Iterative Modification and Incremental Evaluation of Preference Queries*

Jan Chomicki

Dept. of Computer Science and Engineering,
University at Buffalo, Buffalo, NY 14260-2000
chomicki@cse.buffalo.edu

Abstract. We present here a formal foundation for an iterative and incremental approach to constructing and evaluating preference queries. Our main focus is on *query modification*: a query transformation approach which works by revising the preference relation in the query. We provide a detailed analysis of the cases where the order-theoretic properties of the preference relation are preserved by the revision. We consider a number of different revision operators: union, prioritized and Pareto composition. We also formulate algebraic laws that enable incremental evaluation of preference queries.

1 Introduction

The notion of *preference* is common in various contexts involving decision or choice. Classical utility theory [10] views preferences as *binary relations*. This view has recently been adopted in database research [7, 8, 20, 22], where preference relations are used in formulating *preference queries*. In AI, various approaches to compact specification of preferences have been explored [6]. The semantics underlying such approaches typically relies on preference relations between worlds.

Preferences can be embedded into database query languages in several different ways. First, [7, 8, 20, 22] propose to introduce a special operator *"find all the most preferred tuples according to a given preference relation."* This operator is called *winnow* in [7, 8]. A special case of winnow is called *skyline* [5] and has been recently extensively studied [25, 3]. Second, [1, 17] assume that preference relations are defined using numeric utility functions and queries return tuples ordered by the values of a supplied utility function. It is well-known that numeric utility functions cannot represent all strict partial orders [10], not even those that occur in database applications in a natural way [8]. For example, utility functions cannot capture skylines. Also, ordered relations go beyond the classical relational model of data. The evaluation and optimization of queries over such relations requires significant changes to relational query processors and optimizers [18]. On the other hand, winnow can be seamlessly combined with any relational operators.

* Research supported by NSF grant IIS-0307434.

We adopt here the first approach, based on winnow, within the preference query framework of [8] (a similar model was described in [20]). In this framework, preference relations between tuples are defined by first-order logical formulas.

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 1a. The set of tuples $\omega_{C_1}(r_1)$ is shown in Figure 1b.

	Make	*Year*
t_1	VW	2002
t_2	VW	1997
t_3	Kia	1997

(a)

	Make	*Year*
t_1	VW	2002
t_3	Kia	1997

(b)

Fig. 1. (a) The Car relation; (b) Winnow result

In this paper, we focus on preference queries of the form $\omega_{\succ}(R)$, consisting of a single occurrence of winnow. Here $\succ$ is a preference relation (typically defined by a formula), and R is a database relation. The relation R represents the space of possible choices. We also briefly discuss how our results can be applied to more general preference queries.

Past work on preference queries has made the assumption that preferences are *static*. However, this assumption is often not satisfied. User preferences change, sometimes as a direct consequence of evaluating a preference query. Therefore, we view preference querying as a *dynamic, iterative process*. The user submits a query and inspects the result. The result may be satisfactory, in which case the querying process terminates. Or, the result may be too large or too small, contain unexpected answers, or fail to contain expected answers. If the user is not satisfied with the query result, she has several further options:

Modify and resubmit the query. This is appropriate if the user decides to refine or change her preferences. For example, the user may have started with a partial or vague concept of her preferences [26]. We consider here query modification consisting of *revising* the preference relation $\succ$, although, of course, more general transformations may also be envisioned.

Update the database. This is appropriate if the user discovers that there are more (or fewer) possible choices than originally envisioned. For example, in comparison shopping the user may have discovered a new source of relevant data.

In this context we pursue the following research challenges:

Defining a repertoire of suitable preference relation revisions. In this work, we consider revisions obtained by *composing* the original preference relation with a new preference relation, and *transitively closing* the result (to guarantee transitivity). We study different composition operators: union, and prioritized and Pareto composition. Those operators represent several basic ways of combining preferences and have already been incorporated into preference query languages [8, 20]. The operators reflect different user attitudes towards *preference conflicts.* (A conflict is, intuitively, a situation in which two preference relations order the same pair of tuples differently.) Union ignores conflicts (and thus such conflicts need to be prevented if we want to obtain a preference relation which is a strict partial order). Prioritized composition resolves preference conflicts by consistently giving priority to one of the preference relations. Pareto composition resolves conflicts in a symmetric way. We emphasize that revision is done using composition because we want the revised preference relation to be uniquely defined in the same first-order language as the original preference relation. Clearly, the revision repertoire that we study in this paper does not exhaust all meaningful scenarios. One can also imagine approaches where axiomatic properties of preference revisions are studied, as in belief revision [13].

Identifying essential properties of preference revisions. We claim that revisions should preserve the order-theoretic properties of the original preference relations. For example, if we start with a preference relation which is a strict partial order, the revised relation should also have those properties. This motivates, among others, transitively closing preference relations to guarantee transitivity. Preserving order-theoretic properties of preference relations is particularly important in view of the iterative construction of preference queries where the output of a revision can serve as the input to another one. We study both necessary and sufficient conditions on the original and revising preference relations that yield the preservation of their order-theoretic properties. Necessary conditions are connected with the absence of preference conflicts. However, such conditions are typically not sufficient and stronger assumptions about the preference relations need to be made. Somewhat surprisingly, a special class of strict partial orders, interval orders, plays an important role in this context. The conditional preservation results we establish in this paper supplement those in [8, 20] and may be used in other contexts where preference relations are composed, for example in the implementation of preference query languages. Another desirable property of revisions is minimality in some well-defined sense. We define minimality in terms of symmetric difference of preference relations but there are clearly other possibilities.

Incremental evaluation of preference queries. At each point of the interaction with the user, the results of evaluating *previous* versions of the given preference query are available. Therefore, they can be used to make the evaluation of the *current* query more efficient. For both the preference revision and database update scenarios, we formulate algebraic laws that validate new query evaluation plans that use materialized results of past query evaluations. The laws use order-theoretic properties of preference relations in an essential way.

Example 2. Consider Example 1. 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 revising it with the following preference relation $\succ_{C_2}$:

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

As there are no conflicts between $\succ_{C_1}$ and $\succ_{C_2}$, the user chooses union as the composition operator. However, to guarantee transitivity of the resulting preference relation, $\succ_{C_1} \cup \succ_{C_2}$ has to be transitively closed. So the revised relation is $\succ_{C*} \equiv TC(\succ_{C_1} \cup \succ_{C_2})$. (The explicit definition of $\succ_{C*}$ is given in Example 6.) The tuple t_3 is now dominated by t_2 (i.e., $t_2 \succ_{C*} t_3$) and will not be returned to the user.

The plan of the paper is as follows. In Section 2, we define the basic notions. In Section 3, we introduce preference revision. In Section 4, we discuss query modification and the preservation by revisions of order-theoretic properties of preference relations. In Section 5, we discuss incremental evaluation of preference queries in the context of query modification and database updates. In Section 6, we consider finite restrictions of preference relations. We briefly discuss related work in Section 7 and conclude in Section 8. Some proofs are outlined. The remaining results can be proved by exhaustive case analysis.

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, countable 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 [8].

Definition 1. *Given a relation schema $R(A_1 \cdots A_k)$ 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* $(U_1 \times \cdots \times U_k) \times (U_1 \times \cdots \times U_k)$.

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 [10]:

- *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;
- *interval order* (IO) [11] if $\succ$ is an SPO and satisfies the condition

$$\forall x, y, z, w.\ (x \succ y \wedge z \succ w) \Rightarrow (x \succ w \vee z \succ y);$$

- *weak order* (WO) if $\succ$ is a negatively transitive SPO;
- *total order* if $\succ$ is a connected SPO.

Every total order is a WO; every WO is an IO.

Definition 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*: $x\lambda y$ or $x\lambda c$, where $\lambda \in \{=, \neq, <, >, \leq, \geq\}$, x and y are $\mathcal{Q}$-variables, and c is a rational number.

An ipf all of whose atomic formulas are equality (resp. rational-order) constraints will be called an *equality* (resp. *rational-order*) ipf. If both equality and rational-order constraints are allowed in a formula, the formula will be called *equality/rational-order*. Clearly, ipfs are a special case of general constraints [23, 19], and define *fixed*, although possibly infinite, relations.

Proposition 1. *Satisfiability of quantifier-free equality/rational-order formulas is in NP.*

Proof. Satisfiability of conjunctions of atomic equality/rational-order constraints can be checked in linear time [15]. In an arbitrary quantifier-free equality/rational-order formula negation can be eliminated. Then in every disjunction one needs to nondeterministically select one disjunct, ultimately obtaining a conjunction of atomic constraints.

Proposition 1 implies that all the properties that can be polynomially reduced to validity of equality/rational-order formulas, for example all the order-theoretic properties listed above, can be decided in co-NP.

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 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 the following basic ways of composing preference relations over the same schema:

– *union:* $t_1\ (\succ_1 \cup \succ_2)\ t_2$ iff $t_1 \succ_1 t_2 \vee t_1 \succ_2 t_2$;
– *prioritized composition:* $t_1\ (\succ_1 \rhd \succ_2)\ t_2$ iff $t_1 \succ_1 t_2 \vee (t_2 \not\succ_1 t_1 \wedge t_1 \succ_2 t_2)$;
– *Pareto composition:*

$$t_1\ (\succ_1 \otimes \succ_2)\ t_2 \text{ iff } (t_1 \succ_1 t_2 \wedge t_2 \not\succ_2 t_1) \vee (t_1 \succ_2 t_2 \wedge t_2 \not\succ_1 t_1).$$

We will use the above composition operators to construct revisions of given preference relations. We also consider transitive closure:

Definition 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:

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

Clearly, in general Definition 3 leads to infinite formulas. However, in the cases that we consider in this paper the preference relation $\succ_{C*}$ will in fact be defined by a finite formula.

Proposition 2. *Transitive closure of every preference relation defined by an equality/rational-order ipf is an ipf of at most exponential size, which can be computed in exponential time.*

Proof. This is because transitive closure can be expressed in Datalog and the evaluation of Datalog programs over equality and rational-order constraints terminates in exponential time (combined complexity) [19].

In the cases mentioned above, the transitive closure of a given preference relation is a relation definable in the signature of the preference formula. But clearly transitive closure itself, unlike union and prioritized or Pareto composition, is not a first-order definable operator.

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 4. [8] *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 $\omega_{\succ_C}$. A *preference query* is a relational algebra query containing at least one occurrence of the winnow operator.

3 Preference Revisions

The basic setting is as follows: We have an *original* 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 assume that $\succ$, $\succ_0$, and $\succ'$ are preference relations over the same schema, and that all of them satisfy at least the properties of SPOs.

In our setting, a revision is obtained by composing $\succ$ with $\succ_0$ using union, prioritized or Pareto composition, and transitively closing the result (if necessary to obtain transitivity). However, we formulate some properties, like minimality or compatibility, in more general terms.

To define minimality, we order revisions using the symmetric difference ($\triangle$).

Definition 5. *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 $\succ_1 \triangle \succ \,\subset\, \succ_2 \triangle \succ$.*

For finite domains and SPOs, the closeness order defined above concides with the order based on the partial-order distance [4] of the revision to the original relation $\succ$.

To further describe the behavior of revisions, we define several kinds of *preference conflicts*. The intuition here is to characterize those conflicts that, when eliminated by prioritized or Pareto composition, reappear if the resulting preference relation is closed by transitivity.

Definition 6. *A* 0-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* 1-conflict *between $\succ$ and $\succ_0$ is a pair (t_1, t_2) such that $t_1 \succ_0 t_2$ and there exist $s_1, \ldots s_k$, $k \geq 1$, such that $t_2 \succ s_1 \succ \cdots \succ s_k \succ t_1$ and $t_1 \not\succ_0 s_k \not\succ_0 \cdots \not\succ_0 s_1 \not\succ_0 t_2$. A* 2-conflict *between $\succ$ and $\succ_0$ is a pair (t_1, t_2) such that there exist $s_1, \ldots, s_k$, $k \geq 1$ and $w_1, \ldots, w_m$, $m \geq 1$, such that $t_2 \succ s_1 \succ \cdots \succ s_k \succ t_1$, $t_1 \not\succ_0 s_k \not\succ_0 \cdots \not\succ_0 s_1 \not\succ_0 t_2$, $t_1 \succ_0 w_1 \succ_0 \cdots \succ_0 w_m \succ t_2$, and $t_2 \not\succ w_m \not\succ \cdots \not\succ w_1 \not\succ t_1$.*

A 1-conflict is a 0-conflict if $\succ$ is an SPO, but not necessarily vice versa. A 2-conflict is a 1-conflict if $\succ_0$ is an SPO. The different kinds of conflicts are pictured in Figures 2 and 3 ($\bar{\succ}$ denotes the complement of $\succ$).

Example 3. If $\succ_0 = \{(a, b)\}$ and $\succ = \{(b, a)\}$, then (a, b) is a 0-conflict which is not a 1-conflict. If we add (b, c) and (c, a) to $\succ$, then the conflict becomes a 1-conflict ($s_1 = c$). If we further add (c, b) or (a, c) to $\succ_0$, then the conflict is not a 1-conflict anymore. On the other hand, if we add (a, d) and (d, b) to $\succ_0$ instead, then we obtain a 2-conflict.

We assume here that the preference relations $\succ$ and $\succ_0$ are SPOs. If $\succ' = TC(\succ \cup \succ_0)$, then for every 0-conflict between $\succ$ and $\succ_0$, we still obviously have $t_1 \succ' t_2$ and $t_2 \succ' t_1$. Therefore, we say that the union does not resolve any conflicts. On the other hand, if $\succ' = TC(\succ_0 \rhd \succ)$, then for each 0-conflict (t_1, t_2), $t_1 \succ_0 \rhd \succ t_2$ and $\neg(t_2 \succ_0 \rhd \succ t_1)$. In the case of 1-conflicts, we get again $t_1 \succ' t_2$ and $t_2 \succ' t_1$. But in the case where a 0-conflict is not a 1-conflict, we get

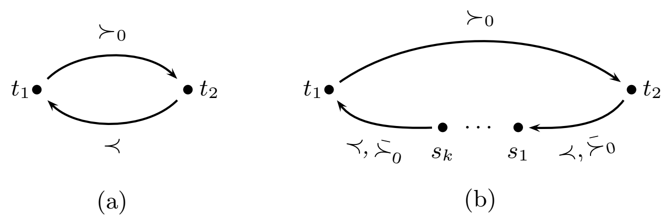

Fig. 2. (a) 0-conflict; (b) 1-conflict

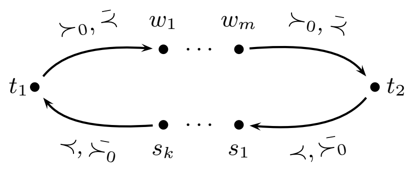

Fig. 3. 2-conflict

only $t_1 \succ' t_2$. Thus we say that prioritized composition resolves those 0-conflicts that are not 1-conflicts. Finally, if $\succ' = TC(\succ \otimes \succ_0)$, then for each 1-conflict (t_1, t_2), $\neg(t_1 \succ \otimes \succ_0 t_2)$ and $\neg(t_2 \succ \otimes \succ_0 t_1)$. We get $t_1 \succ' t_2$ and $t_2 \succ' t_1$ if the conflict is a 2-conflict, but if it is not, we obtain only $t_2 \ \succ' \ t_1$. Thus we say that Pareto composition resolves those 1-conflicts that are not 2-conflicts.

We now characterize those combinations of $\succ$ and $\succ_0$ that avoid different kinds of conflicts.

Definition 7. *A preference relation $\succ$ is i-*compatible*($i = 0, 1, 2$) with a preference relation $\succ_0$ if there are no i-conflicts between $\succ$ and $\succ_0$.*

0- and 2- compatibility are symmetric. 1-compatibility is not necessarily symmetric. For SPOs, 0-compatibility implies 1-compatibility and 1-compatibility implies 2-compatibility. Examples 1 and 2 show a pair of 0-compatible relations. 0-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 $\succ \cap \succ_0 = \emptyset$.

Propositions 1 and 2 imply that all the variants of compatibility defined above are decidable for equality/rational order ipfs. For example, 1-compatibility is expressed by the condition $\succ_0^{-1} \cap TC(\succ - \succ_0^{-1}) = \emptyset$ where $\succ_0^{-1}$ is the inverse of the preference relation $\succ_0$.

0-compatibility of $\succ$ and $\succ_0$ is a *necessary* condition for $TC(\succ \cup \succ_0)$ to be irreflexive, and thus an SPO. Similar considerations apply to $TC(\succ_0 \rhd \succ)$ and 1-compatibility, and $TC(\succ \otimes \succ_0)$ and 2-compatibility. In the next section, we will see that those conditions are not *sufficient*: further restrictions on the preference relations will be introduced.

We conclude by noting that in the absence of conflicts all three notions of preference composition coincide.

Lemma 1. *For every 0-compatible preference relations $\succ$ and $\succ_0$:*

$$\succ_0 \cup \succ = \succ_0 \rhd \succ = \succ_0 \otimes \succ$$

4 Query Modification

In this section, we study preference query modification. A given preference query $\omega_{\succ}(R)$ is transformed to the query $\omega_{\succ'}(R)$ where $\succ'$ is obtained by composing the original preference relation $\succ$ with the revising preference relation $\succ_0$, and transitively closing the result. (The last step is clearly unnecessary if the obtained preference relation is already transitive.) We want $\succ'$ to satisfy the same order-theoretic properties as $\succ$ and $\succ_0$, and to be minimally different from $\succ$. To achieve those goals, we impose additional conditions on $\succ$ and $\succ_0$.

For every $\theta \in \{\cup, \rhd, \otimes\}$, we consider the order-theoretic properties of the preference relation $\succ' = \succ_0 \theta \succ$, or $\succ' = TC(\succ_0 \theta \succ)$ if $\succ_0 \theta \succ$ is not guaranteed to be transitive. To ensure that this preference relation is an SPO, only irreflexivity has to be guaranteed; for weak orders one has also to establish negative transitivity.

4.1 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 [20] and in the language Preference SQL [22] are SPOs. Moreover, if $\succ$ is an SPO, then the winnow $\omega_{\succ}(r)$ is nonempty if (a finite) r is nonempty. The fundamental algorithms for computing winnow require that the preference relation be an SPO [8]. Also, in that case incremental evaluation of preference queries becomes possible (Proposition 4 and Theorem 7).

Theorem 1. *For every* 0*-compatible preference relations* $\succ$ *and* $\succ_0$ *such that one is an interval order (IO) and the other an SPO, the preference relation* $TC(\succ_0 \theta \succ)$*, where* $\theta \in \{\cup, \rhd, \otimes\}$*, is an SPO. Additionally, if the IO is a WO, then* $TC(\succ_0 \theta \succ) = \succ_0 \theta \succ$.

Proof. By Lemma 1, 0-compatibility implies that $\succ_0 \cup \succ = \succ_0 \rhd \succ = \succ_0 \otimes \succ$. Thus, WLOG we consider only union. Assume $\succ_0$ is an IO. 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 edges that are alternately labeled $\succ_0$ (only) and $\succ$ (only). (Otherwise the cycle can be shortened). If there is more than one non-consecutive $\succ_0$-edge in the cycle, then $\succ_0$ being an IO implies that the cycle can be shortened. So the cycle consists of two edges: $t_1 \succ_0 t_2$ and $t_2 \succ t_1$. But this is a 0-conflict violating 0-compatibility of $\succ$ and $\succ_0$.

It is easy to see that there is no preference relation which is an SPO, contains $\succ \cup \succ_0$, and is closer to $\succ$ than $TC(\succ \cup \succ_0)$.

As can be seen from the above proof, the fact that one of the preference relations is an interval order makes it possible to eliminate those paths (and thus also cycles) in $TC(\succ \cup \succ_0)$ that interleave $\succ$ and $\succ_0$ more than once. In this way acyclicity reduces to the lack of 0-conflicts.

It seems that the interval order (IO) requirement in Theorem 1 cannot be weakened without needing to strengthen the remaining assumptions. If neither

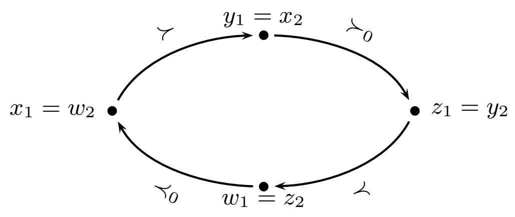

Fig. 4. A cycle for 0-compatible relations that are not IOs

of $\succ$ and $\succ_0$ is an IO, then we can find such elements $x_1, y_1, z_1, w_1, x_2, y_2, z_2, w_2$ that $x_1 \succ y_1, z_1 \succ w_1, x_1 \not\succ w_1, z_1 \not\succ y_1, x_2 \succ_0 y_2, z_2 \succ_0 w_2, x_2 \not\succ_0 w_2$, and $z_2 \not\succ_0 y_2$. If we choose $y_1 = x_2$, $z_1 = y_2$, $w_1 = z_2$, and $x_1 = w_2$, then we get a cycle in $\succ \cup \succ_0$. Note that in this case $\succ$ and $\succ_0$ are still 0-compatible. Also, there is no SPO preference relation which contains $\succ \cup \succ_0$ because each such relation has to contain $TC(\succ \cup \succ_0)$. This situation is pictured in Figure 4.

Example 4. 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 an IO but not a WO:

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

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

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

For dealing with *prioritized composition*, 0-compatibility can be replaced by a less restrictive condition, 1-compatibility, because prioritized composition already provides a way of resolving some conflicts.

Theorem 2. *For every preference relations $\succ$ and $\succ_0$ such that $\succ_0$ is an IO, $\succ$ is an SPO and $\succ$ is 1-compatible with $\succ_0$, the preference relation $TC(\succ_0 \rhd \succ)$ is an SPO.*

Proof. We assume that $TC(\succ_0 \rhd \succ)$ is not irreflexive and consider a cycle of minimum length in $\succ_0 \rhd \succ$. If the cycle has two non-consecutive edges labeled (not necessarily exclusively) by $\succ_0$, then it can be shortened, because $\succ_0$ is an IO. The cycle has to consist of an edge $t_1 \succ_0 t_2$ and a sequence of edges (labeled only by $\succ$): $t_2 \succ t_3, \ldots, t_{n-1} \succ t_n, t_n \succ t_1$ such that $n > 2$ and $t_1 \not\succ_0 t_n \not\succ_0 \ldots \not\succ_0 t_3 \not\succ_0 t_2$. (We cannot shorten sequences of consecutive $\succ$-edges because $\succ$ is not necessarily preserved in $\succ_0 \rhd \succ$.) Thus (t_1, t_2) is a 1-conflict violating 1-compatibility of $\succ$ with $\succ_0$.

Clearly, there is no SPO preference relation which contains $\succ_0 \rhd \succ$, and is closer to $\succ$ than $TC(\succ_0 \rhd \succ)$. Violating any of the conditions of Theorem 2 may lead to a situation in which no SPO preference relation which contains $\succ_0 \rhd \succ$ exists.

If $\succ_0$ is a WO, the requirement of 1-compatibility and the computation of transitive closure are unnecessary.

Theorem 3. *For every preference relations $\succ_0$ and $\succ$ such that $\succ_0$ is a WO and $\succ$ an SPO, the preference relation $\succ_0 \rhd \succ$ is an SPO.*

Let's turn now to *Pareto composition.* There does not seem to be any simple way to *weaken* the assumptions in Theorem 1 using the notion of 2-compatibility. Assuming that $\succ$, $\succ_0$, or even both are IOs does not sufficiently restrict the possible interleavings of $\succ$ and $\succ_0$ in $TC(\succ_0 \otimes \succ)$ because neither of those two preference relations is guaranteed to be preserved in $TC(\succ_0 \otimes \succ)$. However, we can establish a weaker version of Theorem 3.

Theorem 4. *For every preference relations $\succ_0$ and $\succ$ such that both are WOs, the preference relation $\succ_0 \otimes \succ$ is an SPO.*

Proposition 2 implies that for all preference relations defined using equality/-rational-order ipfs, the computation of the preference relations $TC(\succ \cup \succ_0)$, $TC(\succ_0 \rhd \succ)$ and $TC(\succ \otimes \succ_0)$ terminates. The computation of transitive closure is done in a completely database-independent way.

Example 5. Consider Examples 1 and 4. We can infer that

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

because $(''\mathrm{VW}'', 2002) \succ_{C_1} (''\mathrm{VW}'', 1999)$, $(''\mathrm{VW}'', 1999) \succ_{C_3} (''\mathrm{Kia}'', 1999)$, and $(''\mathrm{Kia}'', 1999) \succ_{C_1} (''\mathrm{Kia}'', 1997)$. Note that the tuples $(''\mathrm{VW}'', 1999)$ and $(''\mathrm{Kia}'', 1999)$ are *not* in the database.

If the conditions of Theorems 1 and 2 do not apply, Proposition 2 implies that for equality/rational order ipfs the computation of $TC(\succ \cup \succ_0)$, $TC(\succ_0 \rhd \succ)$ and $TC(\succ \otimes \succ_0)$ yields some finite ipf $C(t_1, t_2)$. Thus the irreflexivity of the resulting preference relation reduces to the unsatisfiability of $C(t,t)$, which by Proposition 1 is a decidable problem for equality/rational order ipfs. Of course, the relation, being a transitive closure, is already transitive.

Example 6. Consider Examples 1 and 2. Neither of the preference relations $\succ_{C_1}$ and $\succ_{C_2}$ is an interval order. Therefore, the results established earlier in this section do not apply. The preference relation $\succ_{C*} = TC(\succ_{C_1} \cup \succ_{C_2})$ is defined as follows (this definition is obtained using Constraint Datalog computation):

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

The preference relation $\succ_{C*}$ is irreflexive (this can be effectively checked). 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.

4.2 Weak Orders

Weak 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 WO, a particularly efficient (essentially single pass) algorithm for computing winnow is applicable [9].

We will see that for weak orders the transitive closure computation is unnecessary and minimal revisions are directly definable in terms of the preference relations involved.

Theorem 5. *For every* 0*-compatible WO preference relations* $\succ$ *and* $\succ_0$*, the preference relations* $\succ \cup \succ_0$ *and* $\succ \otimes \succ_0$ *are WO.*

For prioritized composition, we can relax the 0-compatibility assumption. This immediately follows from the fact that WOs are closed with respect to prioritized composition [8].

Proposition 3. *For every WO preference relations* $\succ$ *and* $\succ_0$*, the preference relation* $\succ_0 \rhd \succ$ *is a WO.*

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

Definition 8. *A real-valued function* u *over a schema* R represents *a preference relation* $\succ$ *over* R *iff*

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

Such a preference relation is called utility-based*.*

Being a WO is a necessary condition for the existence of a numeric representation for a preference relation. However, it is not sufficient for uncountable orders [10]. 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 preference relation $\succ' = (\succ_0\ \theta \succ)$ where $\theta \in \{\cup, \rhd, \otimes\}$. This is indeed the case.

Theorem 6. *Assume that* $\succ$ *and* $\succ_0$ *are WO preference relations such that*

1. $\succ$ *and* $\succ_0$ *are* 0*-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 $\succ' = \succ_0\ \theta \succ$*, where* $\theta \in \{\cup, \rhd, \otimes\}$*, is a WO 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 *and* b *are arbitrary positive real numbers.*

Proof. By case analysis. The assumption of 0-compatibility is essential.

Surprisingly, 0-compatibility requirement cannot in general be replaced by 1-compatibility if we replace $\cup$ by $\rhd$ in Theorem 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 [10]. Thus, preservation of *representability* is possible only under 0-compatibility, in which case $\succ_0 \cup \succ \; = \; \succ_0 \rhd \succ \; = \; \succ_0 \otimes \succ$. (Lemma 1). (The results [10] indicate that for countable domains considered in this paper, the prioritized composition of WOs, being a WO, is representable using a utility function. However, that utility function is not definable in terms of the utility functions representing the given orders.)

We conclude this section by showing a general scenario in which the union of WOs occurs in a natural way. Assume that we have a numeric utility function u representing a (WO) 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* [26]). 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 0-compatible with $\succ$ but not necessarily a WO. Therefore, by Theorem 1 the preference relation $\succ \cup \succ_0$ is an SPO.

5 Incremental Evaluation

5.1 Query Modification

We show here how the already computed result of the original preference query can be reused to make the evaluation of the modified query more efficient. We will use the following result.

Proposition 4. [8] *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.*

Consider the scenario in which we iteratively modify a given preference query by revising the preference relation using only union in such a way that the revised preference relation is an SPO (for example, if the assumptions of Theorem 1 are satisfied). We obtain a sequence of preference relations $\succ_1, \ldots, \succ_n$ such that $\succ_1 \subseteq \cdots \subseteq \succ_n$.

In this scenario, the sequence of query results is:

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

Proposition 4 implies that the sequence $r_0, r_1, \ldots, r_n$ is decreasing:

$$r_0 \supseteq r_1 \supseteq \cdots \supseteq r_n$$

and that it can be computed incrementally:

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

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 $r_1, \ldots, r_n$ are likely to have much smaller cardinality than $r_0 = r$.

It is easy to see that the above comments apply to all cases where the revised preference relation is a superset of the original preference relation. Unfortunately, this is not the case for revisions that use prioritized or Pareto composition. However, given a specific pair of preference relations $\succ$ and $\succ_0$, one can still effectively check whether $TC(\succ_0 \rhd \succ)$ or $TC(\succ_0 \otimes \succ)$ contains $\succ$ if the validity of preference formulas is decidable, as is the case for equality/rational-order formulas (Proposition 1).

5.2 Database Update

In the previous section we studied query modification: the query is modified, while the database remains unchanged. Here we reverse the situation: the query remains the same and the database is updated.

We consider first updates that are insertions of sets of tuples. For a database relation r, we denote by $\Delta^+ r$ the set of inserted tuples. We show how the previous result of a given preference query can be reused to make the evaluation of the same query in an updated database more efficient.

We first establish the following result.

Theorem 7. *For every preference relation $\succ$ over R which is an SPO and every instance r of R:*

$$\omega_\succ(r \cup \Delta^+ r) = \omega_\succ(\omega_\succ(r) \cup \Delta^+ r).$$

Consider now the scenario in which we have a preference relation $\succ$, which is an SPO, and a sequence of relations

$$r_0 = r, r_1 = r_0 \cup \Delta^+ r_0, r_2 = r_1 \cup \Delta^+ r_1, \ldots, r_n = r_{n-1} \cup \Delta^+ r_{n-1}.$$

Theorem 7 shows that

$$\begin{aligned} &\omega_\succ(r_1) = \omega_\succ(\omega_\succ(r_0) \cup \Delta^+ r_0) \\ &\omega_\succ(r_2) = \omega_\succ(\omega_\succ(r_1) \cup \Delta^+ r_1) \\ &\ldots \\ &\omega_\succ(r_n) = \omega_\succ(\omega_\succ(r_{n-1}) \cup \Delta^+ r_{n-1}). \end{aligned}$$

Therefore, each subsequent evaluation of winnow can reuse the result of the previous one. This is advantageous because winnow returns a subset of the given relation and this subset is often much smaller than the relation itself.

Clearly, the algebraic law, stated in Theorem 7, can be used together with other, well-known laws of relational algebra and the laws specific to preference queries [8, 21] to produce a variety of rewritings of a given preference query. To see how a more complex preference query can be handled, let's consider the query consisting of winnow and selection, $\omega_{\succ}(\sigma_\alpha(R))$. We have

$$\omega_{\succ}(\sigma_\alpha(r \cup \Delta^{+}r)) = \omega_{\succ}(\sigma_\alpha(r) \cup \sigma_\alpha(\Delta^{+}r)) = \omega_{\succ}(\omega_{\succ}(\sigma_\alpha(r)) \cup \sigma_\alpha(\Delta^{+}r))$$

for every instance r of R. Here again, one can use the previous result of the query, $\omega_{\succ}(\sigma_\alpha(r))$, to make its current evaluation more efficient. Other operators that distribute through union, for example projection and join, can be handled in the same way.

Next, we consider updates that are deletions of sets of tuples. For a database relation r, we denote by $\Delta^{-}r$ the set of deleted tuples.

Theorem 8. *For every preference relation $\succ$ over R and every instance r of R:*

$$\omega_{\succ}(r) - \Delta^{-}r \subseteq \omega_{\succ}(r - \Delta^{-}r).$$

Theorem 8 gives an incremental way to compute an approximation of winnow from below. It seems that in the case of deletion there cannot be an exact law along the lines of Theorem 7. This is because the deletion of some tuples from the original database may promote some originally dominated (and discarded) tuples into the result of winnow over the updated database.

Example 7. Consider the following preference relation $\succ = \{(a, b_1), \ldots, (a, b_n)\}$ and the database $r = \{a, b_1, \ldots, b_n\}$. Then $\omega_{\succ}(r) = \{a\}$ but $\omega_{\succ}(r - \{a\}) = \{b_1, \ldots, b_n\}$.

6 Finite Restrictions of Preference Relations

It is natural to consider *restrictions* of preference relations to given database instances [27]. If r is an instance of a relation schema R and $\succ$ is a preference relation over R, then $[\succ]_r = \succ \cap\, r \times r$ is also a preference relation over R and $\omega_{[\succ]_r}(r) = \omega_{\succ}(r)$.

The advantage of using $[\succ]_r$ instead of $\succ$ comes from the fact that the former depends on the database contents and can have stronger properties than the latter. For example, $[\succ]_r$ may be an SPO (or a WO), while $\succ$ is not. (Clearly, $[\succ]_r$ inherits all the order-theoretic properties of $\succ$, studied in the present paper.) Similarly, $[\succ]_r$ may be i-compatible with $[\succ_0]_r$, while $\succ$ is not i-compatible with $\succ_0$. On the other hand, $\succ$ makes more elaborate use of the preference information than $[\succ]_r$ and does not require adaptation if the input database changes.

Example 8. Let $\succ = \{(a, b)\}$, $\succ_0 = \{(b, c)\}$, $r = \{a, c\}$. Thus $\omega_{\succ}(r) = \omega_{[\succ]_r}(r) = \{a, c\}$. Consider revision using union, as in Theorem 1. The revised preference relation $\succ_1 = TC(\succ \cup \succ_0) = \{(a, b), (b, c), (a, c)\}$. On the other hand, $[\succ]_r = [\succ_0]_r = \emptyset$. Thus the revised preference relation $\succ_2 = TC([\succ]_r \cup [\succ_0]_r) = \emptyset$. After the revision, $\omega_{\succ_1}(r) = \{a\}$ and $\omega_{\succ_2} = \{a, c\}$. So in the latter case revision has no impact on preference. We also note that $[TC(\succ \cup \succ_0)]_r \neq TC([\succ]_r \cup [\succ_0]_r)$, and thus the correspondence between the unrestricted and the restricted preference relations no longer holds after the revision.

A related issue is that of *non-intrinsic* preference relations. Such relations are defined using formulas that refer not only to built-in predicates.

Example 9. The following preference relation is not intrinsic:

$$x \succ_{Pref} y \equiv Pref(x, y)$$

where $Pref$ is a database relation. One can think of such a relation as representing *stored* preferences.

Revising non-intrinsic preference relations looks problematic. First, it is typically not possible to establish the simplest order-theoretic properties of such relations. For instance, in Example 9 it is not possible to determine the irreflexivity or transitivity of $\succ_{Pref}$ on the basis of its definition. Whether such properties are satisfied depends on the contents of the database relation $Pref$. Second, the transitive closure of a non-intrinsic preference relation may fail to be expressed as a finite formula. Again, Example 9 can be used to illustrate this point. The above problems disappear, however, if we consider $[\succ]_r$ instead of $\succ$.

7 Related Work

[16] presents a general framework for modeling change in preferences. Preferences are represented syntactically using sets of ground preference formulas, and their semantics is captured using sets of preference relations. Thanks to the syntactic representation preference revision is treated similarly, though not identically, to belief revision [13], and some axiomatic properties of preference revisions are identified. The result of a revision is supposed to be minimally different from the original preference relation (using a notion of minimality based on symmetric difference) and satisfy some additional background postulates, for example specific order axioms. [16] does not address the issue of constructing or defining revised relations, nor does it study the properties of specific classes of preference relations. On the other hand, [16] discusses also preference contraction, and domain expansion and shrinking.

In our opinion, there are several fundamental differences between belief and preference revision. 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. Belief revision focuses on axiomatic properties of belief revision operators and various notions of revision minimality. Preference revision focuses on axiomatic, order-theoretic properties of revised preference relations and the definability of such relations (though still taking revision minimality into account).

[28] considers revising a ranking (a WO) of a finite set of product profiles with new information, and shows that a new ranking, satisfying the AGM belief revision postulates [13], can be computed in a simple way. [26] formulates various scenarios of preference revision and does not contain any formal framework. [29] studies revision and contraction of finite WO preference relations by single pairs $t_1 \succ_0 t_2$. [12] describes minimal change revision of *rational* preference relations between propositional formulas.

Two different approaches to preference queries have been pursued in the literature: qualitative and quantitative. In the *qualitative* approach, preferences are specified using binary *preference relations* [24, 14, 7, 8, 20, 22]. In the *quantitative* utility-based approach, preferences are represented using *numeric utility functions* [1, 17], as shown in Section 4. The qualitative approach is strictly more general than the quantitative one, since one can define preference relations in terms of utility functions. However, only WO preference relations can be represented by numeric utility functions [10]. Preferences that are not WOs are common in database applications, c.f., Example 1.

Example 10. There is no utility function that captures the preference relation described in Example 1. Since there is no preference defined between t_1 and t_3 or t_2 and t_3, the score of t_3 should be equal to the scores of both t_1 and t_2. But this implies that the scores of t_1 and t_2 are equal which is not possible since t_1 is preferred over t_2.

This lack of expressiveness of the quantitative approach is well known in utility theory [10].

In the earlier work on preference queries [8, 20], one can find positive and negative results about closure of different classes of orders, including SPOs and WOs, under various composition operators. The results in the present paper are, however, new. Restricting the relations $\succ$ and $\succ_0$ (for example, assuming the interval order property and compatibility) and applying transitive closure where necessary make it possible to come up with positive counterparts of the negative results in [8]. For example, [8] shows that SPOs and WOs are in general not closed w.r.t. union, which should be contrasted with Theorems 1 and 5. In [20], Pareto and prioritized composition are defined somewhat differently from the present paper. The operators combine two preference relations, each defined over some database relation. The resulting preference relation is defined over the Cartesian product of the database relations. So such operators are not useful in the context of revision of preference relations. On the other hand, the careful design of the language guarantees that every preference relation that can be defined is an SPO.

Probably the most thoroughly studied class of qualitative preference queries is the class of *skyline* queries. A skyline query partitions all the attributes of a relation into DIFF, MAX, and MIN attributes. Only tuples with identical values of all DIFF attributes are comparable; among those, MAX attribute values are maximized and MIN values are minimized. The query in Example 1 is a very simple skyline query [5], with *Make* as a DIFF and *Year* as a MAX attribute. Without DIFF attributes, a skyline is a special case of n-ary Pareto composition.

Algorithms for evaluating qualitative preference queries are described in [8, 27], and for evaluating skyline queries, in [5, 25, 3]. [2] describes how to implement preference queries that use Pareto compositions of utility-based preference relations. In Preference SQL [22] general preference queries are implemented by a translation to SQL. [17] describes how materialized results of utility-based preference queries can be used to answer other queries of the same kind.

8 Conclusions and Future Work

We have presented a formal foundation for an iterative and incremental approach to constructing ans evaluating preference queries. Our main focus is on *query modification*, a query transformation approach which works by revising the preference relation in the query. We have provided a detailed analysis of the cases where the order-theoretic properties of the preference relation are preserved by the revision. We considered a number of different revision operators: union, prioritized and Pareto composition. We have also formulated algebraic laws that enable incremental evaluation of preference queries.

Future work includes the integration of our results with standard query optimization techniques, both rewriting- and cost-based. Semantic query optimization techniques for preference queries [9] can also be applied in this context. Another possible direction could lead to the design of a *revision language* in which richer classes of preference revisions can be specified.

One should also consider possible courses of action if the original preference relation $\succ$ and $\succ_0$ lack the property of compatibility, for example if $\succ$ and $\succ_0$ are not 0-compatible in the case of revision by union. Then the target of the revision is an SPO which is the closest to the preference relation $\succ \cup \succ_0$. Such an SPO will not be unique. Moreover, it is not clear how to obtain ipfs defining the revisions. Similarly, one could study *contraction* of preference relations. The need for contraction arises, for example, when a user realizes that the result of a preference query does not contain some expected tuples.

References

1. R. Agrawal and E. L. Wimmers. A Framework for Expressing and Combining Preferences. In *ACM SIGMOD International Conference on Management of Data*, pages 297–306, 2000.
2. W-T. Balke and U. Güntzer. Multi-objective Query Processing for Database Systems. In *International Conference on Very Large Data Bases (VLDB)*, pages 936–947, 2004.
3. W-T. Balke, U. Güntzer, and J. X. Zhang. Efficient Distributed Skylining for Web Information Systems. In *International Conference on Extending Database Technology (EDBT)*, pages 256–273, 2004.
4. K. P. Bogart. Preference Structures I: Distances Between Transitive Preference Relations. *Journal of Mathematical Sociology*, 3:49–67, 1973.

5. S. Börzsönyi, D. Kossmann, and K. Stocker. The Skyline Operator. In *IEEE International Conference on Data Engineering (ICDE)*, pages 421–430, 2001.
6. C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.
7. J. Chomicki. Querying with Intrinsic Preferences. In *International Conference on Extending Database Technology (EDBT)*, pages 34–51. Springer-Verlag, LNCS 2287, 2002.
8. J. Chomicki. Preference Formulas in Relational Queries. *ACM Transactions on Database Systems*, 28(4):427–466, December 2003.
9. J. Chomicki. Semantic Optimization of Preference Queries. In *International Symposium on Constraint Databases*, pages 133–148, Paris, France, June 2004. Springer-Verlag, LNCS 3074.
10. P. C. Fishburn. *Utility Theory for Decision Making.* Wiley & Sons, 1970.
11. P. C. Fishburn. *Interval Orders and Interval Graphs.* Wiley & Sons, 1985.
12. M. Freund. On the Revision of Preferences and Rational Inference Processes. *Artificial Intelligence*, 152:105–137, 2004.
13. P. Gärdenfors and H. Rott. Belief Revision. In D. M. Gabbay, J. Hogger, C, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4, pages 35–132. Oxford University Press, 1995.
14. K. Govindarajan, B. Jayaraman, and S. Mantha. Preference Queries in Deductive Databases. *New Generation Computing*, 19(1):57–86, 2000.
15. S. Guo, W. Sun, and M.A. Weiss. Solving Satisfiability and Implication Problems in Database Systems. *ACM Transactions on Database Systems*, 21(2):270–293, 1996.
16. S. O. Hansson. Changes in Preference. *Theory and Decision*, 38:1–28, 1995.
17. V. Hristidis and Y. Papakonstantinou. Algorithms and Applications for Answering Ranked Queries using Ranked Views. *VLDB Journal*, 13(1):49–70, 2004.
18. I. F. Ilyas, R. Shah, and A. K. Elmagarmid W. G. Aref, J. S. Vitter. Rank-aware Query Optimization. In *ACM SIGMOD International Conference on Management of Data*, pages 203–214, 2004.
19. P. C. Kanellakis, G. M. Kuper, and P. Z. Revesz. Constraint Query Languages. *Journal of Computer and System Sciences*, 51(1):26–52, August 1995.
20. W. Kießling. Foundations of Preferences in Database Systems. In *International Conference on Very Large Data Bases (VLDB)*, pages 311–322, 2002.
21. W. Kießling and B. Hafenrichter. Algebraic Optimization of Relational Preference Queries. Technical Report 2003-1, Institut für Informatik, Universität Augsburg, 2003.
22. W. Kießling and G. Köstler. Preference SQL - Design, Implementation, Experience. In *International Conference on Very Large Data Bases (VLDB)*, pages 990–1001, 2002.
23. G. Kuper, L. Libkin, and J. Paredaens, editors. *Constraint Databases.* Springer-Verlag, 2000.
24. M. Lacroix and P. Lavency. Preferences: Putting More Knowledge Into Queries. In *International Conference on Very Large Data Bases (VLDB)*, pages 217–225, 1987.
25. D. Papadias, Y. Tao, G. Fu, and B. Seeger:. An Optimal and Progressive Algorithm for Skyline Queries. In *ACM SIGMOD International Conference on Management of Data*, pages 467–478, 2003.

26. P. Pu, B. Faltings, and M. Torrens. User-Involved Preference Elicitation. In *IJCAI Workshop on Configuration*, 2003.
27. R. Torlone and P. Ciaccia. Which Are My Preferred Items? In *Workshop on Recommendation and Personalization in E-Commerce*, May 2002.
28. Mary-Anne Williams. Belief Revision via Database Update. In *International Intelligent Information Systems Conference*, 1997.
29. S. T. C. Wong. Preference-Based Decision Making for Cooperative Knowledge-Based Systems. *ACM Transactions on Information Systems*, 12(4):407–435, 1994.