

Monotonic and Nonmonotonic Preference Revision *

Jan Chomicki Joyce Song

University at Buffalo

{chomicki, jsong5}@cse.buffalo.edu

Abstract

We study here preference revision, considering both the *monotonic* case where the original preferences are preserved and the *nonmonotonic* case where the new preferences may override the original ones. We use a relational framework in which preferences are represented using binary relations (not necessarily finite). We identify several classes of revisions that preserve order axioms, for example the axioms of strict partial or weak orders. We consider applications of our results to preference querying in relational databases.

1 Introduction

The notion of *preference* is common in various contexts involving decision or choice. Classical utility theory [Fishburn, 1970] views preferences as *binary relations*. A similar view has recently been espoused in database research [Chomicki, 2003; Kießling, 2002; Kießling & Köstler, 2002], where preference relations are used in formulating *preference queries*. In AI, various approaches to compact specification of preferences have been explored [Boutilier *et al.*, 2004]. The semantics underlying such approaches typically relies on preference relations between worlds.

However, user preferences are rarely static [Pu, Faltings, & Torrens, 2003]. A database user may be disappointed by the result of a preference query and decide to revise the preferences in the query. In fact, a user may start with a partial or vague concept of her preferences, and subsequently refine that concept. An agent may learn more about its task domain and consequently revise its preferences. Thus, it is natural to study *preference revision*, as we do in the present paper.

Preference revision shares some of the principles, namely minimal change and primacy of new information, with classical belief revision [Gärdenfors & Rott, 1995]. However, its basic setting is different. In belief revision, propositional theories are revised with propositional formulas, yielding new theories. In preference revision, binary preference relations are revised with other preference relations, yielding new

preference relations. Preference relations are single, finitely representable (though possibly infinite) first-order structures, satisfying order axioms.

We distinguish between *monotonic* and *nonmonotonic* preference revision. In the former, the original preference relation is fully incorporated into the revised one. In the latter, the original preference relation may conflict with the revising relation, leading to the necessity of retracting some of the original preferences. We focus on two special cases: *refinement* in which both the original and the revising relation are preserved, and *overriding revision* in which the revising relation may override the original one. We adopt the notion of minimal change based on symmetric difference between sets of tuples.

The challenges are: (1) to guarantee that suitable order properties, for example the axioms of strict partial orders, are preserved by the revisions, and (2) to obtain unique revisions. Strict partial orders (and weak orders), apart from being intuitive, enjoy a number of attractive properties in the context of preference queries, as explained later in the paper. So it is desirable for revisions to preserve such orders. The uniqueness property is also important from the user's point of view, as the user typically desires to obtain a single revised preference relation. The presence of multiple revision candidates necessitates some form of aggregation of or choice among the candidates. Fortunately, in the cases studied in this paper there exist least revisions preserving the appropriate order axioms, and thus uniqueness is obtained automatically.

We adopt the preference query framework of [Chomicki, 2003] (a similar model was described in [Kießling, 2002]), in which preference relations between tuples are defined by logical formulas. [Chomicki, 2003] proposed a new relational algebra operator called *winnow* that selects from its argument relation the *most preferred tuples* according to the given preference relation.

Example 1 Consider the relation $Car(Make, Year)$ and the following preference relation \succ_{C_1} between Car tuples:

within each make, prefer a more recent car.

which can be defined as follows:

$$(m, y) \succ_{C_1} (m', y') \equiv m = m' \wedge y > y'.$$

The winnow operator ω_{C_1} returns for every make the most recent car available. Consider the instance r_1 of Car in Figure 1. The set of tuples $\omega_{C_1}(r_1)$ is shown in Figure 2.

*Research supported by NSF grant IIS-0307434.

	Make	Year
t_1	VW	2002
t_2	VW	1997
t_3	Kia	1997

Figure 1: The Car relation

	Make	Year
t_1	VW	2002
t_3	Kia	1997

Figure 2: The result of winnow

Example 2 Example 1 provides a motivation for studying preference revision. Seeing the result of the query $\omega_{C_1}(r_1)$, a user may realize that the preference relation \succ_{C_1} is not quite what she had in mind. The result of the query may contain some unexpected or unwanted tuples, for example t_3 . Thus the preference relation needs to be modified, for example by refining it with the following preference relation \succ_{C_2} :

$$(m, y) \succ_{C_2} (m', y') \equiv m = \text{"VW"} \wedge m' \neq \text{"VW"} \wedge y = y'.$$

The resulting refinement will contain both \succ_{C_1} and \succ_{C_2} . The tuple t_3 is now dominated by t_2 and will not be returned to the user.

In the terminology used in research on preference reasoning in AI [Boutilier *et al.*, 2004], a relational database instance corresponds to the set of *feasible outcomes* and the winnow operator picks the undominated (best) outcomes from this set, according to the given preferences. A preference setting can be affected by a change in preferences or a modification of the set of possible outcomes. In this research, we address the former problem; the latter one, database update, has been extensively studied in database research. Moreover, we limit ourselves to preference revisions in which new preference information is combined, perhaps nonmonotonically, with the old one. We assume that the domains of preferences do not change in revisions.

2 Basic notions

We are working in the context of the relational model of data. Relation schemas consist of finite sets of attributes. For concreteness, we consider two infinite domains: \mathcal{D} (uninterpreted constants, for readability shown as strings) and \mathcal{Q} (rational numbers), but our results, except where explicitly indicated, hold also for finite domains. We assume that database instances are finite sets of tuples. Additionally, we have the standard built-in predicates.

2.1 Preference relations

We adopt here the framework of [Chomicki, 2003].

Definition 2.1 Given a relation schema $R(A_1 \dots A_k)$ such that U_i , $1 \leq i \leq k$, is the domain (either \mathcal{D} or \mathcal{Q}) of the attribute A_i , a relation \succ is a preference relation over R if it is a subset of $(U_1 \times \dots \times U_k) \times (U_1 \times \dots \times U_k)$.

Although we assume that database instances are finite, in the presence of infinite domains preference relations can be infinite.

Typical properties of a preference relation \succ include:

- *irreflexivity*: $\forall x. x \not\succ x$;
- *transitivity*: $\forall x, y, z. (x \succ y \wedge y \succ z) \Rightarrow x \succ z$;
- *negative transitivity*: $\forall x, y, z. (x \not\succ y \wedge y \not\succ z) \Rightarrow x \not\succ z$;
- *connectivity*: $\forall x, y. x \succ y \vee y \succ x \vee x = y$;
- *strict partial order (SPO)* if \succ is irreflexive and transitive;
- *weak order* if \succ is a negatively transitive SPO;
- *total order* if \succ is a connected SPO.

Definition 2.2 A preference formula (pf) $C(t_1, t_2)$ is a first-order formula defining a preference relation \succ_C in the standard sense, namely

$$t_1 \succ_C t_2 \text{ iff } C(t_1, t_2).$$

An intrinsic preference formula (ipf) is a preference formula that uses only built-in predicates.

By using the notation \succ_C for a preference relation, we assume that there is an underlying pf C . Occasionally, we will limit our attention to ipfs consisting of the following two kinds of atomic formulas (assuming we have two kinds of variables: \mathcal{D} -variables and \mathcal{Q} -variables):

- *equality constraints*: $x = y$, $x \neq y$, $x = c$, or $x \neq c$, where x and y are \mathcal{D} -variables, and c is an uninterpreted constant;
- *rational-order constraints*: $x \theta y$ or $x \theta c$, where $\theta \in \{=, \neq, <, >, \leq, \geq\}$, x and y are \mathcal{Q} -variables, and c is a rational number.

An ipf whose all atomic formulas are equality (resp. rational-order) constraints will be called an *equality* (resp. *rational-order*) ipf. Clearly, ipfs are a special case of general constraints [Kuper, Libkin, & Paredaens, 2000], and define *fixed*, although possibly infinite, relations.

Every preference relation \succ generates an indifference relation \sim : two tuples t_1 and t_2 are *indifferent* ($t_1 \sim t_2$) if neither is preferred to the other one, i.e., $t_1 \not\succ t_2$ and $t_2 \not\succ t_1$. We will denote by \sim_C the indifference relation generated by \succ_C .

Composite preference relations are defined from simpler ones using logical connectives. We focus on two basic ways of composing preference relations:

- *union*:

$$t_1 (\succ_1 \cup \succ_2) t_2 \text{ iff } t_1 \succ_1 t_2 \vee t_1 \succ_2 t_2;$$

- *prioritized composition* (where \sim_1 is the indifference relation generated by \succ_1):

$$t_1 (\succ_1 \triangleright \succ_2) t_2 \text{ iff } t_1 \succ_1 t_2 \vee (t_1 \sim_1 t_2 \wedge t_1 \succ_2 t_2).$$

We also consider transitive closure:

Definition 2.3 The transitive closure of a preference relation \succ over a relation schema R is a preference relation $TC(\succ)$ over R defined as:

$$(t_1, t_2) \in TC(\succ) \text{ iff } t_1 \succ^n t_2 \text{ for some } n > 0,$$

where:

$$\begin{aligned} t_1 \succ^1 t_2 &\equiv t_1 \succ t_2 \\ t_1 \succ^{n+1} t_2 &\equiv \exists t_3. t_1 \succ t_3 \wedge t_3 \succ^n t_2. \end{aligned}$$

2.2 Winnow

We define now an algebraic operator that picks from a given relation the set of the *most preferred tuples*, according to a given preference relation.

Definition 2.4 [Chomicki, 2003] *If R is a relation schema and \succ a preference relation over R , then the winnow operator is written as $\omega_{\succ}(R)$, and for every instance r of R :*

$$\omega_{\succ}(r) = \{t \in r \mid \neg \exists t' \in r. t' \succ t\}.$$

If a preference relation is defined using a pf C , we write simply ω_C instead of ω_{\succ_C} . A *preference query* is a relational algebra query containing at least one occurrence of the winnow operator.

2.3 Preference revision

The basic setting is as follows: We have a preference relation \succ and revise it with a *revising* preference relation \succ_0 to obtain a *revised* preference relation \succ' . We also call \succ' a *revision* of \succ . We limit ourselves to preference relations over the same schema.

The revisions are characterized by a number of different parameters:

- *axiom preservation*: what order axioms are preserved in \succ' ;
- *content preservation*: what preference relations are preserved in \succ' ;
- *ordering* (of revisions).

Definition 2.5 *A revision \succ' of \succ with \succ_0 is:*

- a transitive (resp. SPO, a weak order) revision if \succ' is transitive (resp. an SPO, a weak order);
- a monotonic revision if $\succ \subseteq \succ'$;
- a refinement revision (refinement for short) if $\succ \cup \succ_0 \subseteq \succ'$;
- an overriding revision if $\succ_0 \supset \succ \subseteq \succ'$.

A refinement is monotonic. An overriding revision does not have to be monotonic because it may fail to preserve \succ .

We order revisions using the symmetric difference (\ominus).

Definition 2.6 *Assume \succ_1 and \succ_2 are two revisions of a preference relation \succ with a preference relation \succ_0 . We say that \succ_1 is closer than \succ_2 to \succ if $\succ_1 \ominus \succ \subseteq \succ_2 \ominus \succ$.*

Definition 2.7 *A minimal (resp. least) revision of \succ with \succ_0 is a revision that is minimal (resp. least) in the closeness order among all revisions of \succ with \succ_0 .*

Similarly, we talk about least transitive refinements (or overriding revisions), least SPO (or weak order) refinements or overriding revisions etc. It is easy to see that if we consider only refinements or overriding revisions of a fixed preference relation, closeness reduces to set containment.

Example 3 *Consider the preference relation $\succ = \{(a, b), (b, c), (a, c)\}$ representing the preference order $a \succ b \succ c$, and the following revision of \succ , $\succ_1 = \{(b, a), (b, c), (a, c)\}$. The revision \succ_1 is the least SPO overriding revision of \succ with $\succ_0 = \{(b, a)\}$. It achieves the effect of swapping a and b in the preference order.*

To further describe the behavior of revisions, we define *preference conflicts*.

Definition 2.8 *A conflict between a preference relation \succ and a preference relation \succ_0 is a pair (t_1, t_2) such that $t_1 \succ_0 t_2$ and $t_2 \succ t_1$. A hidden conflict between \succ and \succ_0 is a pair (t_1, t_2) such that $t_1 \succ_0 t_2$ and there exist s_1, \dots, s_k , $k \geq 1$, such that $t_2 \succ s_1 \succ \dots \succ s_k \succ t_1$ and $t_1 \not\succ_0 s_1 \not\succ_0 \dots \not\succ_0 s_k \not\succ_0 t_2$.*

A hidden conflict is a conflict (if \succ is an SPO) but not necessarily vice versa.

Example 4 *If $\succ_0 = \{(a, b)\}$ and $\succ = \{(b, a)\}$, then (a, b) is a conflict which is not hidden. If we add (b, c) and (c, a) to \succ , then the conflict is also a hidden conflict ($s_1 = c$). If we further add (c, b) or (a, c) to \succ_0 , then the conflict is not hidden anymore.*

In this paper, we focus on refinement and overriding revisions because in our opinion they capture two basic ways of revising preferences. A refinement does not retract any preferences or resolve conflicts: it only adds new preferences necessitated by order properties. So for a refinement to satisfy SPO properties, all conflicts need to be avoided. An overriding revision, on the other hand, can override some of the original preferences if they conflict with the new ones. Overriding can deal with conflicts which are not hidden and solves all of them in the same fashion: it gives higher priority to new preference information (i.e., \succ_0). Both refinement and overriding revisions preserve the revising relation \succ_0 .

We now characterize those combinations of \succ and \succ_0 that avoid all (or only hidden) conflicts.

Definition 2.9 *A preference relation \succ is compatible (resp. semi-compatible) with a preference relation \succ_0 if there are no conflicts (resp. no hidden conflicts) between \succ and \succ_0 .*

Compatibility is symmetric and implies semi-compatibility for SPOs. Semi-compatibility is not necessarily symmetric. Examples 1 and 2 show a pair of compatible relations. The compatibility of \succ and \succ_0 does not require the acyclicity of $\succ \cup \succ_0$ or that one of the following hold: $\succ \subseteq \succ_0$, $\succ_0 \subseteq \succ$, or $\succ \cap \succ_0 = \emptyset$. For the former, consider $\succ = \{(a, b), (c, d)\}$ and $\succ_0 = \{(b, c), (d, a)\}$. For the latter, consider $\succ = \{(a, b), (b, c), (a, c)\}$ and $\succ_0 = \{(a, b), (a, d)\}$.

All the properties listed above, including both variants of compatibility, are decidable for equality or rational order ipfs. For example, semi-compatibility is expressed by the condition $\succ_0^{-1} \cap TC(\succ^{-1} - \succ_0^{-1}) = \emptyset$ where \succ^{-1} is the inverse of the preference relation \succ .

3 Preservation of order axioms

We prove now a number of results that characterize refinement and overriding revisions of preference relations. The results are of the form:

Given that the original preference relation \succ and the revising relation \succ_0 satisfy certain order axioms, what kind of order axioms does the revision \succ' satisfy?

To capture minimal change of preferences, we typically study *least* revisions. The revision setting helps to overcome

the limitations of *preference composition* [Chomicki, 2003] where it is shown that common classes of orders (SPOs, weak orders) are often not closed w.r.t. basic preference composition operators like union or prioritized composition. In the results that follow, we obtain closure under least revisions thanks to (1) restricting \succ and \succ_0 , and (2) guaranteeing transitivity by explicitly applying transitive closure where necessary.

3.1 General properties

Lemma 3.1 For compatible \succ and \succ_0 ,

$$\succ_0 \cup \succ = \succ_0 \triangleright \succ.$$

Lemma 3.2 The preference relation $\succ \cup \succ_0$ (resp. $\succ_0 \triangleright \succ$) is contained in every refinement (resp. overriding revision) of \succ with \succ_0 and is, therefore, the least refinement (resp. least overriding revision) of \succ with \succ_0 .

Lemma 3.3 The preference relation $TC(\succ \cup \succ_0)$ (resp. $TC(\succ_0 \triangleright \succ)$) is contained in every transitive refinement (resp. every overriding revision) of \succ with \succ_0 and is, therefore, the least transitive refinement (resp. least transitive overriding revision) of \succ with \succ_0 .

3.2 Strict partial orders

SPOs have several important properties from the user's point of view, and thus their preservation is desirable. For instance, all the preference relations defined in [Kießling, 2002] and the language Preference SQL [Kießling & Köstler, 2002] are SPOs. Moreover, if \succ is an SPO, then the winnow $\omega_\succ(r)$ is nonempty if (a finite) r is nonempty. Also, the fundamental algorithms for computing winnow require that the preference relation be an SPO [Chomicki, 2003].

In order to obtain the least SPO revisions, we have to make sure that $TC(\succ \cup \succ_0)$ and $TC(\succ \triangleright \succ_0)$ are irreflexive (they are transitive by definition).

Definition 3.1 An SPO has the single-chain property (SCP) if it has at most one maximal chain (maximal totally-ordered subset) having at least two elements. Such a chain is called a superchain.

The superchain in the above definition does not have to exhaust all the elements of the domain, so an order having SCP does not have to be total or even weak.

Theorem 3.1 For every compatible preference relations \succ and \succ_0 such that both are SPOs and at least one has SCP, the preference relation $TC(\succ \cup \succ_0)$ is the least SPO refinement of \succ with \succ_0 .

Proof. (sketch) Assume \succ has SCP. If $TC(\succ \cup \succ_0)$ is not irreflexive, then $\succ \cup \succ_0$ has a cycle. Consider such cycle of minimum length. It consists of alternating \succ_0 - and \succ -edges (otherwise it can be shortened). If there is more than one \succ -edge in the cycle, then one of the assumptions is violated. So the cycle consists of two edges: $t_1 \succ_0 t_2$ and $t_2 \succ t_1$. But this is a conflict violating compatibility. \square

Example 5 Consider again the preference relation \succ_{C_1} :

$$(m, y) \succ_{C_1} (m', y') \equiv m = m' \wedge y > y'.$$

Suppose that the new preference information is captured as \succ_{C_3} which is a single-chain SPO:

$$\begin{aligned} (m, y) \succ_{C_3} (m', y') &\equiv m = \text{"VW"} \wedge y = 1999 \\ &\wedge m' = \text{"Kia"} \wedge y' = 1999. \end{aligned}$$

Then $TC(\succ_{C_1} \cup \succ_{C_3})$ is defined as the SPO \succ_{C_4} :

$$\begin{aligned} (m, y) \succ_{C_4} (m', y') &\equiv m = m' \wedge y > y' \\ \vee m = \text{"VW"} \wedge y \geq 1999 \wedge m' = \text{"Kia"} \wedge y' \leq 1999. \end{aligned}$$

One can find examples where SCP or the compatibility of \succ and \succ_0 is violated, and a cycle in $\succ \cup \succ_0$ is obtained.

For dealing with overriding revisions compatibility can be replaced by a less restrictive condition, *semi-compatibility*, because prioritized composition already provides a way of resolving some conflicts.

Theorem 3.2 For every preference relations \succ and \succ_0 such that both are SPOs, \succ_0 has SCP and \succ is semi-compatible with \succ_0 , the preference relation $TC(\succ_0 \triangleright \succ)$ is the least SPO overriding revision of \succ with \succ_0 .

Proof. (sketch) We assume that $TC(\succ_0 \triangleright \succ)$ is not irreflexive and consider a cycle of minimum length in $\succ_0 \triangleright \succ$. This cycle has to consist of an edge $t_1 \succ_0 t_2$ and a number of \succ -edges $t_2 \succ t_3, \dots, t_{n-1} \succ t_n, t_n \succ t_1$ such that $n > 2$. (Here we cannot shorten sequences of consecutive \succ -edges because \succ is not necessarily preserved in $\succ_0 \triangleright \succ$.) We have that $t_2 \sim_0 t_3, \dots, t_{n-1} \sim_0 t_n, t_n \sim_0 t_1$. Thus (t_1, t_2) is a hidden conflict violating the semi-compatibility of \succ with \succ_0 . \square

Again, violating any of the conditions of Theorem 3.2 may lead to a situation in which no SPO overriding revision exists.

Proposition 3.1 For the preference relations defined using equality or rational order ipfs, the computation of $TC(\succ \cup \succ_0)$ and $TC(\succ \triangleright \succ_0)$ terminates.

The computation of transitive closure is done in a completely database-independent way using Constraint Datalog techniques [Kuper, Libkin, & Paredaens, 2000].

Example 6 Consider Examples 1 and 5. We can infer that

$$t_1 = (\text{"VW"}, 2002) \succ_{C_4} (\text{"Kia"}, 1997) = t_3,$$

because

$$(\text{"VW"}, 2002) \succ_{C_1} (\text{"VW"}, 1999),$$

$$(\text{"VW"}, 1999) \succ_{C_3} (\text{"Kia"}, 1999),$$

and

$$(\text{"Kia"}, 1999) \succ_{C_1} (\text{"Kia"}, 1997).$$

The tuples $(\text{"VW"}, 1999)$ and $(\text{"Kia"}, 1999)$ are not in the database.

3.3 Weak orders

Weak partial orders are practically important because they capture the situation where the domain can be decomposed into layers such that the layers are totally ordered and all the elements in one layer are mutually indifferent. This is the case, for example, if the preference relation can be represented using a numeric utility function. If the preference

relation is a weak order, a particularly efficient (essentially single pass) algorithm for computing winnow is applicable [Chomicki, 2004].

We first consider combinations of SPOs and weak orders.

Theorem 3.3 *For every compatible preference relations \succ and \succ_0 such that one is an SPO and the other a weak order, the preference relation $\succ \cup \succ_0$ is the least SPO refinement of \succ with \succ_0 .*

In the context of overriding revisions, the requirement of compatibility becomes unnecessary.

Theorem 3.4 *For every preference relations \succ_0 and \succ such that \succ_0 is a weak order and \succ an SPO, the preference relation $\succ_0 \triangleright \succ$ is the least SPO overriding revision of \succ with \succ_0 .*

We consider now combinations of weak orders.

Theorem 3.5 *For every compatible weak order preference relations \succ and \succ_0 , $\succ \cup \succ_0$ is the least weak order refinement of \succ with \succ_0 .*

Again, for overriding revisions, we can relax the compatibility assumption. This immediately follows from the fact that weak orders are closed with respect to prioritized composition [Chomicki, 2003].

Proposition 3.2 *For every weak order preference relations \succ and \succ_0 , the preference relation $\succ_0 \triangleright \succ$ is the least weak order overriding revision of \succ with \succ_0 .*

A basic notion in utility theory is that of *representability* of preference relations using numeric utility functions:

Definition 3.2 *A real-valued function u over a schema R represents a preference relation \succ over R iff*

$$\forall t_1, t_2 [t_1 \succ t_2 \text{ iff } u(t_1) > u(t_2)].$$

Being a weak order is a necessary condition for the existence of a numeric representation for a preference relation. However, it is not sufficient for uncountable orders [Fishburn, 1970]. It is natural to ask whether the existence of numeric representations for the preference relations \succ and \succ_0 implies the existence of such a representation for the least refinement $\succ' = (\succ \cup \succ_0)$. This is indeed the case.

Theorem 3.6 *Assume that \succ and \succ_0 are weak order preference relations such that*

1. \succ and \succ_0 are compatible,
2. \succ can be represented using a real-valued function u ,
3. \succ_0 can be represented using a real-valued function u_0 .

Then $\succ' = \succ \cup \succ_0$ is a weak order preference relation that can be represented using any real-valued function u' such that for all x , $u'(x) = a \cdot u(x) + b \cdot u_0(x) + c$ where $a, b > 0$.

Surprisingly, the compatibility requirement cannot in general be replaced by semi-compatibility if we replace \cup by \triangleright in Theorem 3.6. This follows from the fact that the lexicographic composition of one-dimensional standard orders over \mathcal{R} is not representable using a utility function [Fishburn, 1970]. Thus, preservation of *representability* is possible only under compatibility, in which case $\succ_0 \triangleright \succ = \succ_0 \cup \succ$ (Lemma 3.1) and the revision is monotonic. It

is an open question whether representability can be preserved under nonmonotonic revisions.

We conclude this section by showing a general scenario in which the refinement of weak orders occurs in a natural way. Assume that we have a numeric utility function u representing a (weak order) preference relation \succ . The indifference relation \sim generated by \succ is defined as:

$$x \sim y \equiv u(x) = u(y).$$

Suppose that the user discovers that \sim is too coarse and needs to be further refined. This may occur, for example, when x and y are tuples and the function u takes into account only some of their components. Another function u_0 may be defined to take into account other components of x and y (such components are called *hidden attributes* [Pu, Faltings, & Torrens, 2003]). The revising preference relation \succ_0 is now:

$$x \succ_0 y \equiv u(x) = u(y) \wedge u_0(x) > u_0(y).$$

It is easy to see that \succ_0 is an SPO compatible with \succ but not necessarily a weak order. Therefore, by Theorem 3.3 the preference relation $\succ \cup \succ_0$ is the least SPO refinement of \succ with \succ_0 .

4 Checking axiom satisfaction

If none of the results described so far implies that the least transitive refinement of \succ with \succ_0 is an SPO, then this condition can often be explicitly checked. Specifically, one has to: (1) compute the transitive closure $TC(\succ \cup \succ_0)$, and (2) check whether the obtained relation is irreflexive.

From Proposition 3.1, it follows that for equality and rational order ipfs the computation of $TC(\succ \cup \succ_0)$ yields some finite ipf $C(t_1, t_2)$. Then the second step reduces to checking whether $C(t, t)$ is unsatisfiable, which is a decidable problem for equality and rational order ipfs.

Example 7 *Consider Examples 1 and 2. Neither of the preference relations \succ_{C_1} and \succ_{C_2} is a weak order or has SCP. Therefore, the results established earlier in this paper do not apply. The preference relation $\succ_{C*} = TC(\succ_{C_1} \cup \succ_{C_2})$ is defined as follows:*

$$(m, y) \succ_{C*} (m', y') \equiv m = m' \wedge y > y' \vee m = "VW" \wedge m' \neq "VW" \wedge y \geq y'$$

The preference relation \succ_{C} is irreflexive. It also properly contains $\succ_{C_1} \cup \succ_{C_2}$, because $t_1 \succ_{C*} t_3$ but $t_1 \not\succ_{C_1} t_3$ and $t_1 \not\succ_{C_2} t_3$. The query $\omega_{C*}(Car)$ evaluated in the instance r_1 (Figure 1) returns only the tuple t_1 .*

Similar considerations apply to overriding revisions and weak orders.

5 Iterating monotonic preference revision

Consider the scenario in which we iterate monotonic preference revision to obtain a sequence of preference relations \succ_1, \dots, \succ_n such that each is an SPO and $\succ_1 \subseteq \dots \subseteq \succ_n$. (Recall that refinement is monotonic but overriding revision not necessarily so.) Assume that those relations are used to extract the best tuples from a fixed relation instance r . Such

evaluation provides feedback to the user about the quality of the given preference relation and may be helpful in constructing its subsequent refinements.

In this scenario, the sequence of query results is:

$$r_0 = r, r_1 = \omega_{\succ_1}(r), r_2 = \omega_{\succ_2}(r), \dots, r_n = \omega_{\succ_n}(r).$$

Proposition 5.1 below implies that the sequence r_0, r_1, \dots, r_n is decreasing:

$$r_0 \supseteq r_1 \supseteq \dots \supseteq r_n$$

and that it can be computed incrementally:

$$r_1 = \omega_{\succ_1}(r_0), r_2 = \omega_{\succ_2}(r_1), \dots, r_n = \omega_{\succ_n}(r_{n-1}).$$

To compute r_i , there is no need to look at the tuples in $r - r_{i-1}$, nor to recompute winnow from scratch. The sets of tuples r_1, \dots, r_n are likely to have much smaller cardinality than $r_0 = r$.

Proposition 5.1 [Chomicki, 2003] *If \succ_1 and \succ_2 are preference relations over a relation schema R and $\succ_1 \subseteq \succ_2$, then for all instances r of R :*

- $\omega_{\succ_2}(r) \subseteq \omega_{\succ_1}(r)$;
- $\omega_{\succ_2}(\omega_{\succ_1}(r)) = \omega_{\succ_2}(r)$ if \succ_1 and \succ_2 are SPOs.

6 Related work

CP-nets [Boutilier *et al.*, 2004] are an influential recent formalism for reasoning with conditional preference statements under *ceteris paribus* semantics (such semantics is also adopted in other work [McGeachie & Doyle, 2004; Wellman & Doyle, 1991]). We conjecture that CP-nets can be expressed in the framework of preference relations of [Chomicki, 2003], used in the present paper, by making the semantics explicit. If the conjecture is true, the results of the present paper will be relevant to revision of CP-nets.

Example 8 *The CP-net $M = \{a \succ \bar{a}, a : b \succ \bar{b}, \bar{a} : \bar{b} \succ b\}$ where a and b are Boolean variables, captures the following preferences: (1) prefer a to \bar{a} , all else being equal; (2) if a , prefer b to \bar{b} ; (3) if \bar{a} , prefer \bar{b} to b . We construct a preference relation \succ_{C_M} between worlds, i.e., Boolean valuations of a and b :*

$$\begin{aligned} (a, b) \succ_{C_M} (a', b') &\equiv a = 1 \wedge a' = 0 \wedge b = b' \\ &\vee a = 1 \wedge a' = 1 \wedge b = 1 \wedge b' = 0 \\ &\vee a = 0 \wedge a' = 0 \wedge b = 0 \wedge b' = 1. \end{aligned}$$

Finally, the semantics of the CP-net is fully captured as the transitive closure $TC(\succ_{C_M})$. Such closure can be computed using Constraint Datalog with Boolean constraints [Kuper, Libkin, & Paredaens, 2000].

CP-nets and related formalisms cannot express preference relations over infinite domains which are essential in database applications.

[Pu, Faltings, & Torrens, 2003] formulates different scenarios of preference revision and does not contain any formal framework. [Freund, 2004] describes minimal change revision of *rational* preference relations between propositional formulas. We are not aware of any work on revising infinite preference relations.

7 Conclusions and future work

We have presented a general framework for revising preference relations and established a number of order axiom preservation results for specific classes of revisions. In the future, we plan to consider more general classes of revisions and databases with restricted domains, e.g., Boolean. Another direction is the design of a *revision language* in which different parameters of preference revision can be explicitly specified by the user. Connections to *iterated belief revision* [Darwiche & Pearl, 1997] should also be explored.

References

- [Boutilier *et al.*, 2004] Boutilier, C.; Brafman, R. I.; Domshlak, C.; Hoos, H. H.; and Poole, D. 2004. CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements. *Journal of Artificial Intelligence Research* 21:135–191.
- [Chomicki, 2003] Chomicki, J. 2003. Preference Formulas in Relational Queries. *ACM Transactions on Database Systems* 28(4):427–466.
- [Chomicki, 2004] Chomicki, J. 2004. Semantic Optimization of Preference Queries. In *International Symposium on Constraint Databases*, 133–148. Paris, France: Springer-Verlag, LNCS 3074.
- [Darwiche & Pearl, 1997] Darwiche, A., and Pearl, J. 1997. On the Logic of Iterated Belief Revision. *Artificial Intelligence* 89(1–2):1–29.
- [Fishburn, 1970] Fishburn, P. C. 1970. *Utility Theory for Decision Making*. Wiley & Sons.
- [Freund, 2004] Freund, M. 2004. On the Revision of Preferences and Rational Inference Processes. *Artificial Intelligence* 152:105–137.
- [Gärdenfors & Rott, 1995] Gärdenfors, P., and Rott, H. 1995. Belief Revision. In Gabbay, D. M.; Hogger, C. J.; and Robinson, J. A., eds., *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4. Oxford University Press. 35–132.
- [Kießling & Köstler, 2002] Kießling, W., and Köstler, G. 2002. Preference SQL - Design, Implementation, Experience. In *International Conference on Very Large Data Bases (VLDB)*, 990–1001.
- [Kießling, 2002] Kießling, W. 2002. Foundations of Preferences in Database Systems. In *International Conference on Very Large Data Bases (VLDB)*, 311–322.
- [Kuper, Libkin, & Paredaens, 2000] Kuper, G.; Libkin, L.; and Paredaens, J., eds. 2000. *Constraint Databases*. Springer-Verlag.
- [McGeachie & Doyle, 2004] McGeachie, M., and Doyle, J. 2004. Utility Functions for Ceteris Paribus Preferences. *Computational Intelligence* 20(2).
- [Pu, Faltings, & Torrens, 2003] Pu, P.; Faltings, B.; and Torrens, M. 2003. User-Involved Preference Elicitation. In *IJCAI Workshop on Configuration*.
- [Wellman & Doyle, 1991] Wellman, M. P., and Doyle, J. 1991. Preferential Semantics for Goals. In *National Conference on Artificial Intelligence*, 698–703.