Semantic Optimization of Preference Queries [⋆]

Jan Chomicki

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

Abstract. Preference queries are relational algebra or SQL queries that contain occurrences of the winnow operator (*find the most preferred tuples in a given relation*). We present here a number of semantic optimization techniques applicable to preference queries. The techniques make it possible to remove redundant occurrences of the winnow operator and to apply a more efficient algorithm for the computation of winnow. We also study the propagation of integrity constraints in the result of the winnow. We have identified necessary and sufficient conditions for the applicability of our techniques, and formulated those conditions as *constraint satisfiability* problems.

1 Introduction

The notion of *preference* is becoming more and more ubiquitous in present-day information systems. Preferences are primarily used to filter and personalize the information reaching the users of such systems. In database systems, preferences are usually captured as *preference relations* that are used to build *preference queries* [Cho02,Cho03,Kie02,KK02]. From a formal point of view, preference relations are simply binary relations defined on query answers. Such relations provide an abstract, generic way to talk about a variety of concepts like priority, importance, relevance, timeliness, reliability etc. Preference relations can be defined using logical formulas [Cho02,Cho03] or special preference constructors [Kie02] (preference constructors can be expressed using logical formulas). The embedding of preference relations into relational query languages is typically provided through a relational operator that selects from its argument relation the set of the *most preferred tuples*, according to a given preference relation. This operator has been variously called *winnow* (the term we use here) [Cho02,Cho03], BMO [Kie02], and Best [TC02]. (It is also implicit in skyline queries [BKS01].) Being a relational operator, winnow can clearly be combined with other relational operators, in order to express complex preference queries.

Example 1. We introduce an example used throughout the paper. Consider the relation $Book(ISBN, Vendor, Price)$ and the following preference relation $\succ_{C_1}$ between $Book$ tuples:

[⋆] Research supported by NSF Grant IIS-0307434.

> *prefer one Book tuple to another if and only if their ISBNs are the same and the Price of the first is lower.*

Consider the instance r_1 of *Book* in Figure 1. Then the winnow operator ω_{C_1} returns the set of tuples in Figure 2.

ISBN	*Vendor*	*Price*
0679726691	BooksForLess	\$14.75
0679726691	LowestPrices	\$13.50
0679726691	QualityBooks	\$18.80
0062059041	BooksForLess	\$7.30
0374164770	LowestPrices	\$21.88

Fig. 1. The Book relation

ISBN	*Vendor*	*Price*
0679726691	LowestPrices	\$13.50
0062059041	BooksForLess	\$7.30
0374164770	LowestPrices	\$21.88

Fig. 2. The result of winnow

Example 2. The above example is a one-dimensional skyline query. To see an example of a two-dimensional skyline, consider the schema of *Book* expanded by another attribute *Rating*. Define the following preference relation C_2:

> *prefer one Book tuple to another if and only if their ISBNs are the same and the Price of the first is lower and the Rating of the first is not lower, or the Price of the first is not higher and the Rating of the first is higher.*

Then ω_{C_2} is equivalent to the following skyline (in the terminology of [BKS01]):

```
SKYLINE ISBN DIFF, Price MIN, Rating MAX.
```

The above notation indicates that only books with the same ISBN should be compared, that Price should be minimized, and Rating maximized. In fact, the tuples in the skyline satisfy the property of *Pareto-optimality*, well known in economics.

Preference queries can be reformulated in relational algebra or SQL, and thus optimized and evaluated using standard relational techniques. However, it has been recognized that specialized evaluation and optimization techniques promise in this context performance improvements that are otherwise unavailable. A number of new algorithms for the evaluation of skyline queries (a special class of preference queries) have been proposed [BKS01,CGGL03,KRR02,PTFS03]. Some of them can be used to evaluate general preference queries [Cho03]. Also, algebraic laws that characterize the interaction of winnow with the standard operators of relational algebra have been formulated [Cho03,KH02,KH03]. Such laws provide a foundation for the rewriting of preference queries. For instance, necessary and sufficient conditions for pushing a selection through winnow are described in [Cho03]. The algebraic laws cannot be applied unconditionally. In fact, the preconditions of their applications refer to the *validity* of certain *constraint formulas.*

In this paper, we pursue the line of research from [Cho03] a bit further. We study *semantic optimization* of preference queries. Semantic query optimization has been extensively studied for relational and deductive databases [CGM90]. As a result, a body of techniques dealing with specific query transformations like join elimination and introduction, predicate introduction etc. has been developed. We view semantic query optimization very broadly and classify as *semantic* any query optimization technique that makes use of integrity constraints. In the context of preference queries, we focus on the winnow operator. Despite the presence of specialized evaluation techniques, winnow is still quite an expensive operation. We develop optimizing techniques that:

1. remove redundant occurrences of winnow;
2. recognize when more efficient evaluation of winnow is possible.

More efficient evaluation of winnow can be achieved, for example, if the given preference relation is a *weak order* (a negatively transitive strict partial order). We show that even when the preference relation is not a weak order (as in Example 1), it may become equivalent to a weak order on the relations satisfying certain integrity constraints. We show a very simple, single-pass algorithm for evaluating winnow under those conditions. We also pay attention to the issue of satisfaction of integrity constraints in the result of applying winnow. In fact, some constraints may hold in the result of winnow, even though they do not hold in the relation to which winnow is applied. Combined with known results about the preservation of integrity constraints by relational algebra operators [Klu80,KP82], our results provide a way for optimizing not only single occurrences of winnow but also complex preference queries. As in the case of the algebraic transformations described in [Cho03], the semantic transformations described in this paper have preconditions referring to the validity of certain constraint formulas. Thus, such preconditions can be checked using well established constraint satisfaction techniques [GSW96][1].

The plan of the paper is as follows. In Section 2 we define basic notions. We limit ourselves here to integrity constraints that are *functional dependencies*. In Section 3 we address the issue of eliminating redundant occurrences of winnow. In Section 4 we study weak orders. In Section 5 we characterize dependencies holding in the result of winnow. In Section 6 we show how our results can be

[1] A formula is valid iff its negation is unsatisfiable.

generalized to *constraint-generating dependencies* [BCW99]. We briefly discuss related work in Section 7 and conclude in Section 8.

2 Basic notions

We are working in the context of the relational model of data. For concreteness, we consider two infinite domains: $\mathcal{D}$ (uninterpreted constants) and $\mathcal{Q}$ (rational numbers). Other domains could be considered as well without influencing most of the results of the paper. We assume that database instances are finite. Additionally, we have the standard built-in predicates.

2.1 Preference relations

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 $(U_1 \times \cdots \times U_k) \times (U_1 \times \cdots \times U_k)$.*

Intuitively, $\succ$ will be a binary relation between tuples from the same (database) relation. We say that a tuple t_1 *dominates* a tuple t_2 in $\succ$ if $t_1 \succ t_2$.

Typical properties of the relation $\succ$ include:

- *irreflexivity*: $\forall x.\ x \not\succ x$,
- *asymmetry*: $\forall x, y.\ x \succ y \Rightarrow y \not\succ x$,
- *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*: $\forall x, y.\ x \succ y \vee y \succ x \vee x = y$.

The relation $\succ$ is:

- a *strict partial order* if it is irreflexive and transitive (thus also asymmetric);
- a *weak order* if it is a negatively transitive strict partial order;
- a *total order* if it is a connected strict partial order.

At this point, we do not assume any properties of $\succ$, although in most applications it will satisfy at least the properties of *strict partial order*.

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.*

We will limit our attention to preference relations defined using intrinsic preference formulas.

Because we consider two specific domains, $\mathcal{D}$ and $\mathcal{Q}$, we will have two kinds of variables, $\mathcal{D}$-variables and $\mathcal{Q}$-variables, and two kinds of atomic formulas:

- *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.

Without loss of generality, we will assume that ipfs are in DNF (Disjunctive Normal Form) and quantifier-free (the theories involving the above domains admit quantifier elimination). We also assume that atomic formulas are closed under negation (also satisfied by the above theories). 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 [KLP00], and define *fixed*, although possibly infinite, relations. By using the notation $\succ_C$ for a preference relation, we assume that there is an underlying preference formula C.

Definition 3. *Given an instance r of R and a preference relation $\succ_C$ over R, the* restriction $\succ_C|_r$ *of $\succ_C$ to r is defined as*

$$\succ_C|_r = \succ_C \ \cap\ r \times r.$$

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 formula.

Definition 4. *If R is a relation schema and C a preference formula defining a preference relation $\succ_C$ over R, then the* winnow operator *is written as $\omega_C(R)$, and for every instance r of R:*

$$\omega_C(r) = \{t \in r \mid \neg\exists t' \in r.\ t' \succ_C t\}.$$

A preference query is a relational algebra query containing at least one occurrence of the winnow operator.

Example 3. Consider the relation $Book(ISBN, Vendor, Price)$ (Example 1). The preference relation $\succ_{C_1}$ from this example can be defined using the formula C_1:

$$(i, v, p) \succ_{C_1} (i', v', p') \equiv i = i' \wedge p < p'.$$

The answer to the preference query $\omega_{C_1}(Book)$ provides for every book the information about the vendors offering the lowest price for that book.

2.3 Indifference

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

Proposition 1. *For every preference relation $\succ_C$, every relation r and every tuple $t_1, t_2 \in \omega_C(r)$, we have $t_1 = t_2$ or $t_1 \sim_C t_2$.*

2.4 Functional dependencies

We assume that we are working in the context of a single relation schema R and all the integrity constraints are over that schema. The set of all instances of R satisfying a set of integrity constraints F is denoted as $\mathit{Sat}(F)$. We say that F *entails* an integrity constraint f if every instance satisfying F also satisfies f.

A functional dependency (FD) $f \equiv X \rightarrow Y$, where X and Y are sets of attributes of R can be written down as the following logic formula:

$$\forall t_1 . \forall t_2 . \ [R(t_1) \wedge R(t_2) \wedge t_1[X] = t_2[X]] \Rightarrow t_1[Y] = t_2[Y].$$

We use the following notation:

$$\varphi_f(t_1, t_2) \equiv t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y].$$

For a set of FDs F, we define

$$\varphi_F \equiv \bigwedge_{f \in F} \varphi_f.$$

The *arity* of an FD $f \equiv X \rightarrow Y$ is the cardinality $|X \cup Y|$ of the set of attributes $X \cup Y$. The *arity* of a set of FDs F is the maximum arity of any FD in F.

Note that the set of attributes X in $X \rightarrow Y$ may be empty, meaning that each attribute in Y can assume only a single value.

3 Eliminating redundant occurrences of winnow

Given an instance r of R, the operator ω_C is redundant if $\omega_C(r) = r$. If we consider the class of all instances of R, then such an operator is redundant for every instance iff $\succ_C$ is an empty relation. The latter holds iff C is unsatisfiable. However, we are interested only in the instances satisfying a given set of integrity constraints. Therefore, we will check whether the restriction $\succ_C|_r$ is empty for every instance r satisfying the given set of integrity constraints.

Definition 5. *Given a set of integrity constraints F, the operator ω_C is* redundant w.r.t. a set of integrity constraints F *if $\forall r \in \mathit{Sat}(F)$, $\omega_C(r) = r$.*

Theorem 1. *ω_C is redundant w.r.t. a set of FDs F iff the following formula is unsatisfiable:*

$$\varphi_F(t_1, t_2) \wedge t_1 \succ_C t_2$$

Proof. Assume that formula in the theorem is satisfiable. Then there are tuples t_a and t_b such that $\varphi_F(t_a, t_b)$ and $t_a \succ_C t_b$. Thus $t_b \notin \omega_C(\{t_a, t_b\})$ and thus ω_C is not redundant w.r.t. F. For the other direction, assume ω_C is not redundant w.r.t. F. Then there is an instance $r_0 \in \mathit{Sat}(F)$ and a tuple $t_b \in r_0$ such that $t_b \notin \omega_C(r_0)$. Thus, there must be a tuple t_a in r_0 such that $t_a \succ_c t_b$. Clearly, $\varphi_F(t_a, t_b)$ and therefore the formula in the theorem is satisfiable.

Theorem 1 shows that checking for redundancy w.r.t. a set of FDs F is a *constraint satisfiability* problem.

Example 4. Consider Example 3 in which the FD $ISBN \rightarrow Price$ holds. Then

$$\varphi_F \equiv i_1 = i_2 \Rightarrow p_1 = p_2$$

and $\varphi_F(t_1, t_2) \wedge t_1 \succ_{C_1} t_2$ is

$$(i_1 = i_2 \Rightarrow p_1 = p_2) \wedge i_1 = i_2 \wedge p_1 < p_2.$$

The last formula is clearly unsatisfiable, and thus the implication in Theorem 1 holds and we can infer that ω_{C_1} is redundant w.r.t. $ISBN \rightarrow Price$.

How hard is it to check for redundancy w.r.t. a set of FDs F? We assume that the size of a preference formula C (over a relation R) in DNF is characterized by two parameters: $width(C)$ – the number of disjuncts in C, and $span(C)$ – the maximum number of conjuncts in a disjunct of C. Namely, if $C = D_1 \vee \cdots \vee D_m$, and each $D_i = C_{i,1} \wedge \cdots C_{i,k_i}$, then $width(C) = m$ and $span(C) = \max\{k_1, \ldots, k_m\}$.

Theorem 2. *If:*

- *the cardinality of the set of FDs F is $|F|$ and its arity is at most k;*
- *the given preference relation is defined using an ipf C containing only atomic constraints over the same domain and such that $width(C) \leq m$, $span(C) \leq n$;*
- *the time complexity of checking satisfiability of a conjunctive ipf with n conjuncts is in $O(T(n))$,*

then the time complexity of checking ω_C for redundancy with respect to F is in $O(m\ k^{k|F|}\ T(\max(k|F|, n)))$.

The paper [GSW96] contains several results about checking satisfiability of conjunctive formulas. For instance, in the case of rational-order formulas, this problem is shown to be solvable in $O(n)$. This implies, for example, the following corollary.

Corollary 1. *If a preference relation is defined by a conjunctive rational-order ipf ($m = 1$) and the arity of F is at most 2, then checking ω_C for redundancy w.r.t. F can be done in time $O(n\ 2^{|F|})$.*

An analogous result can be derived for equality formulas. From now on we will only present detailed complexity analysis for rational-order formulas.

4 Weak orders

We have defined weak orders as negatively transitive strict partial orders. Equivalently, they can be defined as strict partial orders for which the indifference relation is transitive. Intuitively, a weak order consists of a number (perhaps infinite) of linearly ordered layers. In each layer, all the elements are mutually indifferent and they are all above all the elements in lower layers.

Example 5. In the preference relation $\succ_{C_1}$ in Example 3, the first, second and third tuples are indifferent with the fourth and fifth tuples. However, the first tuple is preferred to the second, violating the transitivity of indifference. Therefore, the preference relation $\succ_{C_1}$ is not a weak order.

Example 6. A preference relation $\succ_{C_f}$, defined as

$$x \succ_{C_f} y \equiv f(x) > f(y)$$

for some real-valued function f, is a weak order but not necessarily a total order.

4.1 Computing winnow

Many algorithms for evaluating winnow are possible. However, we discuss here those that have a good *blocking* behavior and thus are capable of efficiently processing very large data sets.

We first review BNL (Figure 3), a basic algorithm for evaluating winnow, and then show that for preference relations that are weak orders a much simpler and more efficient algorithm is possible. BNL was proposed in [BKS01] in the context of *skyline queries*. However, [BKS01] also noted that the algorithm requires only the properties of strict partial orders. BNL uses a fixed amount of main memory (a *window*). It also needs a temporary table for the tuples whose status cannot be determined in the current pass, because the available amount of main memory is limited.

BNL keeps in the window the best tuples discovered so far (some of them may also be in the temporary table). All the tuples in the window are mutually indifferent and they all need to be kept, since each may turn out to dominate some input tuple arriving later. For weak orders, however, if a tuple t_1 dominates t_2, then any tuple indifferent to t_1 will also dominate t_2. In this case, indifference is an equivalence relation, and thus it is enough to keep in main memory only a single tuple *top* from the top equivalence class. In addition, one has to keep track of all members of that class (called the *current bucket* B), since they may have to be returned as the result of the winnow. The new algorithm WWO (Winnow for Weak Orders) is shown in Figure 4.

It is clear that WWO requires only a single pass over the input. It uses additional memory (whose size is at most equal to the size of the input) to keep track of the current bucket. However, this memory is only written and read once, the latter at the end of the execution of the algorithm. Clearly, for weak orders WWO is considerably more efficient than BNL. Note that for weak orders BNL

```
1. clear the window $W$ and the temporary table $F$;
2. make $r$ the input;
3. repeat the following until the input is empty:
   (a) for every tuple $t$ in the input:
       – $t$ is dominated by a tuple in $W \Rightarrow$ ignore $t$,
       – $t$ dominates some tuples in $W \Rightarrow$ eliminate the dominated tuples and insert $t$ into $W$,
       – if $t$ and all tuples in $W$ are mutually indifferent $\Rightarrow$ insert $t$ into $W$ (if there is room), otherwise add $t$ to $F$;
   (b) output the tuples from $W$ that were added there when $F$ was empty,
   (c) make $F$ the input, clear the temporary table.
```

Fig. 3. BNL: Blocked Nested Loops

```
1. $top :=$ the first input tuple
2. $B := \{top\}$
3. for every subsequent tuple $t$ in the input:
   – $t$ is dominated by $top \Rightarrow$ ignore $t$,
   – $t$ dominates $top \Rightarrow top := t$; $B := \{t\}$
   – $t$ and $top$ are indifferent $\Rightarrow B := B \cup \{t\}$
4. output $B$
```

Fig. 4. WWO: Weak Order Winnow

does not simply reduce to WWO. Note also that if additional memory is not available, WWO can execute in a small, fixed amount of memory by using two passes over the input: in the first, a top tuple is identified, and in the second, all the tuples indifferent to it are selected.

In [CGGL03] we proposed SFS, a more efficient variant of BNL for skyline queries, in which a presorting step is used. Because sorting may require more than one pass over the input, that approach will also be less efficient than WWO for weak orders (unless the input is already sorted).

4.2 Relative weak orders

Even if a preference relation $\succ_C$ is not a weak order in general, its restriction to a specific instance or a class of instances may be a weak order, and thus WWO may be applied to the computation of winnow. Again, we are going to consider the class of instances $Sat(F)$ for a set of integrity constraints F.

Definition 6. *A preference relation $\succ_C$ is a* weak order relative to a set of integrity constraints F *if* $\forall r \in Sat(F)$, $\succ_C|_r$ *is a weak order.*

Theorem 3. *An irreflexive preference relation $\succ_C$ is a weak order relative to a set of FDs F iff the following formula is unsatisfiable:*

$$\varphi_F(t_1, t_2) \wedge \varphi_F(t_2, t_3) \wedge \varphi_F(t_1, t_3) \wedge t_1 \succ_C t_2 \wedge t_1 \sim_C t_3 \wedge t_2 \sim_C t_3.$$

Example 7. Consider Example 3, this time with the 0-ary FD $\emptyset \Rightarrow ISBN$. (Such a dependency might hold, for example, in a relation resulting from the selection $\sigma_{ISBN=c}$ for some constant c.) Note that

$$(i, v, p) \sim_c (i', v', p') \equiv i \neq i' \vee p = p'.$$

We construct the following formula, according to Theorem 3:

$$i_1 = i_2 \wedge i_2 = i_3 \wedge i_1 = i_3 \wedge i_1 = i_2 \wedge p_1 < p_2 \wedge (i_1 \neq i_3 \vee p_1 = p_3) \wedge (i_2 \neq i_3 \vee p_2 = p_3)$$

which is unsatisfiable. Therefore, $\succ_{C_1}$ is a weak order relative to the FD $\emptyset \Rightarrow ISBN$, and for every instance r satisfying this dependency, $\omega_{C_1}(r)$ can be computed using the single-pass algorithm WWO.

Theorem 4. *If:*

- *the cardinality of the set of FDs F is $|F|$ and its arity is at most k;*
- *the given preference relation is defined using an ipf C containing only atomic constraints over the same domain and such that $width(C) \leq m$, $span(C) \leq n$;*
- *the time complexity of checking satisfiability of a conjunctive ipf with n conjuncts is in $O(T(n))$,*

then the time complexity of checking whether $\succ_C$ is a weak order relative to F is in $O(m\ n^{4m}\ k^{k|F|}\ T(\max(k|F|, m, n)))$.

Corollary 2. *If a preference relation is defined by a conjunctive rational-order ipf $(m = 1)$ and the arity of F is at most 2, then then the time complexity of checking whether $\succ_C$ is a weak order relative to F is in $O(n^5\ 2^{|F|})$.*

5 Propagation of integrity constraints

The study of propagation of integrity constraints by relational operators is essential for semantic optimization of complex queries. We need to know which integrity constraints hold in the results of such operators. The winnow operator returns a subset of a given relation, thus it preserves all the functional dependencies holding in the relation. However, we also know that winnow returns a set of tuples which are mutually indifferent. This property can be used to derive *new* dependencies that hold in the result of winnow without necessarily holding in the input relation. (New dependencies can also be derived for other relational operators, for example selection, as in Example 7.)

Theorem 5. *Let f be an FD and $\succ_C$ an irreflexive preference relation over R. The following formula*

$$t_1 \sim_C t_2 \wedge \neg\varphi_f(t_1, t_2)$$

is unsatisfiable iff for every instance r of R, $\omega_C(r)$ satisfies f.

Proof. We will call the FDs satisfying the condition in Theorem 5 *generated* by $\succ_C$ and denote the set of all such dependencies by G_C. It is easy to show that G_C is closed w.r.t. FD implication. Assume $f \notin G_C$. Then the formula in the theorem is satisfiable. Assume it is satisfied by tuples t_a and t_b ($t_a \neq t_b$ because otherwise $\neg\varphi(t_a, t_b)$ is false). Thus $r_0 = \{t_a, t_b\} \notin Sat(f)$. But $t_a \sim_C t_b$, $t_a \not\succ_C t_a$, and $t_b \not\succ_C t_b$. Thus $r_0 = \omega_C(r_0) \notin Sat(f)$.

In the other direction, assume that there is an instance r_0 such that $\omega_C(r_0) \notin Sat(f)$. By the properties of FDs, we can assume that $\omega_C(r_0)$ consists of two distinct tuples t_a and t_b. By Proposition 1, we know that $t_a \sim_C t_b$. Thus the formula is satisfied by t_a and t_b.

Example 8. Consider Example 3. Then the formula from Theorem 5 is

$$(i_1 \neq i_2 \vee p_1 = p_2) \wedge i_1 = i_2 \wedge p_1 \neq p_2$$

which is clearly unsatisfiable. Thus, the FD $ISBN \rightarrow Price$ holds in the result of ω_{C_1}, even though it might not hold in the input relation.

Theorem 6. *If:*

- *the arity of f is k;*
- *the given preference relation is defined using an ipf C containing only atomic constraints over the same domain and such that $width(C) \leq m$, $span(C) \leq n$;*
- *the time complexity of checking satisfiability of a conjunctive ipf with n conjuncts is in $O(T(n))$,*

then the time complexity of checking checking the condition in Theorem 5 is in $O(kn^{2m}\, T(\max(k, m)))$.

Corollary 3. *If a preference relation is defined by a conjunctive rational-order ipf ($m = 1$) and the arity of f is at most 2, then the time complexity of checking the condition in Theorem 5 is in $O(n^2)$.*

6 Constraint-generating dependencies

Functional dependencies are a special case of *constraint-generating dependencies* [BCW99].

Definition 7. *A* constraint-generating dependency (CGD) *can be expressed a formula of the following form:*

$$\forall t_1. \ldots \forall t_n.\ [R(t_1) \wedge \cdots \wedge R(t_n) \wedge \gamma(t_1, \ldots t_n)] \Rightarrow \gamma'(t_1, \ldots t_n)$$

where $\gamma(t_1, \ldots t_n)$ and $\gamma'(t_1, \ldots t_n)$ are constraints over some constraint theory.

CGDs are equivalent to denial constraints.

Example 9. We give here some examples of CGDs. Consider the relation *Emp* with attributes *Name*, *Salary*, and *Manager*, with *Name* being the primary key. The constraint that *no employee can have a salary greater that that of her manager* is a CGD:

$$\forall n, s, m, s', m'.\ [Emp(n, s, m) \wedge Emp(m, s', m')] \Rightarrow s \leq s'.$$

Similarly, single-tuple constraints (`CHECK` constraints in SQL2) are a special case of CGDs. For example, the constraint that *no employee can have a salary over $200000* is expressed as:

$$\forall n, s, m.\ Emp(n, s, m) \Rightarrow s \leq 200000].$$

It turns out that the problems studied in the present paper can be viewed as specific instances of the *entailment* (implication) of CGDs. To see that, let's define two special CGDs d_2^C and d_3^C for a given preference relation $\succ_C$ (and the corresponding indifference relation $\sim_C$):

$$d_2^C \equiv \forall t_1.\forall t_2.\ R(t_1) \wedge R(t_2) \Rightarrow t_1 \sim_C t_2$$

and

$$d_3^C \equiv \forall t_1.\forall t_2.\forall t_3.\ \ R(t_1) \wedge R(t_2) \wedge R(t_3) \Rightarrow \neg(t_1 \succ_C t_2 \wedge t_1 \sim_C t_3 \wedge t_2 \sim_C t_3).$$

Then we have the following properties that generalize Theorems 1, 3, and 5.

Theorem 7. *ω_C is redundant w.r.t. a set of CGDs F iff F entails d_2^C.*

Theorem 8. *If $\succ_C$ is irreflexive, then $\succ_C$ is a weak order relative to a set of CGDs F iff F entails d_3^C.*

Theorem 9. *If $\succ_C$ is irreflexive, then a CGD f is entailed by d_2^C iff for every instance r of R, $\omega_C(r)$ satisfies f.*

Example 10. Consider the following preference relation $\succ_{C_\alpha}$ where α is a selection condition over the schema R:

$$t_1 \succ_{C_\alpha} t_2 \equiv \alpha(t_1) \wedge \neg\alpha(t_2).$$

This is a very common preference relation expressing the preference for the tuples satisfying some property over those that do not satisfy it. The corresponding indifference relation $\sim_{C_\alpha}$is defined as follows:

$$t_1 \sim_{C_\alpha} t_2 \equiv \alpha(t_1) \wedge \alpha(t_2) \vee \neg\alpha(t_1) \wedge \neg\alpha(t_2).$$

Theorem 7 implies that ω_{C_α} is redundant w.r.t. a set of CGDs F iff F implies the CGD

$$\forall t_1.\forall t_2.\ R(t_1) \wedge R(t_2) \Rightarrow \alpha(t_1) \wedge \alpha(t_2) \vee \neg\alpha(t_1) \wedge \neg\alpha(t_2).$$

The latter dependency is satisfied by an instance r of R if and only if all the tuples in r satisfy α or none does. In both cases $\omega_{C_\alpha}(r) = r$.

The paper [BCW99] contains an effective reduction using *symmetrization* from entailment of CGDs to validity of ∀-formulas in the underlying constraint theory. (A similar construction using *symbol mappings* is presented in [ZO97].) This immediately gives the decidability of the problems discussed in the present paper for equality and rational-order constraints (as well as other constraint theories for which satisfiability of quantifier-free formulas is decidable). A more detailed complexity analysis can be carried out along the lines of Theorems 2, 4, and 6.

For theorems 7, 8 and 9 to hold for a class of integrity constraints, two conditions need to be satisfied: (a) the class should be able to express constraints equivalent to d_2^C and d_3^C , and (b) the notions of entailment and finite entailment (entailment on finite relations) for the class should coincide. If (b) is not satisfied, then the theorems will still hold if reformulated by replacing "entailment" with "finite entailment". Thus, assuming that (a) is satisfied, the effectiveness of checking the preconditions of the above theorems depends on the decidability of finite entailment for the given class of integrity constraints.

7 Related work

The basic reference for semantic query optimization is [CGM90]. The most common techniques are: join elimination/introduction, predicate elimination and introduction, and detecting an empty answer set. [CGK$^+$99] discusses the implementation of predicate introduction and join elimination in an industrial query optimizer. Semantic query optimization techniques for relational queries are studied in [ZO97] in the context of denial and referential constraints, and in [MW00] in the context of constraint tuple-generating dependencies (a generalization of CGDs and classical relational dependencies). FDs are used for reasoning about sort orders in [SSM96].

Two different approaches to preference queries have been pursued in the literature: qualitative and quantitative. In the *qualitative* approach, represented by [LL87,KG94,KKTG95,BKS01,GJM01,Cho02,Cho03,Kie02,KH02,KK02], the preferences between tuples in the answer to a query are specified directly, typically using binary *preference relations.* In the *quantitative* approach, as represented by [AW00,HKP01], preferences are specified indirectly using *scoring functions* that associate a numeric score with every tuple of the query answer. Then a tuple t_1 is preferred to a tuple t_2 iff the score of t_1 is higher than the score of t_2. The qualitative approach is strictly more general than the quantitative one, since one can define preference relations in terms of scoring functions However, not every intuitively plausible preference relation can be captured by scoring functions.

Example 11. There is no scoring function that captures the preference relation described in Example 1. Since there is no preference defined between any of the first three tuples and the fourth one, the score of the fourth tuple should be equal to all of the scores of the first three tuples. But this implies that the scores of the first three tuples are the same, which is not possible since the second tuple is preferred to the first one which in turn is preferred to the third one.

This lack of expressiveness of the quantitative approach is well known in utility theory [Fis99,Fis70]. The importance of weak orders in this context comes from the fact that only weak orders can be represented using real-valued scoring functions (and for countable domains this is also a sufficient condition for the existence of such a representation [Fis70]). In the present paper we do not assume that preference relations are weak orders. We only characterize a condition under which preference relations become weak orders relative to a set of integrity constraints.

[Cho03,KH02,KH03] discuss algebraic optimization of preference queries.

8 Conclusions and further work

We have presented some techniques for semantic optimization of preference queries, focusing on the winnow operator. The simplicity of our results attests to the power of logical formulation of preference relations. However, our results are applicable not only to the original logical framework of [Cho02,Cho03], but also to preference queries defined using preference constructors [Kie02,KK02] and skyline queries [BKS01,CGGL03,KRR02,PTFS03] because those queries can be expressed using preference formulas.

Further work can address, for example, the following issues:

- identifying other semantic optimization techniques for preference queries,
- expanding the class of integrity constraints by considering, for example, tuple-generating dependencies and referential integrity constraints,
- identifying weaker but easier to check sufficient conditions for the application of our techniques,
- considering other preference-related operators like *ranking* [Cho03].

References

[AW00] 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.

[BCW99] M. Baudinet, J. Chomicki, and P. Wolper. Constraint-Generating Dependencies. *Journal of Computer and System Sciences*, 59:94–115, 1999. Preliminary version in ICDT'95.

[BKS01] S. Börzsönyi, D. Kossmann, and K. Stocker. The Skyline Operator. In *IEEE International Conference on Data Engineering (ICDE)*, pages 421–430, 2001.

[CGGL03] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with Presorting. In *IEEE International Conference on Data Engineering (ICDE)*, 2003. Poster.

[CGK$^+$99] Q. Cheng, J. Gryz, F. Koo, C. Leung, L. Liu, X. Qian, and B. Schiefer. Implementation of Two Semantic Query Optimization Techniques in DB2 Universal Database. In *International Conference on Very Large Data Bases (VLDB)*, 1999.

[CGM90] U. S. Chakravarthy, J. Grant, and J. Minker. Logic-Based Approach to Semantic Query Optimization. *ACM Transactions on Database Systems*, 15(2):162–207, 1990.

[Cho02] J. Chomicki. Querying with Intrinsic Preferences. In *International Conference on Extending Database Technology (EDBT)*, pages 34–51. Springer-Verlag, LNCS 2287, 2002.

[Cho03] J. Chomicki. Preference Formulas in Relational Queries. *ACM Transactions on Database Systems*, 28(4):427–466, December 2003.

[Fis70] P. C. Fishburn. *Utility Theory for Decision Making.* Wiley & Sons, 1970.

[Fis99] P. C. Fishburn. Preference Structures and their Numerical Representations. *Theoretical Computer Science*, 217:359–383, 1999.

[GJM01] K. Govindarajan, B. Jayaraman, and S. Mantha. Preference Queries in Deductive Databases. *New Generation Computing*, pages 57–86, 2001.

[GSW96] 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.

[HKP01] V. Hristidis, N. Koudas, and Y. Papakonstantinou. PREFER: A System for the Efficient Execution of Multiparametric Ranked Queries. In *ACM SIGMOD International Conference on Management of Data*, pages 259–270, 2001.

[KG94] W. Kießling and U. Güntzer. Database Reasoning – A Deductive Framework for Solving Large and Complex Problems by means of Subsumption. In *3rd. Workshop On Information Systems and Artificial Intelligence*, pages 118–138. Springer-Verlag, LNCS 777, 1994.

[KH02] W. Kießling and B. Hafenrichter. Optimizing Preference Queries for Personalized Web Services. In *IASTED International Conference on Communications, Internet and Information Technology*, November 2002. Also Tech. Rep. 2002-12, July 2002, Institute of Computer Science, University of Augsburg, Germany.

[KH03] W. Kießling and B. Hafenrichter. Algebraic Optimization of Relational Preference Queries. Technical Report 2003-1, Institut für Informatik, Universität Augsburg, 2003.

[Kie02] W. Kießling. Foundations of Preferences in Database Systems. In *International Conference on Very Large Data Bases (VLDB)*, 2002.

[KK02] W. Kießling and G. Köstler. Preference SQL - Design, Implementation, Experience. In *International Conference on Very Large Data Bases (VLDB)*, 2002.

[KKTG95] G. Köstler, W. Kießling, H. Thöne, and U. Güntzer. Fixpoint Iteration with Subsumption in Deductive Databases. *Journal of Intelligent Information Systems*, 4:123–148, 1995.

[KLP00] G. Kuper, L. Libkin, and J. Paredaens, editors. *Constraint Databases.* Springer-Verlag, 2000.

[Klu80] A. Klug. Calculating Constraints on Relational Tableaux. *ACM Transactions on Database Systems*, 5:260–290, 1980.

[KP82] A. Klug and R. Price. Determining View Dependencies Using Tableaux. *ACM Transactions on Database Systems*, 7, 1982.

[KRR02] D. Kossmann, F. Ramsak, and S. Rost. Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. In *International Conference on Very Large Data Bases (VLDB)*, 2002.

[LL87] M. Lacroix and P. Lavency. Preferences: Putting More Knowledge Into Queries. In *International Conference on Very Large Data Bases (VLDB)*, pages 217–225, 1987.

[MW00] M. Maher and J. Wang. Optimizing Queries in Extended Relational Databases. In *International Conference on Database and Expert Systems Applications (DEXA)*, pages 386–396, 2000.

[PTFS03] 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.

[SSM96] David E. Simmen, Eugene J. Shekita, and Timothy Malkemus. Fundamental Techniques for Order Optimization. In *ACM SIGMOD International Conference on Management of Data*, pages 57–67, 1996.

[TC02] R. Torlone and P. Ciaccia. Which Are My Preferred Items? In *Workshop on Recommendation and Personalization in E-Commerce*, May 2002.

[ZO97] X. Zhang and Z. M. Ozsoyoglu. Implication and Referential Constraints: A New Formal Reasoning. *IEEE Transactions on Knowledge and Data Engineering*, 9(6):894–910, 1997.