models F$) is inductively defined as follows: (1) $(\sigma, i) \models (X = a)$ iff $o_i[X] = a$; (2) $(\sigma, i) \models F \wedge G$ iff $(\sigma, i) \models F$ and $(\sigma, i) \models G$; (3) $(\sigma, i) \models F \vee G$ iff $(\sigma, i) \models F$ or $(\sigma, i) \models G$; (4) $(\sigma, i) \models \neg F$ iff $(\sigma, i) \not\models F$;
(5) $(\sigma, i) \models F$ **Until** G iff there exists j such that $i < j \leq |\sigma|$ and $(\sigma, j) \models G$ and for all k such that $i < k < j$ we have $(\sigma, k) \models F$.
(6) $(\sigma, i) \models F$ **Since** G iff there exists j such that $1 \leq j \leq i$ and $(\sigma, j) \models G$ and for all k such that $j < k < i$ we have $(\sigma, k) \models F$.
We say that σ *satisfies* a **STL** formula F (denoted by $\sigma \models F$) if $(\sigma, k) \models F$, where $k = |\sigma|$. We say that F is *satisfiable* if there exists $\sigma \in Seq(\mathcal{O})$ such that $\sigma \models F$. The formula **true** (resp. **false**) is satisfied by any sequence (resp. by no sequence) $\sigma \in Seq(\mathcal{O})$. We say that two **STL** formulae F,G are *equivalent* iff for every sequence σ, $\sigma \models F$ iff $\sigma \models G$. We say that F,G are *globally equivalent* (g-equivalent) iff for every sequence σ, $(\sigma, i) \models F$ iff $(\sigma, i) \models G$, for all $i \in \{1, ..., |\sigma|\}$.

Derived Formulae:
Prev F = **false Since** F ("in the previous state F"); **Next** F = **false Until** F ("in the next state F"); **First** = ¬ **Prev true** (*"I am at the first state"*); **Last** = ¬ **Next true** (*"I am at the last state"*).; $\blacklozenge F$ = **true Since** F (*"Sometimes in the past F"*); $\Diamond F$ = **true Until** F (*"Sometimes in the future F"*); $\blacksquare F = \neg\blacklozenge\neg F$ (*"Always in the past F"*); $\Box F = \neg\Diamond\neg F$ (meaning *"Always in the future F"*)

A very important property verified by PTL formulae (and consequently, by STL formula) is the *separability property*: it says that every PTL formula is g-equivalent to a boolean combination of *pure past*, *pure future* and *pure present* formulae. Let us define these kind of formulae:

Definition 8 (Present, Past and Future Formulae) A *pure present* formula is inductively defined by the following rules: (1) a proposition $X = a$ is a pure present formula. (2) a boolean combination of pure present formulae is a pure present formula. A *pure past* formula (resp. a *pure future* formula) is inductively defined as follows: (1) if F and G are pure present formulae then F **Since** G (resp. F **Until** G) are pure past formula (resp. a pure future formula). (2) If F and G are pure past formulae (resp. pure future formulae) then F **Since** G (resp. F **Until** G) is a pure past formula (resp. a pure future formula). (3) a boolean combination of pure past formulae (resp. pure future formulae) is a pure past formula (resp. a pure future formula). We say that a formula F is *separated* if F is of the form $F_1 \vee \ldots \vee F_n$, with each F_i of the form $F_i^0 \wedge F_i^+ \wedge F_i^-$, where F_i^0 is pure present formula, F_i^+ is a pure future formula and F_i^- is a pure past formula.

From a semantic point of view, the pure present, pure past and pure future formulae verifies the following properties which are easily proved by induction on the formulae construction.

Proposition 1 Let F be a STL formula.

- F is a pure present formula iff for all $\sigma = \langle o_1, \ldots, o_{i-1}, o_i, o_{i+1}, \ldots, o_k\rangle \in Seq_k(\mathcal{O})$ we have: $(\sigma, i) \models \varphi$ iff $(\sigma', i) \models \varphi$ for any sequence $\sigma' = \langle o_1, \ldots, o_{i-1}, o'_i, o_{i+1}, \ldots, o_k\rangle \in Seq_k(\mathcal{O})$ which differ from σ only at state i.
- F is pure past formula iff for all $\sigma = \langle o_1, \ldots, o_{i-1}, o_i, \ldots, o_k\rangle \in Seq_k(\mathcal{O})$ we have: $(\sigma, i) \models \varphi$ iff $(\sigma', i) \models \varphi$ for any sequence $\sigma' = \langle o_1, \ldots, o_{i-1}, o'_i, \ldots, o'_k\rangle$.
- F is pure future formula iff for all $\sigma = \langle o_1, \ldots, o_i, o_{i+1} \ldots, o_k\rangle \in Seq_k(\mathcal{O})$ we have: $(\sigma, i) \models \varphi$ iff $(\sigma', i) \models \varphi$ for any sequence $\sigma' = \langle o'_1, \ldots, o'_i, o_{i+1}, \ldots, o_k\rangle$.

Intuitively, pure past formulae are not "aware" of what is happening in the current state or in future states. Pure present formulae are not "aware" of what has happened in the past states or of what is going to happen in future states. And pure future formulae are not "aware" of what is happening in the current state or has happened in past states.

Theorem 5 (Separation Theorem (Gabbay, 1989)) Let F be a STL formula. Then F is g-equivalent to a separated formula.

For instance, $\Diamond((X = a) \wedge \blacksquare(Y = b))$ is not separated but is equivalent to the separated formula $\blacksquare(Y = b) \wedge (Y = b) \wedge ((Y = b)$ **Until** $(X = a))$. The property of separation of propositional temporal formulae is not trivial. In fact, separation is closely related to the expressivity power of a temporal language. For details on this important subject see (Gabbay, 1989). For a discussion about open problems concerning the complexity of separating a formula into its past, future and present components see (Hodkinson & Reynolds, 2005).

4.2 A temporal preference language

Now, we introduce the specification language for our temporal preference model. A temporal preference will be characterized by a set of *temporal conditional preference rules* that we formally define next.

Definition 9 (Temporal Conditional Preference Rule) A*temporal conditional preference rule* (or *tcp-rule*) is an expression of the form: $\varphi : F \rightarrow (X = x > X = x')$ where $X \in V$, $x, x' \in$ **dom**(X) and F is a STL separated formula. A *simple* tcp-rule is a tcp-rule where the temporal condition contains a unique disjunct. It is easy to see that a tcp-rule is equivalent to a set of *simple* tcp-rules.

Definition 10 (Temporal Conditional Preference Theory) A *Temporal Conditional Preference-Theory* is a finite set Φ of simple tcp-rules $F \rightarrow (X = x) > (X = x')$, where $X \notin \mathrm{Attr}(F_0)$. In what follows, sometimes it will be useful to use the following notation for the elements appearing in a tcp-rule φ: $F^-_\varphi \wedge F^0_\varphi \wedge F^+_\varphi$ denotes its temporal condition and $(X_\varphi = x_\varphi) > (X_\varphi = x'_\varphi)$ denotes the expression appearing in its right side.

Example 12 Let us consider the situation of our film festival program presented in Example 1. The statements are expressed by the following tcp-rules:

1. $\varphi_1 : (G = c) \rightarrow (D = w) > (D = c)$
2. $\varphi_2 : (G = d) \rightarrow (D = n) > (D = w)$. Here, the conditions in the tcp-rules are pure present formulae.
3. φ_3 : **First** $\rightarrow (G = c) > (G = d)$. Here, the condition in the tcp-rule is a pure past formula since **First** ≡ **Prev False**.
4. φ_4 : **Prev**$(G = c) \rightarrow (G = d) > (G = c)$
5. φ_5 : **Prev**$((G = d) \wedge (D = w)) \rightarrow (G = c) > (G = d)$
6. φ_6 : **Prev**$((G = d) \wedge (D = n)) \rightarrow (G = d) > (G = c)$. Here, the conditions in the tcp-rule are pure past formulae.
7. φ_7: $(\Diamond(G = d) \wedge \blacklozenge(G = c)) \rightarrow (G = c) > (G = d))$. Here, the conditions in the tcp-rules are separated formulae of the form $F^- \wedge F^+$ (with pure past and pure future components only).

The ordering induced by a Temporal Preference Theory. First of all we will show how two sequences in $Seq(\mathcal{O})$, differing at one single position i, can be compared via a temporal preference theory. Afterwards, we show how two sequences in $Seq(\mathcal{O})$, differing in k positions $i_1, \ldots, i_k$ can be compared.

Definition 11 (Sequences differing at one single position) Let φ be a tcp-rule. Let R_φ be the relation over $Seq_n(\mathcal{O})$ defined as follows: if $\sigma = \langle o_1, \ldots, o_n \rangle$ and $\sigma' = \langle o'_1, \ldots, o'_n \rangle$ then $\sigma R_\varphi \sigma'$ iff there exists $j \in \{1, \ldots, n\}$ such that: (1) $o_j \neq o'_j$ and $o_i = o'_i$ for every $i \in \{1, \ldots, n\} \setminus \{j\}$; (2) $(\sigma, j) \models F_\varphi$ and $(\sigma', j) \models F_\varphi$; (3) $o_j[X_\varphi] = x_\varphi$ and $o'_j[X_\varphi] = x'_\varphi$; (4) For every $Y \in V \setminus \{X_\varphi\}$, $o_j[Y] = o'_j[Y]$. If such position j exists, it is unique and denoted by $\delta(\sigma, \sigma')$.

Thus, two sequences of the same size can be compared via R_φ only if they differ at one single position. Roughly speaking, in order to compare two sequences differing at $k > 1$ positions, via a temporal conditional preference theory Φ, we will consider the union of R_φ, for $\varphi \in \Phi$ and the transitive closure of this union. More precisely: Given a set Φ of tcp-rules, we denote by R_Φ the set $\bigcup_{\varphi \in \Phi} R_\varphi$ and by $>_\Phi$ the transitive closure of R_Φ. We say that σ is *preferred* to σ'

w.r.t. the theory Φ if $\sigma >_\Phi \sigma'$. Lemma 1 below gives a necessary and sufficient condition in order to a sequence σ be preferred to a sequence σ' w.r.t. Φ. Before stating this result, we need the following definition:

Definition 12 (Improving Flipping Sequence (IFS)) Let σ and σ' be two sequences of length n. We say that there exists an *Improving Flipping Sequence (IFS)* from σ to σ' w.r.t Φ if there exists a set of sequences $\{\sigma_1, \ldots, \sigma_{p+1}\}$ and a set of tcp-rules $\{\varphi_1, \ldots, \varphi_p\}$ in Φ such that $\sigma_1 = \sigma$, $\sigma_{p+1} = \sigma'$ and $\sigma_k\ R_{\varphi k}\ \sigma_{k+1}$ for every $k \in \{1, \ldots, p\}$.

Lemma 1 Let Φ be a set of tcp-rules. Let σ and σ' be two sequences of length n. Then $\sigma >_\Phi \sigma'$ iff there exists an IFS from σ to σ' w.r.t. Φ.

Example 13 Let us consider the theory $\Phi = \{\varphi_1, \ldots, \varphi_7\}$ of Example 12 and the following sequences: $\sigma_1 = \langle(c, n), (d,w)\rangle$, $\sigma_2 = \langle(d, n), (d,w)\rangle$ and $\sigma_3 = \langle(d, n), (c,w)\rangle$. Note that $\delta(\sigma_1, \sigma_2) = 1$ and $\delta(\sigma_2, \sigma_3) = 2$. So, σ_1 and σ_3 differ in two positions, 1 and 2. We have $\sigma_1\ R_{\varphi 7}\ \sigma_2$ and $\sigma_2\ R_{\varphi 6}\ \sigma_3$. Then there exists an IFS from σ_1 to σ_3, and so $\sigma_1 >_\Phi \sigma_3$.

As we see, a temporal conditional preference theory Φ is a compact way of expressing preference between sequences of objects: we can reason with any theory Φ the user gives us, provided this theory is *consistent*. More precisely:

Definition 13 (Consistency) Let Φ be a temporal preference theory. We say that Φ is *consistent* iff $>_\Phi$ is irreflexive, that is, $>_\Phi$ is a partial order over $Seq_n(\mathcal{O})$, for all $n > 0$ (remind that, by definition, $>_\Phi$ is transitive; and that transitivity and irreflexivity imply anti-symmetry).

4.3 Consistency test

The main purpose of this section is to give necessary and sufficient conditions for a temporal conditional preference theory Φ to be consistent. In this paper, we only give necessary and sufficient conditions when tcp-rules in Φ use only conjunctions of pure past and pure present formulae of STL, i.e. for all $\varphi \in \Phi$, $F_\varphi = F^-_\varphi \wedge F^0_\varphi$. In the following, we denote by **TPref*** the set of all tcp-rules of this form.

A Method for Testing Consistency. We will show (Theorem 6) that testing the consistency of a temporal conditional preference theory Φ reduces to test the consistency of a number $l(\Phi)$ of conditional preference theories over objects. Before proving this result, we need to introduce some notation first.

Let $\sigma = \langle o_1, \ldots, o_n\rangle$ be a sequence in $Seq_n(\mathcal{O})$ and o_{n+1} be an object in $\mathcal{O}$. In the following, we denote by *rlo* (for Remove Last Object) and *add* the operators defined by: $rlo(\sigma) = \langle o_1, \ldots, o_{n-1}\rangle$ and $add(\sigma, o_{n+1}) = \langle o_1, \ldots, o_n, o_{n+1}\rangle$. Let φ be a tcp-rule where $F_\varphi = F^-_\varphi \wedge F^0_\varphi \wedge F^+_\varphi$. We denote by φ^0 the cp-rule defined by: $\varphi^0 : F^0_\varphi \rightarrow (X_\varphi = x_\varphi) > (X_\varphi = x_\varphi)$. Given a tcp-theory Φ and a sequence $\sigma \in Seq(\mathcal{O})$, we define for every integer $j \in \{1, \ldots, |\sigma|\}$ the cp-theory $\Gamma_j(\Phi, \sigma)$ as follows:

$$\Gamma_j(\Phi,\sigma) = \{\varphi^0 \mid \varphi \in \Phi \wedge (\sigma, j) \models F^-_\varphi \wedge F^+_\varphi\}.$$

Intuitively, $\Gamma_j(\Phi, \sigma)$ is the set of the present components of the tcp-rules conditions whose past and future components are satisfied by σ at position *j*. Note that if σ and σ′ are two sequences in $Seq_n(\mathcal{O})$ such that *rlo*(σ) = *rlo*(σ′) then $\Gamma_n(\Phi,\sigma) = \Gamma_n(\Phi,\sigma')$. The following lemma gives a necessary condition for two sequences σ and σ′ satisfy $\sigma >_\Phi \sigma'$, where Φ is a theory in **TPref*** (without future components).

Lemma 2 Let Φ be a tcp-theory such that for every $\varphi \in \Phi$, $\varphi \in$ **TPref***. For every pair of sequences $\sigma = \langle o_1, \ldots, o_{n+1}\rangle$ and $\sigma' = \langle o'_1, \ldots, o'_{n+1}\rangle$ in $Seq_{n+1}(\mathcal{O})$ with $n > 0$, if $\sigma >_\Phi \sigma'$, then $rlo(\sigma) >_\Phi rlo(\sigma')$, or $rlo(\sigma) = rlo(\sigma')$ and $o_{n+1} >_\Gamma o'_{n+1}$ where $\Gamma = \Gamma_{n+1}(\Phi,\sigma) = \Gamma_{n+1}(\Phi,\sigma')$.

Proof. Let $\sigma = \langle o_1, \ldots, o_{n+1}\rangle$ and $\sigma' = \langle o'_1, \ldots, o'_{n+1}\rangle$ be two sequences such that $\sigma >_\Phi \sigma'$. If $\sigma >_\Phi \sigma$, it means that there exists an IFS from σ to σ′ w.r.t. Φ. Thus, there exists a set of sequences $\{\tau_1, \ldots, \tau_{p+1}\}$ in $Seq_{n+1}(\mathcal{O})$ and a set of tcp-rules $\{\varphi_1, \ldots, \varphi_p\}$ such that $\tau_1 = \sigma$, $\tau_{p+1} = \sigma'$ and for every $k \in \{1, \ldots, p\}$, $\tau_k R_{\varphi k} \tau_{k+1}$. For every $k \in \{1, \ldots, p + 1\}$, let τ'_k be the sequence in $Seq_n(\mathcal{O})$ defined by $\tau'_k = rlo(\tau_k)$. It can be easily seen that for every $k \in \{1, \ldots, p\}$, we have:

- $\tau'_k = \tau'_{k+1}$ if $\tau_k[n + 1] \neq \tau_{k+1}[n + 1]$, or
- $\tau'_k \neq \tau'_{k+1}$ if $\tau_k[n+1] = \tau_{k+1}[n+1]$. In that case, we have $j = \delta(\tau_k, \tau_{k+1}) = \delta(\tau'_k, \tau'_{k+1}) < n+ 1$. Therefore, since $\varphi \in$ **TPref***, $(\tau_k, j) \models F_{\varphi k}$ and $(\tau_{k+1}, j) \models F_{\varphi k}$ implies that $(\tau'_k, j) \models F_{\varphi k}$ and $(\tau'_{k+1}, j) \models F_{\varphi k}$. Since $\tau_k R_{\varphi k} \tau_{k+1}$, it follows that we also have $\tau'_k R_{\varphi k} \tau'_{k+1}$.

We now have to distinguish two cases:

1. Assume that there exists an integer $k \in \{1, \ldots, p\}$ such that $\tau'_k \neq \tau'_{k+1}$. In that case, since $\tau'_1 = rlo(\sigma), \tau'_{p+1} = rlo(\sigma')$, we have shown that there exists an IFS from σ to σ′ w.r.t. Φ. It shows that $rlo(\sigma) >_\Phi rlo(\sigma')$.
2. Assume now that for every $k \in \{1, \ldots, p\}$, we have $\tau'_k = \tau'_{k+1}$. It means that for every $k \in \{1, \ldots, p+1\}$, $rlo(\tau_k) = rlo(\sigma) = rlo(\sigma')$. Moreover, since $\tau_k R_{\varphi k} \tau_{k+1}$ and $\delta(\tau_k, \tau_{k+1}) = n+1$, we have $(\tau_k, n + 1) \models F_{\varphi k}$. It follows that $(\tau_k, n + 1) \models F^-_{\varphi k}$. Thus, since $rlo(\tau_k) = rlo(\sigma)$, we have $(\sigma, n + 1) \models F^-_{\varphi k}$ and $\varphi^0_k \in \Gamma_{n+1}(\Phi,\sigma)$. Now, it is easy to see that $(\tau_1[n + 1] = o_{n+1}) >_\Gamma (\tau_{p+1}[n + 1] = o'_{n+1})$ where $\Gamma = \Gamma_{n+1}(\Phi,\sigma)$, which completes the proof of Proposition 2. □

We now are ready to state the main result of this section. Its proof uses Lemma 2.

Theorem 6 Let Φ be a set of tcp-rules such that for every $\varphi \in \Phi$, $\varphi \in$ **TPref***. Φ is *consistent* iff for every sequence σ of length $k > 0$, $\Gamma_k(\Phi,\sigma)$ is consistent.

Proof. In order to prove that Φ is consistent, we have to show that $>_\Phi$ is irreflexive. First, we show that if for every sequence σ of length $k > 0$, $\Gamma_k(\Phi,\sigma)$ is consistent, then the relation $>_\Phi$ is irreflexive. We show this property by induction on the length of sequences.

Let $\sigma = \langle o\rangle$ be a sequence of length n = 1. If $\sigma >_\Phi \sigma$, it means that there exists an IFS from σ to σ, i.e. a set of sequences $\{\langle o_1\rangle, \ldots, \langle o_{p+1}\rangle\}$ and a set of tcp-rules $\{\varphi_1, \ldots, \varphi_p\}$ such that $o = o_1 = o_{p+1}$

and for every $k \in \{1, \ldots, p\}$, $\langle o_k \rangle R_{\varphi k} \langle o_{k+1} \rangle$. By Definition 11, for every $k \in \{1, \ldots, p\}$, we have $(\langle o_k \rangle, 1) \models F_{\varphi k}$. Thus, we have $(\langle o_k \rangle, 1) \models F^{-}_{\varphi k}$ and $\varphi^{0}_{k} \in \Gamma_1(\Phi, \langle o_k \rangle) = \Gamma_1(\Phi, \langle o \rangle)$ because $rlo(\langle o_k \rangle) = rlo(\langle o \rangle)$. Finally, for every $k \in \{1, \ldots, p\}$, we have $o_k \; R_{\varphi^0_k} \; o_{k+1}$. It shows that $o >_\Gamma o$ with $\Gamma = \Gamma_1(\Phi, \langle o \rangle)$, which contradicts the hypothesis that $\Gamma_k(\Phi, \sigma_k)$ is consistent for every sequence σ_k of length $k > 0$ and proves that $>_\Phi$ is irreflexive on $Seq_1(\mathcal{O})$.

Assuming that $>_\Phi$ is irreflexive on $Seq_n(\mathcal{O})$, we now have to prove that $>_\Phi$ is a irreflexive on $Seq_{n+1}(\mathrm{O})$. Suppose that there exists a sequence $\sigma_{n+1} = \langle o_1, \ldots, o_{n+1} \rangle$ in $Seq_{n+1}(\mathcal{O})$ such that $\sigma_{n+1} >_\Phi \sigma_{n+1}$. Using Proposition 2, we have to distinguish two cases:

1. If $rlo(\sigma_{n+1}) >_\Phi rlo(\sigma_{n+1})$, it shows that $>_\Phi$ is not irreflexive on $Seq_n(\mathcal{O})$, which contradicts the hypothesis.
2. If $rlo(\sigma_{n+1}) = rlo(\sigma_{n+1})$, then we have $o_{n+1} >_\Gamma o_{n+1}$ where $\Gamma = \Gamma_{n+1}(\Phi, \sigma_{n+1})$. It shows that $>_\Gamma$ is not irreflexive, which contradicts the hypothesis that $\Gamma_k(\Phi, \sigma_k)$ is consistent for every sequence σ_k of length $k > 0$.

So, we have proved by induction that if for every sequence σ of length $k > 0$, $\Gamma_k(\Phi, \sigma)$ is consistent, then $>_\Phi$ is a SPO on $Seq_n(\mathcal{O})$ for every integer $n \geq 1$.

We now prove that if $>_\Phi$ is a SPO, then $\Gamma_k(\Phi, \sigma)$ is consistent for every sequence σ of length $k > 0$. Assume that there exists a sequence σ of length k such that $\Gamma = \Gamma_k(\Phi, \sigma)$ is not consistent. It means that $>_\Phi$ is not irreflexive, i.e. that there exists an object o_{k+1} such that $o_{k+1} >_\Gamma o_{k+1}$. Let σ_{k+1} be the sequence defined by $\sigma_{k+1} = add(\sigma, o_{k+1})$. It is easy to see that $\sigma_{k+1} >_\Phi \sigma_{k+1}$, which contradicts the fact that $>_\Phi$ is irreflexive and completes the proof. □

Theorem 6 is not true when the tcp-rules in Φ contain past and future components, as we show in the following example:

Example 14 Let $\Phi = \{ \varphi_1, \varphi_2, \varphi_3, \varphi_4 \}$ be the set of tcp-rules defined by:

- $\varphi_1 : Next(G = d) \rightarrow (D = n) > (D = w)$
- $\varphi'_1 : Next(G = c) \rightarrow (D = w) > (D = n)$
- $\varphi_2 : Prev(D = n) \rightarrow (G = c) > (G = d)$
- $\varphi'_2 : Prev(D = w) \rightarrow (G = d) > (G = c)$

Since the STL formulae $\varphi_1 \wedge \varphi'_1$ and $\varphi_2 \wedge \varphi'_2$ cannot be satisfied, it is easy to see that for every sequence σ of length k, $\Gamma = \Gamma_k(\Phi, \sigma)$ is locally consistent. Moreover, for every sequence σ of length k, $G(\Gamma_k(\Phi, \sigma)) = (\{G, D\}, 3)$ is acyclic. Therefore, for every sequence σ of length k, $\Gamma = \Gamma_k(\Phi, \sigma)$ is consistent. We now show that Φ is not consistent, which does not contradict Theorem 6 since Φ uses past and future STL formulae in the conditions of the tcp-rules. Given the objects $o_1 = (c, n)$, $o_2 = (d, w)$, $o'_1 = (c, w)$ and $o'_2 = (c, w)$, consider the sequences $\sigma_1 = \langle o_1, o_2 \rangle$, $\sigma_2 = \langle o'_1, o_2 \rangle$, $\sigma_3 = \langle o'_1, o'_2 \rangle$ and $\sigma_4 = \langle o_1, o'_2 \rangle$. It is easy to verify the following:

- $\sigma_1 \; R_{\varphi 1} \; \sigma_2$ since $(\sigma_1, 1) \models Next(G = d)$, $(\sigma_2, 1) \models Next(G = d)$, $o_1[D] = n$ and $o'_1[D] = w$.
- $\sigma_2 \; R_{\varphi' 2} \; \sigma_3$ since $(\sigma_2, 2) \models Prev(D = w)$, $(\sigma_3, 2) \models Prev(D = w)$, $o_2[G] = d$ and $o'_2[G] = c$.

- $\sigma_3\ R_{\varphi'1}\ \sigma_4$ since $(\sigma_3, 1)\ |= Next(G = c)$, $(\sigma_4, 1)\ |= Next(G = c)$, $o'_1[D] = w$ and $o_1[D] = n$.
- $\sigma_4\ R_{\varphi 2}\ \sigma_1$ since $(\sigma_4, 2)\ |= Prev(D = n)$, $(\sigma_1, 2)\ |= Prev(D = n)$, $o'_2[G] = c$ and $o_2[G] = d$.

Thus, we have $\sigma_1 >_\Phi \sigma_1$, which shows that $>_\Phi$ is not consistent since it is not irreflexive.

Complexity Issues. In practice, the condition provided by Theorem 6 to test consistency is unfeasible, since it involves testing consistency of the non-temporal theories $\Gamma_k(\Phi, \sigma)$ for every sequence σ of length k. Fortunately, for some fragments of STL, we can find a very satisfatory bound for the size of the sequences σ which must be considered in the tests.

Theorem 7 Let $L(\blacklozenge, \lozenge)$ be the fragment of *STL* whose formulae satisfy the following conditions: (1) negation appear only in front of basic propositions; (2) the only temporal operators are $\blacklozenge$ and $\lozenge$. Let $F \in L(\blacklozenge,\lozenge)$ be satisfiable. Then there exists a sequence σ such that $|\sigma| \leq length(F)$ and such that σ satisfies F. The *length* of a formula F (denoted by *length(F)*) is the number of symbols appearing in F.

Proof. Let $\sigma = \langle o_1, ..., o_k\rangle$ be a sequence. A subsequence of σ is a sequence $\tau = \langle u_1, ..., u_m\rangle$ such that for all $i \in \{1, ...,m\}$ there exists $j_i \in \{1, ..., k\}$ such that $o_{i_j} = u_i$. We denote the fact that τ is a subsequence of σ by $\tau \preceq \sigma$.

Let $F \in STL$ and $\sigma = (o_1, ..., o_k)$ such that $(\sigma, i)\ |= F$. We will prove that there exists a subsequence $\tau \preceq \sigma$ such that (1) τ contains the object o_i, (2) $|\tau| \leq$ length(F) and (3) for all sequence σ' such that $\tau \preceq \sigma' \preceq \sigma$ we have $(\sigma', i)\ |= F$. Particularly, we can affirm that $(\tau, i)\ |= F$, since $\tau \preceq \tau \preceq \sigma$.

The proof is by induction on the structure of F.

- If F is atomic and $(\sigma, i)\ |= F$, then let $\tau =< o_i >$. We have that $\tau \preceq \sigma$, $|\tau| = 1 =$ length(F) and for all σ' such that $\tau \preceq \sigma' \preceq \sigma$ we have $(\sigma', i)\ |= F$, since σ' contains the object σ_i.
- If F is $\neg F_1$, where F_1 is an atomic formula, the proof is similar: we take $\tau =< o_i >$. In this case, $|\tau| = 1 <$ length(F) = 2.
- The cases where $F = G \vee H$ and $F = G \wedge H$ do not present any difficulty and we omit it here.
- If $F = \lozenge F_1$ and $(\sigma, i)\ |= F$. Then there exists $j > i$ such that $(\sigma, j)\ |= F_1$. By the induction hypothesis, we can affirm that there exist a subsequence $\tau \preceq \sigma$, such that τ contains the object o_j, $|\tau| \leq$ length(F_1), and for all σ' verifying $\tau \preceq \sigma' \preceq \sigma$ we have $(\sigma', j)\ |= F_1$. If $o_i \in \tau$ we define $\tau' = \tau$. Otherwise, τ' is obtained from τ by inserting the object o_i in it (in the same order as it appears in σ). Then, it is clear that $(\tau', i)\ |= \lozenge F_1$, since $(\tau', j)\ |= F_1$. Moreover, $\tau' \leq \tau + 1 =$ length(F). The proof is similar for $F = \blacklozenge F_1$.
- If $F =$ **Next** F_1 and $(\sigma, i)\ |= F$. Then $i < |\sigma|$ and $(\sigma, i+1)\ |= F_1$. By the induction hypothesis, we can affirm that there exist a subsequence $\tau \preceq \sigma$, such that τ contains the object o_{i+1}, $|\tau| \leq$ length(F_1), and for all σ' verifying $\tau \preceq \sigma' \preceq \sigma$ we have $(\sigma', i + 1)\ |= F_1$. If $o_i \in \tau$ we define $\tau' = \tau$. Otherwise, τ' is obtained from τ by inserting the object o_{i+1} in it (following the object o_i). Then, it is clear that $(\tau', i)\ |=$ **Next** F_1 since $(\tau', i + 1)\ |= F_1$. Moreover $\tau' \leq \tau + 1 =$ length(F). The proof is similar for $F =$ **Prev** F_1.

According to (Sistla & Clarke, 1985), the satifiability problem for STL is NP-complete for $L(\blacklozenge, \lozenge)$ and PSPACE-complete for the logic STL.

Proposition 2 Let Φ be a set of tcp-rules in **TPref*** such that the temporal conditions are formula of $L(\blacklozenge, \lozenge)$. Then Φ is consistent iff $\Gamma_k(\Phi, \sigma)$ is consistent for every sequence σ of length $\leq$ length(Φ), where length(Φ) = max{ length(φ) | $\varphi \in \Phi$ }.

Notice that if we place ourselves in a context where the universe of sequences is finite, then there is no need to restrict the conditions of the tcp-rules to be formulae in $L(\blacklozenge, \lozenge)$. As the whole universe of sequences is contained in $Seq_n(\mathcal{O})$, for some $n > 0$, then in order to test consistency of a tcp theory Φ, it suffices to test the consistency of the non-temporal theories $\Gamma_k(\Phi, \sigma)$ for each sequence σ of size $k \leq n$. In such cases, there is no relation between the size of the temporal conditions and the maximal size of the sequences to be tested. A situation where restricting the type of the formulas considered in the conditions of tcp-rules is worthwhile is when working in a context where the universe of sequences is *potentially* infinite, that is, the maximal size of the sequences evolves with time (for instance, in a temporal database context).

The following proposition relates the result stated in Theorem 4 and the result given in Proposition 2.

Proposition 3 Let Φ be a tcp-theory and $l(\Phi)$ be the length of Φ. Let us suppose that $G(\Phi)$ is acyclic and for every $\varphi \in \Phi$, $\varphi \in$ **TPref***. Then, Φ is consistent iff for every sequence σ of length $k \leq \mathrm{l}(\Phi)$, $\Gamma_k(\Phi, \sigma)$ is locally consistent. Besides, if all variables in V are binary, then consistency of Φ can be determined in time proportional to $|\Gamma|^2 \times |V| \times 2^{l(\Phi)}$.

4.4 Finding optimal sequences

In this section, given a tcp-theory Φ, we show how to determine the optimal sequences in $Seq_n(\mathcal{O})$, i.e. the maximal sequences in $Seq_n(\mathcal{O})$ with respect to $>_\Phi$ that satisfy some set of simple temporal constraints. Our approach is *incremental*, meaning that for every integer n, we show how to compute the optimal sequences in $Seq_{n+1}(\mathcal{O})$ from the set of optimal sequences in $Seq_n(\mathcal{O})$.

First, we specify the set of temporal constraints that we consider. For every integer $k > 0$, let $\mathbf{At}_k(X = a)$ be the temporal formula defined as $(\mathbf{Is}_k \wedge (X = a)) \vee \blacklozenge(\mathbf{Is}_k \wedge (X = a))$ where $\mathbf{Is}_k$ is defined by induction on k as: $\mathbf{Is}_1 = \mathbf{First}$ and $\mathbf{Is}_{i+1} = \mathbf{Prev}\ \mathbf{Is}_i$ for every integer $i > 0$. We can see that for every sequence $\sigma \in Seq(\mathcal{O})$, $\sigma \models \mathbf{At}_k(X = a)$ iff $(\sigma, k) \models (X = a)$. Intuitively, the formula $\mathbf{At}_k(X = a)$ means that in a sequence of objects, the object at state k has value a for the variable X.

In the following, we denote by **AtState** the set of formulas of the form $\mathbf{At}_k(X = a)$. Given a subset $\mathcal{C}$ of **AtState**, we say that $\mathcal{C}$ is *consistent* if there exists a sequence $\sigma \in Seq(\mathcal{O})$ such that for every $F \in \mathcal{C}$, σ satisfies F (denoted by $\sigma \models \mathcal{C}$). We can easily see that $\mathcal{C}$ is consistent iff for every pair (F, F') in $\mathcal{C}^2$ where $F = \mathbf{At}_k(X = a)$ and $F' = \mathbf{At}_{k'}(X' = a')$, if $k = k'$ and $X = X'$, then we have $a = a'$. Given a consistent subset $\mathcal{C}$ of **AtState** and an integer k, we denote by: (1) $Attr_k(\mathcal{C})$ the set of attributes $X \in V$ such that there exists a formula $\mathbf{At}_k(X = a)$ in $\mathcal{C}$. (1) $\mathcal{C}_k$ the subset of $\mathcal{C}$ defined by: $\mathcal{C}_k = \{\mathbf{At}_i(X = a) \in \mathcal{C} \mid (i \leq k)\}$. (2) $Tuple_k(\mathcal{C})$ the set of present STL formulae defined by: $Tuple_k(\mathcal{C}) = \{(X = a) \mid At_k(X = a) \in \mathcal{C}\}$.

Example 15 Let $\mathcal{C} = \{\mathbf{At}_1(G = c), \mathbf{At}_2(G = d), \mathbf{At}_2\,(D = n)\}$. It is easy to see that $\mathcal{C}$ is consistent. Moreover, $Tuple_1(\mathcal{C}) = \{(G = c)\}$ and $Tuple_2(\mathcal{C}) = \{(G = d), (D = n)\}$. Finally, we have $\mathcal{C}_1 = \{\mathbf{At}_1(G = c)\}$ and $\mathcal{C}_2 = \mathcal{C}$.

Let $\mathcal{C}$ be a consistent subset of **AtState**. Given a consistent **TPref** theory Φ, we now show how to compute for every integer n, the subset $\mathcal{S}_n(\Phi, \mathcal{C})$ of $Seq_n(\mathcal{O})$ defined by: $\mathcal{S}_n(\Phi, \mathcal{C}) = max_{>_\Phi}\{\sigma \in Seq_n(\mathcal{O}) \mid \sigma \models \mathcal{C}_n\}$. The set $\mathcal{S}_n(\Phi, \mathcal{C})$ contains the optimal sequences in $Seq_n(\mathcal{O})$, i.e. the maximal sequences in $Seq_n(\mathcal{O})$ w.r.t. $>_\Phi$ that satisfy the constraints in $\mathcal{C}_n$.

Let σ_k be a sequence of length k. In the following, given the cp-theory $\Gamma = \Gamma(\Phi, \sigma_k)$ and the set of present STL formula $\mathcal{T} = Tuple_k(\mathcal{C})$, we denote by $BestObjs(\Gamma, \mathcal{T})$ the set of optimal objects in $\mathcal{O}$ that satisfy $\mathcal{T}$, i.e.

$BestObjs(\Gamma, \mathcal{T}) = max_{>_\Gamma}\{o \in \mathcal{O} \mid o \models \mathcal{T}\}$. It is shown in (Wilson, 2004) how to compute this set of optimal objects.

Finally, given a tcp-theory Φ such that for every tcp-rule $\varphi \in \Phi$, $\varphi \in$ **TPref***. We can notice that for every sequence σ and σ' of length $n + 1$, if $rlo(\sigma) = rlo(\sigma')$, then $\Gamma_{n+1}(\Phi,\sigma) = \Gamma_{n+1}(\Phi,\sigma')$. Therefore, for every sequence σ of length n, we introduce the following notation: $\Gamma^*(\Phi, \sigma) = \Gamma_n(\Phi, add(\sigma, o))$ where o is any object in $\mathcal{O}$.

We now state the following theorem that shows how to compute $\mathcal{S}_{n+1}(\Phi, \mathcal{C})$ from $\mathcal{S}_n(\Phi, \mathcal{C})$.

Theorem 8 Let Φ be a consistent tcp-theory such that for every tcp-rule $\varphi \in \Phi$, $\varphi \in$ **TPref***. Let $\mathcal{C}$ be a consistent subset of *AtState*. For every sequence $\sigma = \langle o_1, \ldots, o_{n+1}\rangle \in Seq_{n+1}(\mathcal{O})$, σ is in $\mathcal{S}_{n+1}(\Phi, \mathcal{C})$ iff $rlo(\sigma) \in \mathcal{S}_n(\Phi, \mathcal{C})$ and $o_{n+1} \in BestObjs(\Gamma, \mathcal{T})$ where $\Gamma = \Gamma^*(\Phi, rlo(\sigma))$ and $\mathcal{T} = Tuple_{n+1}(\mathcal{C})$.

Proof Assume that $\sigma = \langle o_1, \ldots, o_{n+1}\rangle$ is in $\mathcal{S}_{n+1}(\Phi, \mathcal{C})$. Let $\sigma_n = rlo(\sigma)$. If $\sigma_n \notin \mathcal{S}_n(\Phi, \mathcal{C})$, it means that there exists a sequence $\sigma'_n \in \mathcal{S}_n(\Phi, \mathcal{C})$ such that $\sigma'_n \models \mathcal{C}$ and $\sigma'_n >_\Phi \sigma_n$. Since $\sigma'_n >_\Phi \sigma_n$, there exists an IFS from σ'_n to σ_n w.r.t. Φ, i.e. there exist a set of sequences $\{\tau_1, \ldots, \tau_{p+1}\}$ and a set of tcp-rules $\{\varphi_1, \ldots, \varphi_p\}$ such that $\tau_1 = \sigma'_n$, $\tau_{p+1} = \sigma_n$ and for every $k \in \{1, \ldots, p\}$, $\tau_k\, R_{\varphi k}\, \tau_{k+1}$. For every $k \in \{1, \ldots, p + 1\}$, let $\tau'_k = add(\tau_k, o_{n+1})$. Since for every tcp-rule $F_{\varphi k} = F^{-}_{\varphi k} \wedge F^{0}_{\varphi k}$ and $\tau'_k\ R_{\varphi k}\, \tau'_{k+1}$, we also have $\tau'_k\, R_{\varphi k}\, \tau'_{k+1}$. Thus, since $\tau'_{p+1} = \sigma$, there exists an IFS from τ'_1 to σ w.r.t. Φ, i.e. $\tau'_1 >_\Phi \sigma$. Moreover, we can easily see that $\tau'_1 \models \mathcal{C}$. Thus, we have $\tau'_1 >_\Phi \sigma$ and $\tau'_1 \models \mathcal{C}$ which contradicts the fact that $\sigma \in \mathcal{S}_{n+1}(\Phi, \mathcal{C})$.

On the other hand, assume that $o_{n+1} \notin BestObjs(\Gamma, \mathcal{T})$. It means that there exists an object $o'_{n+1} \in BestObjs(\Gamma, \mathcal{T})$ such that $o'_{n+1} \models \mathcal{T}$ and $o'_{n+1} >_\Gamma o_{n+1}$. Let $\sigma' = add(\sigma_n, o'_{n+1})$. We can easily show that $\sigma' >_\Phi \sigma$ and $\sigma' \models \mathcal{C}$ where $\sigma' = add(\sigma_n, o'_{n+1})$ which contradicts the hypothesis that σ is in $\mathcal{S}_{k+1}(\Phi, \mathcal{C})$. Thus, we have proved that if σ is in $\mathcal{S}_{n+1}(\Phi, \mathcal{C})$, then $rlo(\sigma) \in \mathcal{S}_n(\Phi, \mathcal{C})$ and $o_{n+1} \in BestObjs(\Gamma, \mathcal{T})$.

Conversely, assume that $\sigma_n = rlo(\sigma) \in \mathcal{S}_n(\Phi, \mathcal{C})$ and $o_{n+1} \in BestObjs(\Gamma, \mathcal{T})$. If σ is not in $\mathcal{S}_{n+1}(\Phi, \mathcal{C})$, then there exists a sequence $\sigma' = \langle o'_1, \ldots, o'_{n+1}\rangle \in \mathcal{S}_{n+1}(\Phi, \mathcal{C})$ such that $\sigma' >_\Phi \sigma$ and $\sigma' \models \mathcal{C}$. Using Proposition 2, we now distinguish two cases:

- If $rlo(\sigma') >_\Phi rlo(\sigma)$, then it is easy to see that we also have $rlo(\sigma') \models \mathcal{C}_n$. Thus $rlo(\sigma') >_\Phi \sigma_n$ and $rlo(\sigma') \models \mathcal{C}_n$, which contradicts the fact that $\sigma_n \in \mathcal{S}_n(\Phi, \mathcal{C})$.
- If $rlo(\sigma') = rlo(\sigma')$, then $o'_{n+1} >_\Gamma o_{n+1}$ with $\Gamma = \Gamma_{n+1}(\Phi, \sigma_{n+1})$. Moreover, we can easily see that $o'_{n+1} \models \mathcal{T}$ since $\sigma' \models \mathcal{C}$. Thus, we have $o'_{n+1} >_\Gamma o_{n+1}$ and $o'_{n+1} \models \mathcal{T}$, which contradicts the fact that $o_{n+1} \in BestObjs(\Gamma, \mathcal{T})$.

Thus, we show that if $rlo(\sigma) \in \mathcal{S}_n(\Phi, \mathcal{C})$ and $o_{n+1} \in BestObjs(\Gamma, \mathcal{T})$, then σ is in $\mathcal{S}_{n+1}(\Phi, \mathcal{C})$, which completes the proof.

Using Theorem 8, it is easy to see that for every consistent tcp-theory and every consistent subset $\mathcal{C}$ of **AtState**, we have $\mathcal{S}_n(\Phi, \mathcal{C}) = BestSeqs(\langle\rangle, \Phi, \mathcal{C}, n)$ where $\langle\rangle$ represents the empty sequence and *BestSeqs* is the algorithm presented in Figure 5.

```
Function BestSeqs(σ, Φ, C, n)
-----------------------------------------------------------
Input:    A sequence $\sigma$ of length $k$ such that $\sigma \models \mathcal{C}_k$
          A consistent tcp-theory $\Phi$
          A consistent subset $\mathcal{C}$ of AtState
          An integer $n > k$
Output:   The set of optimal sequences $\mathcal{S}$ of size $n$ and
          prefix $\sigma$ that satisfy $\mathcal{C}_n$
-----------------------------------------------------------
1.   Let $\mathcal{S} = \emptyset$ and $k = |\sigma|$
2.   If $(k < n)$
3.       Let $\Gamma = \Gamma^*(\Phi, \sigma)$
4.       Let $\mathcal{T} = Tuple_{k+1}(\mathcal{C})$
5.       For every $o_{k+1} \in BestObjs(\Gamma, \mathcal{T})$ do
6.           Let $\sigma' = add(\sigma, o_{k+1})$
7.           $\mathcal{S} = \mathcal{S} \cup BestSeqs(\sigma', \Phi, \mathcal{C}, n)$
8.       End for
9.       Return $\mathcal{S}$
10.  Else
11.      Return $\mathcal{S}$
```

Fig. 5. Computation of Optimal Sequences

Example 16 Let $\Phi = \{\varphi_1, \ldots, \varphi_6\}$ be the tcp-theory presented in our Running Example. Let $\mathcal{C} = \{At_1(G = c), At_3(D = w)\}$. We show in this example how the set $\mathcal{S}_3(\Phi, \mathcal{C})$ is computed using the algorithm presented Figure 5. Initially, we compute $\mathcal{S} = BestSeqs(\langle\rangle, \Phi, \mathcal{C}, 3)$. First, we have $\Gamma_1 = \Gamma^*(\Phi, \langle\rangle) = \{\varphi_1, \varphi_2\}$ since $F_{\varphi 1}$ and $F_{\varphi 2}$ are STL formulae in *Present*. Moreover, we have $\mathcal{T}_1 = Tuple_1(\mathcal{C}) = \{(G = c)\}$. Thus, we compute $BestObjs(\Gamma_1, \mathcal{T}_1) = \{o_1\}$ where $o_1 = (G = c, D = w)$. Then, we build the sequence $\sigma'_1 = add(\langle\rangle, o'_1) = \langle o_1\rangle$ and compute $\mathcal{S} = BestSeqs(\sigma'_1, \Phi, \mathcal{C}, 3)$.

Computing $\mathcal{S} = BestSeqs(\sigma'_1, \Phi, \mathcal{C}, 3)$, we successively obtain $\Gamma_2 = \Gamma^*(\Phi, \langle o_1 \rangle) = \{\varphi_1, \varphi_2, \varphi^0_4\}$, $\mathcal{T}_2 = Tuple_2(\mathcal{C}) = 3$ and $BestObjs(\Gamma_2, \mathcal{T}_2) = \{o_2\}$ where $o_2 = (G = d, D = n)$. Thus, we build the sequence $\sigma'_2 = add(\langle o_1 \rangle, o_2) = \langle o_1, o_2 \rangle$ and compute $\mathcal{S} = BestSeqs(\sigma'_2, \Phi, \mathcal{C}, 3)$.

Then, computing $\mathcal{S} = BestSeqs(\sigma'_2, \Phi, \mathcal{C}, 3)$, we successively obtain $\Gamma_3 = \Gamma^*(\Phi, \langle o_1, o_2 \rangle) = \{\varphi_1, \varphi_2, \varphi^0_6\}$, $\mathcal{T}_3 = Tuple_3(\mathcal{C}) = \{(D = w)\}$ and $BestObjs(\Gamma_3, \mathcal{T}_3) = \{o_3\}$ where $o_3 = (G = d, D = w)$. Thus, we build the sequence $\sigma'_3 = add(\langle o_1, o_2 \rangle, o_3) = \langle o_1, o_2, o_3 \rangle$ and compute $\mathcal{S} = BestSeqs(\sigma'_3, \Phi, \mathcal{C}, 3)$.

Since $|\sigma'_3| = 3$, we finally obtain $\mathcal{S} = BestSeqs(\langle\rangle, \Phi, \mathcal{C}, 3) = \{\langle o_1, o_2, o_3 \rangle\}$. Note that in this example, we only obtain one optimal sequence. In general, we can obtain a set of optimal sequences since $>_\Phi$ is a *partial* order.

5. Conclusion and further research

In this chapter, we have presented several approaches for treating preferences over objects, sets of objects and sequences of objects. The main contribution is centered in Section 4 which presents a method for preference elicitation and reasoning over sequence of objects. An algorithm for finding the most preferred sequences satisfying a set of temporal constraints is introduced. A lot of work has to be done to improve our approach. (1) Concerning the algorithm for finding the best sequences, we intend to generalize our method in order to treat more general temporal constraints. (2) Concerning the expressivity power of our preference language: we note that in TPref the temporal aspect is related only to the rule conditions, that is, only to the left side of the preference rules. We are not able, for the time being, to treat preference statements such as *I prefer "this" before "that"*. (3) Concerning the consistency test: we must investigate methods to ensure consistency when the temporal conditions involve both past and future operators. (4) Concerning dominance queries: we have to investigate efficient methods to determine, given two sequences, which is the preferred one. That implies investigating efficient methods to decide, given two sequence, if there exists a IFS between them. (5) Finally, concerning a database context, the work proposed in this paper is a first step towards incorporating a formalism for *reasoning* with preferences over sequences of objects into a temporal relational query language, and so, building a bridge between the two disciplines (AI and Temporal Databases).

6. References

Bacchus, F.; Boutilier, C. & Grove, A. (1996). Rewarding behaviors, *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp 1160–1167, Portland, Oregon, USA, 1996. AAAI Press / The MIT Press.

Bacchus, F. & Grove, A. (1995). Graphical models for preference and utility. *In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 3–10, Montreal.

Bacchus, F. & Grove, A. (1996). Utility independence in qualitative decision theory, Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning, pp. 542–552, Cambridge.

Bienvenu, M.; Fritz, C. & McIlraith, S. A. (2006). Planning with qualitative temporal preferences. *KR*, 134–144, 2006.

Boutilier, C.; Brafman, R.; Geib, C. & Poole, D. (1997). A constraint-based approach to preference elicitation and decision making, *AAAI Spring Symposium on Qualitative Decision Theory*, Stanford, 1997.

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

Brafman, R.; Domshlak, C. & Shimony, S.E. (2006a). On graphical modeling of preference and importance. *Journal of Artificial Intelligence Research*, 25:389–424, 2006.

Brafman, R.; Domshlak, C. & Shimony, S.E. (2006b). Preferences over sets. *AAAI*, 2006.

Chomicki, J. (2003). Preference formulas in relational queries. *ACM Transactions on Database Systems*, pp. 427–466, 2003.

de Amo, S. & Giacometti, A. (2007). Temporal Conditional Preferences over Sequences of Objects. *19th IEEE International Conference on Tools with Artificial Intelligence*, Patras, Greece, pp. 246-253, 2007.

des Jardins, M. & Wagstaff, K. (2005). Dd-pref: A language for expressing preferences over sets. *AAAI*, pp. 620–626, 2005.

Domshlak, C. & Brafman, R. (2002). CP-nets - reasoning and consistency testing, *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning*, pp. 121–132, Toulouse, France, 2002.

Doyle, J.; Shoham, Y. & Wellman, M. (1991). A logic of relative desire (preliminary report), *Proceedings of the Sixth International Symposium on Methodologies for Intelligent Systems (ISMIS 91)*, Lecture Notes in Computer Science, pp. 16–31. Springer-Verlag, 1991.

Endres, M. & Kießling, W. (2006). Transformation of tcp-net queries into preference database queries. *Proceedings of the ECAI 2006 Multidisciplinary Workshop on Advances in Preference Handling Riva del Garda*, Italy, August 2006, pp. 23–30.

Gabbay, D. M. (1989). The declarative past and imperative future: Executable temporal logic for interactive systems. *Lecture Notes in Computer Science*, Volume 398, pp 67–89.

Springer-Verlag, 1989. Hodksinson, I. & Reynolds, M. (2005). Separation – past, present, and future. *We Will Show Them! Essays in Honour of Dov Gabbay*, Volume 2, 2005.

Khatib, L.; Morris, P.; Morris, R.A. & Rossi, F. (2001). Temporal Constraint Reasoning With Preferences. *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pp. 322–327, 2001.

Kießling, W. (2002). Foundations of preferences in database systems. *Proceedings of 28th International Conference on Very Large Data Bases*, Hong Kong, China, pp. 311–322, 2002.

Kießling, W. & Köstler, G. (2002). Preference SQL - design, implementation, experiences. *Proceedings of 28th International Conference on Very Large Data Bases*, Hong Kong, China, pp. 990–1001, 2002.

Kumar, T.K.S. (2007). Fast (Incremental) Algorithms for Useful Classes of Simple Temporal Problems with Preferences. *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2007.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, 1988.

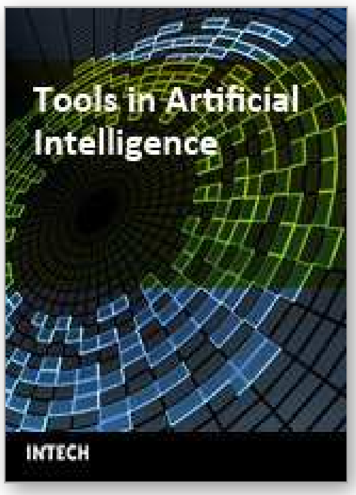
Prestwich, S. D.; Rossi, F.; Venable, K. B. & Walsh, T. (2005). Constraint-Based Preferential Optimization. *AAAI 2005*, pp. 461–466, 2005.

Prior, A. N. (1997). *Past, Present and Future*, Oxford: Clarendon Press, 1967.

Sistla, A. P. & Clarke, E.M. (1985). The complexity of propositional linear temporal logic. *Journal of the ACM*, 32(3), pp 733–749, 1985.

Son, T.C. & Pontelli, E. (2006). Planning with preferences using logic programming. *TPLP*,6(5):559–607, 2006.

Wilson, N. (2004). Extending cp-nets with stronger conditional preference statements. *AAAI*, pp. 735–741, 2004.


Tools in Artificial Intelligence
INTECH