Semantic Optimization Techniques for 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*). Such queries are parameterized by specific *preference relations*. Semantic optimization techniques make use of integrity constraints holding in the database. In the context of semantic optimization of preference queries, we identify two fundamental properties: *containment* of preference relations relative to integrity constraints and *satisfaction of order axioms* relative to integrity constraints. We show numerous applications of those notions to preference query evaluation and optimization. As integrity constraints, we consider *constraint-generating dependencies*, a class generalizing functional dependencies. We demonstrate that the problems of containment and satisfaction of order axioms can be captured as specific instances of constraint-generating dependency entailment. This makes it possible to formulate necessary and sufficient conditions for the applicability of our techniques as *constraint validity* problems. We characterize the computational complexity of such problems.

Key words: preference queries, query optimization, query evaluation, integrity constraints

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* [9, 10, 21, 25]. From a formal point of view, preference relations

⋆ Research supported by NSF Grant IIS-0307434. An earlier version of some of the results in this paper was presented in [11].

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 [9, 10] or special preference constructors [21] (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) [9, 10], BMO [21], and Best [36]. It is also implicit in *skyline queries* [6]. 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:*

> 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

Figure 1. The Book relation

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

Figure 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* $\succ_{C_2}$*:*

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 [6]):

```
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 [3, 6, 13, 16, 28, 34]. Some of them can be used to evaluate more general preference queries [2, 10]. Also, algebraic laws that characterize the interaction of winnow with the standard operators of relational algebra have been formulated [10, 23, 24]. 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 [10]. 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 [10] a bit further. We study *semantic optimization* of preference queries. Semantic query optimization has been extensively studied for relational and deductive databases [7]. 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 semantic optimization of preference queries, we identify two fundamental *semantic properties*: *containment* of preference relations relative to integrity constraints and *satisfaction of order axioms* relative to integrity constraints. We show that those notions make it possible to formulate semantic query optimization techniques for preference queries in a uniform way.

We focus on the winnow operator. Despite the presence of specialized evaluation techniques, winnow, being essentially an anti-join, is still quite an expensive operation. We develop optimizing techniques that:

(1) remove redundant occurrences of winnow;

(2) coalesce consecutive applications of winnow;
(3) 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 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 integrity 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 [26, 27], our results provide a way for optimizing not only single occurrences of winnow but also complex preference queries.

As integrity constraints, we consider *constraint-generating dependencies* [4], a class generalizing functional dependencies. Constraint-generating dependencies seem particularly well matched with preference queries, since both the former and the latter are formulated using constraints. We demonstrate that the problems of containment of preference relations and satisfaction of order axioms, relative to integrity constraints, can be captured as specific instances of dependency entailment. Our approach makes it possible to formulate necessary and sufficient conditions for the applicability of the proposed semantic query optimization techniques as *constraint validity* problems and precisely characterize the computational complexity of such problems, partly adopting some of the results of [4].

The plan of the paper is as follows. In Section 2, we provide background material on preference queries and constraint-generating dependencies. In Section 3, we introduce two basic semantic properties: relative containment and relative satisfaction of order axioms. In Section 4, we address the issue of eliminating redundant occurrences of winnow. In Section 5, we study weak orders. In Section 6, we characterize dependencies holding in the result of winnow. In Section 7, we consider the computational complexity of the semantic properties studied in the present paper. We discuss related work in Section 8, and conclude in Section 9.

2 Basic notions

We are working in the context of the relational model of data. For concreteness, we consider two infinite domains: $\mathbf{D}$ (uninterpreted constants) and $\mathbf{Q}$ (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. We refer to relation attributes using their names or positions.

We define *constraints* to be quantifier-free formulas over some signature of built-in operators, interpreted over a fixed domain - in our case **D** or **Q**. We will allow both atomic- and tuple-valued variables in constraints. The notation $t[X]$ denotes the fragment of a tuple t consisting of the values of the attributes in the set X.

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* **D** *or* **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.

Example 3 *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: there may be two different elements x_1 and x_2 such that $f(x_1) \neq f(x_2)$.

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 defin-*

ing 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. Most preference relations are of this form. Moreover, for intrinsic preference relations testing a pair of tuples for dominance can be entirely done on the basis of the contents of those tuples; no database queries need to be evaluated.

Because we consider two specific domains, **D** and **Q**, we will have two kinds of variables, **D**-variables and **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 **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 **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. 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 [30], 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.

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 $t_1 \sim_C t_2$.*

Complex preference relations can be easily defined using Boolean connectives. Here we define a special operator: prioritized composition. The prioritized composition $\succ_{C_1} \rhd \succ_{C_2}$ has the following intuitive reading: *prefer according to $\succ_{C_2}$ unless $\succ_{C_1}$ is applicable.*

Definition 3 *Consider two preference relations $\succ_{C_1}$ and $\succ_{C_2}$ defined over the same schema R. The* prioritized composition $\succ_{C_{1,2}} = \succ_{C_1 \rhd C_2}$ *of $\succ_{C_1}$ and $\succ_{C_2}$*

is a preference relation over R defined as:

$$t_1 \succ_{C_1 \triangleright C_2} t_2 \equiv t_1 \succ_{C_1} t_2 \vee (t_1 \sim_{C_1} t_2 \wedge t_1 \succ_{C_2} t_2).$$

2.2 Winnow

We define now an algebraic operator that selects 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 4 *Consider the relation $Book(ISBN, Vendor, Price)$ (Example 1). The preference relation $\succ_{C_1}$ from this example can be defined using the rational-order ipf 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. Note that the preference relation $\succ_{C_1}$ is a strict partial order.

Example 5 *To see another kind of preferences, consider the following preference relation $\succ_{C_3}$:*

I prefer Warsaw to any other city and prefer any city to Moscow.

This preference relation can be formulated as an equality ipf C_3:

$$x \succ_{C_3} y \equiv x = \text{'Warsaw'} \wedge y \neq \text{'Warsaw'} \vee x \neq \text{'Moscow'} \wedge y = \text{'Moscow'}.$$

2.3 Constraint-generating 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 $Sat(F)$. We say that F *entails* an integrity constraint f, written $F \vdash f$, if every instance satisfying F also satisfies f.

Remember that constraints are arbitrary quantifier-free formulas over some constraint theory - here **D** or **Q**.

Definition 5 [4] *A* constraint-generating dependency (CGD) *can be expressed a formula of the following form:*

$$\forall t_1 \ldots \forall t_k.\ R(t_1) \wedge \cdots \wedge R(t_k) \wedge \gamma(t_1, \ldots t_k) \Rightarrow \gamma'(t_1, \ldots t_k)$$

where $\gamma(t_1, \ldots t_k)$ *and* $\gamma'(t_1, \ldots t_k)$ *are constraints. Such a dependency is called a* k*-dependency.*

CGDs are equivalent to denial constraints. Functional dependencies (FDs) are 2-CGDs, because 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:

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

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.

Example 6 *We give here further 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 than 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:*

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

The paper [4] contains an effective reduction using *symmetrization* from the entailment of CGDs to the validity of $\forall$-formulas (or, equivalently, to the unsatisfiability of quantifier-free formulas) in the underlying constraint theory. This reduction is descibed in Section 7. A similar construction using *symbol mappings* is presented in [37].

3 Properties relative to integrity constraints

We define here two properties fundamental to semantic query optimization of preference queries: containment of preference relations and satisfaction of order axioms.

Definition 6 *A preference relation $\succ_{C_1}$ over a schema R is* contained *in a preference relation $\succ_{C_2}$ over the same schema* relative to a set of integrity constraints F*, written as $\succ_{C_1} \subseteq_F \succ_{C_2}$ if*

$$\forall r \in Sat(F).\ \forall t_1, t_2 \in r.\ t_1 \succ_{C_1} t_2 \Rightarrow t_1 \succ_{C_2} t_2.$$

Clearly, $\succ_{C_1} \subseteq_F \succ_{C_2}$ iff $F \vdash d_0^{C_1,C_2}$, where

$$\boxed{d_0^{C_1,C_2} :\ \forall t_1, t_2.\ R(t_1) \wedge R(t_2) \wedge t_1 \succ_{C_1} t_2 \Rightarrow t_1 \succ_{C_2} t_2.}$$

Satisfaction of order axioms relative to integrity constraints is defined similarly – by relativizing the universal quantifiers in the axioms. Since in this paper we are interested in strict partial and weak orders, we define the following.

Definition 7 *A preference $\succ_C$ over a schema R is a* strict partial order relative to a set of integrity constraints F *if:*

$$\forall r \in Sat(F).\ \forall t \in r.\ t \not\succ_C t$$

$$\forall r \in Sat(F).\ \forall t_1, t_2, t_3 \in r.\ t_1 \succ_C t_2 \wedge t_2 \succ_C t_3 \Rightarrow t_1 \succ_C t_3.$$

Definition 8 *A preference $\succ_C$ over a schema R is a* weak order relative to a set of integrity constraints F *if it is a strict partial order relative to F and*

$$\forall r \in Sat(F).\ \forall t_1, t_2, t_3 \in r.\ t_1 \not\succ_C t_2 \wedge t_2 \not\succ_C t_3 \Rightarrow t_1 \not\succ_C t_3.$$

Again, it is clear that the above properties can be expressed in terms of the entailment of CGDs.

4 Eliminating redundant occurrences of winnow

We consider here two situations in which an occurrence of winnow in a preference query may be eliminated. The first case is that when a single application of winnow does not remove any tuples, and is thus redundant. The second case is more subtle: the interaction between two consecutive applications of winnow is such that one can be eliminated.

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.

Definition 9 *Given a set of integrity constraints F, the operator ω_C is* redundant relative to a set of integrity constraints F *if* $\forall r \in Sat(F)$, $\omega_C(r) = r$.

Theorem 1 *ω_C is redundant relative to a set of FDs F iff $\succ_C \subseteq_F \succ_{False}$ where*

$$t_1 \succ_{False} t_2 \equiv False.$$

Proof. Assume $t_1, t_2 \in r$ for some $r \in Sat(F)$ and $t_1 \succ_C t_2$. Then $t_2 \notin \omega_C(r)$. In the other direction, assume that for some $r \in Sat(F)$, $\omega_C(r) \subset r$. Thus, there must be $t_1, t_2 \in r$ such that $t_1 \succ_C t_2$. ■

Clearly, ω_C is redundant relative to F iff F entails the following CGD:

$$\boxed{d_1^C : \ \forall t_1, t_2.\ R(t_1) \wedge R(t_2) \Rightarrow t_1 \not\succ_C t_2.}$$

The CGD d_1^C always holds in the result of winnow $\omega_C(R)$ and simply says that all the tuples in this result are mutually indifferent.

Example 7 *Consider Example 4 in which the FD $ISBN \rightarrow Price$ holds. ω_{C_1} is redundant relative to $ISBN \rightarrow Price$ because this dependency entails (is, in fact, equivalent to) the dependency*

$$\forall i_1, v_1, p_1, i_2, v_2, p_2.\ Book(i_1, v_1, p_1) \wedge Book(i_2, v_2, p_2) \Rightarrow i_1 \neq i_2 \vee p_1 \geq p_2.$$

The second case where an occurrence of winnow can be eliminated is as follows.

Theorem 2 *Assume F is a set of integrity constraints over a schema R. If $\succ_{C_1}$ and $\succ_{C_2}$ are preference relations over R such that $\succ_{C_1}$ and $\succ_{C_2}$ are strict partial orders relative to F and $\succ_{C_1} \subseteq_F \succ_{C_2}$, then for all instances $r \in Sat(F)$:*

$$\omega_{C_1}(\omega_{C_2}(r)) = \omega_{C_2}(\omega_{C_1}(r)) = \omega_{C_2}(r).$$

Proof. Theorem 6.1 in [10] is a similar result which does not, however, relativize the properties of the given preference relations to the set of instances satisfying the given integrity constraints. The proof of that result can be easily adapted here. ■

Note that in the case of strict partial orders, Theorem 2 implies one direction of Theorem 1.

5 Weak orders

We have defined weak orders as negatively transitive strict partial orders.

We recall below some basic properties of weak orders [14].

Proposition 2 *Let $\succ_C$ be a strict partial order. Then:*

- *$\succ_C$ is negatively transitive iff $\sim_C$ is transitive;*
- *if $\succ_C$ is negatively transitive, then for every x, y and z:*
  - *if $x \succ_C y$ and $y \sim_C z$, then $x \succ_C z$;*
  - *if $x \sim_C y$ and $y \succ_C z$, then $x \succ_C z$.*

Proof. It follows directly from the definition of the indifference relation $\sim_C$ that the negative transitivity of $\succ_C$ implies the transitivity of $\sim_C$.

We show now the second part of the theorem. Assume $x \succ_C y$, $y \sim_C z$ and $x \not\succ_C z$. If $z \succ_C x$, then by transitivity of $\succ_C$, $z \succ_C y$, which contradicts $y \sim_C z$. If $x \sim_C z$, then by transitivity of $\sim_C$ (established above), $x \sim_C y$, which contradicts $x \succ_C y$. The other case is similar.

Assume now that $\sim_C$ is transitive, and that $x \not\succ_C y$ and $y \not\succ_C z$. Then we consider several cases:

- $y \succ_C x$ and $z \succ_C y$, which implies $z \succ_C x$;
- $y \succ_C x$ and $y \sim_C z$, which implies $z \succ_C x$;
- $x \sim_C y$ and $z \succ_C y$, which implies $z \succ_C x$;
- $x \sim_C y$ and $y \sim_C z$, which implies $x \sim_C z$.

In all of those cases, $x \not\succ_C z$. ■

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 8 *In the preference relation $\succ_{C_1}$ in Example 4, 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.*

5.1 Computing winnow

Various algorithms for evaluating winnow have been proposed in the literature. 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 [6] in the context of *skyline queries*. However, [6] 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.

(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.

Figure 3. BNL: Blocked Nested Loops

BNL keeps in the window the best tuples discovered so far (some of such tuples 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. Those ideas are behind a new algorithm WWO (Winnow for Weak Orders), 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

(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

Figure 4. WWO: Winnow for Weak Orders

orders BNL does not simply reduce to WWO: BNL keeps the mutually indifferent tuples from the currently top layer in the main memory window (or in the temporary table) and compares all of them with the input tuple. The latter is clearly superfluous for preference relations that are weak orders. 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 [13] 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).

Even if a preference relation $\succ_C$ is not a weak order in general, it may be a weak order *relative to a class of integrity constraints* F. In those cases, WWO is still applicable if the given relation instance satisfies F. Note that in such a case the original definition of $\succ_C$ can still be used for tuple comparison.

Example 9 *Consider Example 4, 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.) We already know that the preference relation* $\succ_{C_1}$ *is a strict partial order. Being a weak order relative to this FD is captured by the following CGD:*

$$\forall i_1, v_1, p_1, i_2, v_2, p_2, i_3, v_3, p_3.$$
$$Book(i_1, v_1, p_1) \wedge Book(i_2, v_2, p_2) \wedge Book(i_3, v_3, p_3) \wedge \phi_1 \Rightarrow \phi_2$$

where

$$\phi_1 : \ (i_1 \neq i_2 \vee p_1 \geq p_2) \wedge (i_2 \neq i_3 \vee p_2 \geq p_3)$$

and

$$\phi_2 : \ (i_1 \neq i_3 \vee p_1 \geq p_3).$$

We show now that this CGD is entailed by the FD $\emptyset \Rightarrow ISBN$. Assume this is not the case. Then there is an instance of the relation Book consisting of tuples (i_1, v_1, p_1), (i_2, v_2, p_2), and (i_3, v_3, p_3) such ϕ_1 is satisfied but ϕ_2 is not. This instance also satisfies the FD, thus $i_1 = i_2 = i_3$. We consider the formula $\phi_1 \wedge \neg\phi_2$ which can be simplified to

$$p_1 \geq p_2 \wedge p_2 \geq p_3 \wedge p_1 < p_3.$$

The last formula is unsatisfiable.

5.2 Collapsing winnow

We show here that for weak orders consecutive applications of winnow can be collapsed to a single one, using prioritized composition. In contrast with Theorem 2, here we do not impose any conditions on the relationship between the preference relations involved. Recall that

$$d_1^C : \ \forall t_1, t_2.\ R(t_1) \wedge R(t_2) \Rightarrow t_1 \not\succ_C t_2.$$

Theorem 3 *Assume F is a set of integrity constraints over a schema R. If $\succ_{C_1}$ and $\succ_{C_2}$ are preference relations over R such that $\succ_{C_1}$ is a weak order relative to F, then for all instances $r \in Sat(F)$:*

$$\omega_{C_1 \triangleright C_2}(r) = \omega_{C_2}(\omega_{C_1}(r)).$$

Additionally, if $\succ_{C_2}$ is a weak order relative to $F \cup d_1^{C_1}$, then also $\succ_{C_1 \triangleright C_2}$ is a weak order relative to F.

Proof. Let $r \in Sat(F)$. Assume $t \in \omega_{C_2}(\omega_{C_1}(r))$ and $t \notin \omega_{C_1 \triangleright C_2}(r)$. Then there exists $s \in r$ such that $s \succ_{C_1 \triangleright C_2} t$. If $s \succ_{C_1} t$, then $t \notin \omega_{C_1}(r)$ and $t \notin \omega_{C_2}(\omega_{C_1}(r))$. Otherwise, $s \sim_{C_1} t$ and $s \succ_{C_2} t$. If $s \in \omega_{C_1}(r)$, then $t \notin \omega_{C_2}(\omega_{C_1}(r))$. If $s \notin \omega_{C_1}(r)$, then for some $s' \in r$, $s' \succ_{C_1} s$. But then $s' \succ_{C_1} t$ by Proposition 2, because $\succ_{C_1}$ is a weak order. Consequently, $t \notin \omega_{C_1}(r)$.

In the other direction, assume $t \in \omega_{C_1 \triangleright C_2}(r)$ and $t \notin \omega_{C_2}(\omega_{C_1}(r))$. If $t \notin \omega_{C_1}(r)$, then for some $s \in r$, $s \succ_{C_1} t$. Thus, $s \succ_{C_1 \triangleright C_2} t$ and $t \notin \omega_{C_1 \triangleright C_2}(r)$. If $t \in \omega_{C_1}(r)$, then for some $s \in \omega_{C_1}(r)$, $s \sim_{C_1} t$ and $s \succ_{C_2} t$. Thus again, $s \succ_{C_1 \triangleright C_2} t$.

The second part of the theorem can be proved in the same way as Proposition 4.6 in [10]. We can require that $\succ_{C_2}$ be a weak order relative to $F \cup d_1^{C_1}$, not just to F, because the dependency $d_1^{C_1}$ is guaranteed to hold in $\omega_{C_1}(r)$. ■

The assumption that $\succ_{C_1}$ be a weak order in Theorem 3 is essential, as shown in the following example.

Example 10 *Consider a strict partial order $\succ_{C_1}$ which is not a weak order. Then there are tuples t_1, t_2 and t_3 such that $t_1 \succ_{C_1} t_2$ and $t_1 \sim_{C_1} t_3$ and $t_2 \sim_{C_1} t_3$. Consider now a different strict partial order $\succ_{C_2}$ such that $t_2 \succ_{C_2} t_1$ and $t_2 \succ_{C_2} t_3$. Then*

$$\omega_{C_1 \triangleright C_2}(\{t_1, t_2, t_3\}) = \{t_1\} \neq \{t_1, t_3\} = \omega_{C_2}(\{t_1, t_3\}) = \omega_{C_2}(\omega_{C_1}(\{t_1, t_2, t_3\})).$$

We show now how Theorem 3 can be used in query optimization. Consider the choice between WWO and BNL in the context of Theorem 3. If both $\succ_{C_1}$ and $\succ_{C_2}$ are weak orders (relative to F), then it is better to evaluate $\omega_{C_1 \triangleright C_2}(r)$ than $\omega_{C_2}(\omega_{C_1}(r))$ because the former does not require creating intermediate results. In both cases WWO can be used. If $\succ_{C_2}$ is a strict partial order but not necessarily a weak order (relative to F), then in both cases we will have to use BNL ($C_1 \triangleright C_2$ is a strict partial order [12]), so again $\omega_{C_1 \triangleright C_2}(r)$ wins. However, if $\omega_{C_1}(r)$ is small, it would be better to use WWO to compute $r_1 = \omega_{C_1}(r)$ and then compute $\omega_{C_2}(r_1)$ using BNL.

Consider now the presence of views. If $\omega_{C_1}(R)$ is a non-materialized view, then the query $\omega_{C_2}(\omega_{C_1}(R))$ can be first rewritten as $\omega_{C_1 \triangleright C_2}(R)$ and then evaluated without the need for the nested evaluation of $\omega_{C_1}(R)$. On the other hand, if $\omega_{C_1}(R)$ is a materialized view V, then it can be used to answer the query $\omega_{C_1 \triangleright C_2}(R)$ by computing $\omega_{C_2}(V)$.

5.3 Further properties

The list of preference query properties that hold relative to a set of integrity constraints does not end with those formulated above. There are other algebraic properties that hold conditionally [10]. Such properties can often be formulated in a more general form using CGDs.

For example, consider the commutativity of winnow and selection. [10] shows the following result:

Proposition 3 *Given a relation schema R, a selection condition C_1 over R and a preference formula C_2 over R, if the formula*

$$\forall t_1, t_2[(C_1(t_2) \wedge C_2(t_1, t_2)) \Rightarrow C_1(t_1)]$$

is valid, then for all instances r of R:

$$\sigma_{C_1}(\omega_{C_2}(r)) = \omega_{C_2}(\sigma_{C_1}(r)).$$

This result can be generalized to hold relative to a set of integrity constraints.

Theorem 4 *Given a relation schema R, a selection condition C_1 over R, a preference formula C_2 over R, and a set of integrity constraints F over R, if $F \vdash d_2^{C_1,C_2}$ where*

$$\boxed{d_2^{C_1,C_2} : \ \forall t_1, t_2.\ R(t_1) \wedge R(t_2) \wedge C_1(t_2) \wedge C_2(t_1, t_2) \Rightarrow C_1(t_1),}$$

then for all instances $r \in Sat(F)$:

$$\sigma_{C_1}(\omega_{C_2}(r)) = \omega_{C_2}(\sigma_{C_1}(r)).$$

6 Propagation of integrity constraints

How do we know whether a specific CGD holds in a relation? If this is a database relation, then the CGD may be enforced by the DBMS or the application. If the relation is computed, then we need to determine if the CGD is preserved in the expression defining the relation. [26, 27] characterize cases where functional and join dependencies hold in the results of relational algebra expressions.

We already know that the CGD d_1^C holds in the result of the winnow ω_C. Winnow returns a subset of a given relation, thus it preserves all the CGDs holding in the relation. The following theorem characterizes all the dependencies holding in the result of winnow.

Theorem 5 *Assume F is a set of CGDs, f a CGD over a schema R, and $\succ_C$ an irreflexive preference relation over R. Then $F \cup d_1^C \vdash f$ iff for every $r \in Sat(F)$, $\omega_C(r) \in Sat(f)$.*

Proof. Assume it is not the case that $F \cup d_1^C \vdash f$. Thus for some instance r_0, $r_0 \in Sat(F \cup d_1^C)$ but $r_0 \notin Sat(f)$. Then for all $t_1, t_2 \in r_0$, $t_i \sim_C t_j$, and thus $r_0 = \omega_C(r_0)$. Therefore, $\omega_C(r_0) \notin Sat(f)$. In the other direction, assume for some $r_0 \in Sat(F)$, $\omega_C(r_0) \notin Sat(f)$. Thus $r_1 = \omega_C(r_0)$ is the instance satisfying $F \cup d_1^C$ and violating f, which provides a counterexample to the entailment of f by $F \cup d_1^C$. ∎

Example 11 *Consider Example 4. Thus, the FD $ISBN \rightarrow Price$ holds in the result of ω_{C_1}, because it is entailed by the CGD $d_1^{C_1}$*

$$d_1^{C_1} : \ \forall i_1, v_1, p_1, i_2, v_2, p_2, i_3, v_3, p_3.$$
$$Book(i_1, v_1, p_1) \wedge Book(i_2, v_2, p_2) \Rightarrow (i_1 \neq i_2 \vee p_1 \geq p_2).$$

However, the FD might not hold in the input relation Book.

7 Computational complexity

Here we address the computational issues involved in checking the semantic properties essential for the semantic optimization of preference queries. We have shown that such properties can be formulated in terms of the entailment of CGDs. We assume that we are dealing with k-dependencies for some fixed $k \geq 1$. For example, for FDs $k = 2$. Notice also that all the interesting properties studied in this paper, e.g., containment or weak order, can be expressed as k-dependencies for $k \leq 3$.

We assume here that the CGDs under consideration are *clausal*: the constraint in the body is a conjunction of atomic constraints and the head consists of a disjunction of atomic constraints. All the dependencies that we have found useful in the context of semantic optimization of preference queries are clausal.

7.1 Upper bounds

[4] shows a reduction from the entailment of CGDs to the validity of universal formulas in the constraint theory. The basic idea is simple: the entailment of k-dependencies needs to be considered over relation instances of cardinality at most k, and each such instance can be represented by k tuple variables. Each dependency f is mapped to a constraint formula $cf_k(f)$. Then the entailment of a CGD f_0 by a set of CGDs F is expressed as the validity of the formula:

$$\forall^*.(\bigwedge_{f \in F} cf_k(f)) \Rightarrow cf_k(f_0),$$

or equivalently, as the unsatisfiability of a quantifier-free CNF formula obtained from its negation.

The following example illustrates the construction of $cf_k(f)$.

Example 12 *Consider the dependency:*

$$d_0^{C_1,C_2} : \ \forall t_1, t_2.\ R(t_1) \wedge R(t_2) \wedge t_1 \succ_{C_1} t_2 \Rightarrow t_1 \succ_{C_2} t_2.$$

We have that $cf_2(d_0^{C_1,C_2})$ *is equal to*

$$[C_1(t_1,t_1) \Rightarrow C_2(t_1,t_1)] \wedge [C_1(t_2,t_2) \Rightarrow C_2(t_2,t_2)]$$
$$\wedge [C_1(t_1,t_2) \Rightarrow C_2(t_1,t_2)] \wedge [C_1(t_2,t_1) \Rightarrow C_2(t_2,t_1)]$$

which for irreflexive $\succ_{C_1}$ *is equivalent to*

$$[C_1(t_1,t_2) \Rightarrow C_2(t_1,t_2)] \wedge [C_1(t_2,t_1) \Rightarrow C_2(t_2,t_1)].$$

For a fixed k, the size of $cf_k(f)$ is linear in the size of f. Thus we can easily characterize the complexity of dependency entailment.

Theorem 6 *Assume F is a set of k-CGDs for a fixed $k \geq 1$ over a constraint theory of equality/rational-order constraints, and preference relations are defined by ipfs over the same constraint theory. Checking containment, dependency propagation, and weak or strict partial order property, relative to F, are all in co-NP.*

Proof. Satisfiability of conjunctions of atomic constraints in this constraint theory can be checked in polynomial time [19]. Thus satisfiability of quantifier-free formulas in this constraint theory is in NP. ■

What remains now to be shown is that (1) the intractability is, in general, unavoidable; and (2) special tractable cases exist.

7.2 Lower bounds

[4] show a number of co-NP-completeness results for the entailment problem restricted to special classes of CGDs. To adopt those results to the context of the semantic properties of preference queries studied in the present paper, we need to show that the hardest (co-NP-hard) cases of the entailment can be equivalently expressed in terms of such properties. Such an approach is adopted in the proofs of Theorems 7 and 8 to characterize the complexity of testing redundancy of winnow and propagation of integrity constraints. On the other hand, in Theorem 9, a new reduction is introduced for the problem of testing the (relative) weak order property.

Theorem 7 *Checking whether ω_C is redundant relative to F, where F is a set of 2-CGDs and C is a rational-order ipf defining a strict partial order, is co-NP-hard.*

Proof. We adapt the proof of Theorem 4.3 in [4]. The reduction there is from SET SPLITTING but the same reduction applies to MONOTONE 3-SAT. Assume we are given a propositional formula ϕ with n variables $p_1, \ldots, p_n$, consisting of l positive clauses of the form $c_h \equiv p_i \vee p_j \vee p_m$, $h = 1, \ldots, l$, and k negative clauses $c_h \equiv \neg p_i \vee \neg p_j \vee \neg p_m$, $h = 1, \ldots, k$. We consider a relation R with $n + k + 1$ attributes. The truth of a propositional variable p_i is represented by equality on the attribute i. We build the set F of 2-CGDs in stages. A positive clause $p_i \vee p_j \vee p_m$ is mapped to a CGD

$$\forall t_1, t_2.\ R(t_1) \wedge R(t_2) \wedge t_1[i] \neq t_2[i] \wedge t_1[j] \neq t_2[j] \Rightarrow t_1[m] = t_2[m].$$

The construction for a negative clause $c_h \equiv \neg p_i \vee \neg p_j \vee \neg p_m$ is more compli-

cated. We construct the following FDs:

$$\forall t_1, t_2.\ R(t_1) \wedge R(t_2) \wedge t_1[i] = t_2[i] \wedge t_1[n+h] = t_2[n+h]$$
$$\Rightarrow t_1[n+k+1] = t_2[n+k+1],$$
$$\forall t_1, t_2.\ R(t_1) \wedge R(t_2) \wedge t_1[j] = t_2[j] \wedge t_1[m] = t_2[m] \Rightarrow t_1[n+h] = t_2[n+h].$$

Finally, we define the preference relation $\succ_C$:

$$t_1 \succ_C t_2 \equiv t_1[n+k+1] > t_2[n+k+1].$$

Thus the CGD d_1^C is

$$\forall t_1, t_2.\ R(t_1) \wedge R(t_2) \Rightarrow t_1[n+k+1] \leq t_2[n+k+1].$$

Along the same lines as in the proof in [4], we can show that ϕ is unsatisfiable iff $F \vdash d_1^C$. ■

Theorem 8 *Checking whether $F \cup d_1^C \vdash f$, where F is a set of 2-CGDs, f is a 2-CGD, and C is a rational-order ipf defining a strict partial order, is co-NP-hard.*

Proof. Essentially the same as that of Theorem 7. We use the same set of CGDs F and the CGD f is identical to the dependency d_1^C from that proof. On the other hand, we define C to be *False* and thus d_1^C is always satisfied and $F \cup d_1^C$ is equivalent to F. ■

Theorem 9 *Checking whether $\succ_C$ is a weak order relative to F, where F is a set of 3-CGDs and C is an equality ipf defining a strict partial order, is co-NP-hard.*

Proof. Reduction from 3-colorability. Assume we are given a graph $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$. We construct the set F consisting of the following CGDs:

$$\forall t.\ R(t) \Rightarrow t[i] = 0 \vee t[i] = 1,$$
$$\forall t.\ R(t) \Rightarrow t[n+1] = 1 \vee t[n+1] = 2 \vee t[n+1] = 3,$$
$$\forall t_1, t_2, t_3.\ R(t_1) \wedge R(t_2) \wedge R(t_3) \wedge t_1[n+1] \neq t_2[n+1]$$
$$\wedge t_1[n+1] \neq t_3[n+1] \wedge t_2[n+1] \neq t_3[n+1] \Rightarrow \gamma(t_1[i], t_2[i], t_3[i]),$$

where $i = 1, \ldots n$ and $\gamma(x, y, z)$ is a formula saying that exactly one of x, y and z is equal to 1. The last dependency is not clausal but can easily be represented as a set of clausal CGDs. Also, for every edge $(v_i, v_j) \in E$, we

include the following CGD:

$$\forall t_1, t_2, t_3.\ R(t_1) \wedge R(t_2) \wedge R(t_3) \wedge t_1[n+1] \neq t_2[n+1] \wedge t_1[n+1] \neq t_3[n+1]$$
$$\wedge t_2[n+1] \neq t_3[n+1] \Rightarrow t_1[i] \neq t_1[j] \vee t_2[i] \neq t_2[j] \vee t_3[i] \neq t_3[j].$$

Finally, we define the strict partial order $\succ_C$ as follows:

$$t \succ_C t' \equiv t[n+1] = 1 \wedge t'[n+1] = 2.$$

Assume now that G is 3-colorable. We construct an instance $r = \{t_1, t_2, t_3\}$ as follows. We will use t_k, $k = 1, \ldots, 3$, to represent the vertices colored with the color k. We make $t_k[i] = 1$ if v_i is colored with k; $t_k[i] = 0$ otherwise. We make $t_1[n+1] = 1$, $t_2[n+1] = 2$ and $t_3[n+1] = 3$. By construction, r satisfies F but $\succ_C$ on r is not a weak order. Therefore, $\succ_C$ is not a weak order relative to F.

In the other direction, take an instance $r = \{t_1, t_2, t_3\}$ satisfying F but such that $\succ_C$ on r is not a weak order. Such an instance has to have a subset $r_0 = \{t_1, t_2, t_3\}$ satisfying F and such that $\{t_1[n+1], t_2[n+1], t_3[n+1]\} = \{1, 2, 3\}$. Then r_0 encodes a 3-coloring for G. ■

7.3 Tractable cases

We obtain our first tractability results by identifying a new case of PTIME entailment. The case involves the entailment of a CGD over equality constraints by a set of FDs. This case was not studied in [4]. Note that it is more general than the standard FD entailment because the CGD may contain general equality constraints.

Theorem 10 *Let $F = \{f_1, \ldots, f_n\}$ be a set of FDs and f_0 a clausal k-CGD over equality constraints. Then checking whether $F \vdash f_0$ is in PTIME.*

Proof. The dependency f_0 is of the form

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

As explained earlier in this section, the entailment $F \vdash f_0$ reduces to the validity of the formula

$$\forall t_1, \ldots, t_k.(\bigwedge_{f \in F} cf_k(f)) \Rightarrow cf_k(f_0),$$

which is the same as the unsatisfiability of the formula

$$(\bigwedge_{f \in F} cf_k(f)) \wedge \neg cf_k(f_0).$$

We note that for any fd $f \equiv X \rightarrow Y$, $cf_k(f)$ is a conjunction E of implications

$$\bigwedge_{i,j=1,\ldots,k} t_i[X] = t_j[X] \Rightarrow t_i[Y] = t_j[Y].$$

On the other hand, $\neg cf_k(f_0)$ is a disjunction of conjunctions $S_1, \ldots, S_m$ of atomic equality constraints. Each S_i, $i = 1, \ldots, m$, is of the form $\phi_i(t_{i_1}, \ldots, t_{i_k}) \wedge \psi_i(t_{i_1}, \ldots, t_{i_k})$ where $i_1, \ldots, i_k \in \{1, \ldots, k\}$, $\phi_i(t_{i_1}, \ldots, t_{i_k})$ is a conjunction of equalities, and $\psi_i(t_{i_1}, \ldots, t_{i_k})$ is a conjunction of inequalities. We view $\phi(t_{i_1}, \ldots, t_{i_k})$ as a set of equalities $W_i^=$ and $\psi_i(t_{i_1}, \ldots, t_{i_k})$ as a set of equalities $W_i^{\neq}$. Both of those conjunctions can be viewed as sets of atomic constraints.

To determine the satisfiability of the formula $E \wedge (S_1 \vee \cdots \vee S_m)$, we need to check whether $E \wedge S_i$ is satisfiable for any $i = 1, \ldots, m$. This can be done in four steps:

(1) Deriving a set of equalities $E_i^=$ from $W_i^=$ using the set of implications E together with transitivity, symmetry and reflexivity. This is all Horn reasoning, and thus can be done in polynomial time and produces a polynomially-sized result.
(2) Checking if $E_i^=$ contains any internal contradiction, i.e., a formula $a = b$ where a and b are distinct constants. If so, $E \wedge S_i$ is unsatisfiable.
(3) Checking if $E_i^=$ contains any contradiction with $W_i^{\neq}$, i.e., a pair of terms (variables or constants) X and Y such that $X = Y \in E_i^=$ and $X \neq Y \in W_i^{\neq}$. If so, $E \wedge S_i$ is unsatisfiable.
(4) If no contradictions are discovered in the previous steps, then $E \wedge S_i$ is satisfiable.

The satisfiability of $E \wedge S_i$ can thus be determined in polynomial time. ■

We establish now a relationship between general and clausal CGDs.

Proposition 4 *Every CGD*

$$\forall t_1 \ldots \forall t_k.\ R(t_1) \wedge \cdots \wedge R(t_k) \wedge \gamma(t_1, \ldots t_k) \Rightarrow \gamma'(t_1, \ldots t_k),$$

where $\gamma(t_1, \ldots t_k)$ is a formula in DNF and $\gamma'(t_1, \ldots t_k)$ is a formula in CNF, is equivalent to polynomially many clausal CGDs.

Proof. This follows from two simple observations. First, the formula $A \wedge (B_1 \vee B_2) \Rightarrow C$ is equivalent to the formulas $A \wedge B_1 \Rightarrow C$ and $A \wedge B_2 \Rightarrow C$. Second,

the formula $A \wedge B \Rightarrow C_1 \wedge C_2$ is equivalent to the formulas $A \wedge B \Rightarrow C_1$ and $A \wedge B \Rightarrow C_2$. ■

Corollary 1 *Given a set of FDs F and equality ipfs C_1 (in DNF) and C_2 (in CNF), the the containment of $\succ_{C_1}$ in $\succ_{C_2}$ relative to F can be checked in PTIME.*

Proof. The containment can be expressed as the entailment $F \vdash d_0^{C_1,C_2}$. By Proposition 4, the dependency $d_0^{C_1,C_2}$ is equivalent to polynomially many clausal CGDs (under the given assumptions). By Theorem 10, the entailment can thus be checked in PTIME. ■

To obtain further tractability results we need to consider the structure of preference formulas in more detail.

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*$(C) = \max\{k_1, \ldots, k_m\}$.

Corollary 2 *Given a set of FDs F and an equality ipf C in DNF, checking whether $\succ_C$ is a strict partial order (or a weak order) relative to F can be done in PTIME if width$(C) \leq 1$ or span$(C) \leq 1$.*

Proof. These properties can be expressed in terms of entailment of polynomially many clausal CGDs by F and the result follows by Proposition 4. For example, consider strict partial orders. The CGDs in question are

$$\forall t.\ R(t) \Rightarrow \neg C(t,t)$$

and

$$\forall t_1, t_2, t_3.\ R(t_1) \wedge R(t_2) \wedge R(t_3) \wedge C(t_1,t_2) \wedge C(t_2,t_3) \Rightarrow C(t_1,t_3).$$

If *width*$(C) \leq 1$ or *span*$(C) \leq 1$, then $\neg C(t,t)$ is a formula in CNF of size linear in the size of C. Under the same assumptions, the formula $C(t_1,t_3)$ is in CNF, and the formula $C(t_1,t_2) \wedge C(t_2,t_3)$ is in DNF or can be transformed into DNF with at most polynomial increase in size. ■

We obtain here further tractable cases of the semantic properties studied in the present paper by adapting the results of [4]. That paper identifies several classes of CGDs for which the entailment problem is tractable.

The restrictions we impose on CGDs and preference formulas may be of the following kinds:

- the atomic constraints should be *typed*;
- the number of atomic constraints should be bounded.

Definition 10 *A constraint formula $C(t_1, \ldots, t_n)$ over tuple variables $t_1, \ldots, t_n$ is* typed *if all its atomic subformulas are of the form $t_i[A]\theta t_j[A]$ or $t_i[A]\theta c$, where A is an attribute, c a constant, and $\theta \in \{=, \neq, <, >, \leq, \geq\}$. A CGD is typed if all its constraints are typed.*

Consider first the containment problem. To check whether $\succ_{C_1} \subseteq_F \succ_{C_2}$, we need to determine whether $F \vdash d_0^{C_1,C_2}$ where

$$d_0^{C_1,C_2} : \ \forall t_1, t_2.\ R(t_1) \wedge R(t_2) \wedge t_1 \succ_{C_1} t_2 \Rightarrow t_1 \succ_{C_2} t_2.$$

To obtain tractability we need to impose simultaneous restrictions on F, $\succ_{C_1}$, and $\succ_{C_2}$.

Theorem 11 *Let F be a set of typed clausal 2-CGDs with two atomic constraints over a schema R, and C_1 and C_2 typed preference formulas over the same schema and the same constraint theory (either equality or rational order). Moreover, none of F, C_1, and C_2 contains constants. Then*

- *checking whether $\succ_{C_1} \subseteq_F \succ_{C_2}$ can be done in PTIME if $span(C_1) \leq 1$ and $width(C_2) \leq 1$, and*
- *checking whether $\succ_{C_1} \subseteq_F \succ_{False}$ can be done in PTIME if $span(C_1) \leq 2$.*

Proof. Under the given assumptions, the dependency $d_0^{C_1,C_2}$ is equivalent to polynomially many typed clausal 2-CGDs with at most two atomic constraints over the same constraint theory. For such CGDs, the entailment is in PTIME [4, Theorem 4.2]. ∎

Note that, for example, unary FDs are typed 2-CGDs with two atomic equality constraints.

Consider now the problem of propagating integrity constraints.

Theorem 12 *Let F be a set of clausal k-CGDs, f a clausal k-CGD and C a preference formula over the same schema, and none of F, f, and C contains constants. Then checking whether $F \cup d_1^C \vdash f$ can be done in PTIME if:*

- *F, f, and $\neg C$ have at most one atomic constraint each, or*
- *$k = 2$, and F, f and $\neg C$ are typed and contain each at most two atomic constraints over the same constraint theory (either equality or rational order).*

Proof. In the first case, the dependency d_1^C contains one constraint. Thus [4, Theorem 4.1] implies that $F \cup d_1^C \vdash f$ is in PTIME. In the second case, [4, Theorem 4.2] applies, yielding the same result. ∎

The results of [4] cannot be applied to identify tractable cases of the weak order or the strict partial order property, because those properties are formulated using 3-CGDs with three or more atomic constraints. Such CGDs do not fall into any of the tractable classes of [4].

8 Related work

The basic reference for semantic query optimization is [7]. The most common techniques are: join elimination/introduction, predicate elimination and introduction, and detecting an empty answer set. [8] discusses the implementation of predicate introduction and join elimination in an industrial query optimizer. Semantic query optimization techniques for relational queries are studied in [37] in the context of denial and referential constraints, and in [32] 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 [35].

The literature on preference queries and related approaches was extensively discussed in [10]. We provide only a brief discussion here.

Two different approaches to preference queries have been pursued in the literature: qualitative and quantitative. In the *qualitative* approach, represented by [31, 22, 29, 6, 18, 9, 10, 21, 23, 25], 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 [1, 20], 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 13 *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.*

Example 14 *Another common example of a preference relation that is not representable using a utility function is the* threshold of detectable difference

relation $\succ_t$*:*

$$x \succ_t y \equiv x \geq y + c$$

where c *is the threshold value* ($c > 0$)*.*

This lack of expressiveness of the quantitative approach is well known in utility theory [15, 14]. 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 [14]). However, even if a utility function is known to exist, its definition may be non-explicit [14] and thus unusable in the context of database queries. 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. In such cases, we can exploit the benefits of a preference relation being a (relative) weak order, for example the possibility of using WWO for computing winnow, without a need to construct a specific utility function representing the preference relation. Algebraic optimization of preference queries is discussed in the papers [10, 23, 24].

A logical approach to preferences, similar to the one proposed in [10] and elaborated in the present paper, was first studied in the context of deductive databases in [22, 29] and – independently – in [17, 18]. All of those papers propose extending Datalog with clausally-defined preference relations. The cited papers present declarative and operational semantics for the proposed Datalog extensions. The operational semantics extends the standard bottom-up [29, 18] or top-down [17] evaluation method for logic programs. In the context of database queries, the approach proposed in the present paper achieves similar goals to that of [22, 29, 17, 18], remaining, however, entirely *within the relational data model and SQL*, and not requiring a specialized query evaluation engine. Moreover, [22, 29, 17, 18] do not consider integrity constraints and semantic query optimization.

A quantitative approach to preferences in constraint logic programs is presented in [33]. The programs are extended with optimization goals that return solutions minimizing the value of a given expression. In the spirit of constraint logic programming, the solutions can be non-ground. Constraint optimization is also possible in the framework of semiring-based constraint logic programming [5]. Those approaches can in principle deal with arbitrary partial orders by building such orders into the language. They cannot, however, handle multiple, user-defined preference relations that are arbitrary partial orders. For preference queries the ability to use multiple preference relations in a single query is essential [10, 21].

9 Conclusions and further work

We have presented several novel techniques for semantic optimization of preference queries, focusing on the winnow operator. We characterized the necessary and sufficient conditions for the applications of those techniques in terms of the entailment of constraint-generating dependencies. (This idea was suggested but not fully developed in [11].) As a consequence, we were able to leverage some of the computational complexity results from [4]. Moreover, we proved here several new complexity results: Theorems 9 and 10. Theorem 3 is also completely new. Other results are reformulations of those presented in [11].

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 [9, 10], but also to preference queries defined using preference constructors [21, 25] and skyline queries [6, 13, 28, 34] because such queries can be expressed using preference formulas.

The ideas presented in this paper in the context of winnow can be adapted to other preference-related operators. For example, *ranking* [10] associates with each tuple in a relation its rank. The best tuples (computed by winnow) have rank 1, the second-best tuples have rank 2, etc. The algorithm WWO can be extended to compute ranking instead of winnow, and thus the computation of ranking will also benefit if the given preference relation is a weak order relative to the given integrity constraints.

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, e.g., tuple-generating dependencies and referential integrity constraints,
- deriving further tractable cases of (relative) containment and satisfaction of order axioms,
- studying the preservation of general constraint-generating dependencies by relation algebra operators and expressions ([27] consider this problem for functional and join dependencies);
- identifying weaker but easier to check sufficient conditions for the application of our techniques.

References

[1] R. Agrawal and E. L. Wimmers. A Framework for Expressing and Combining Preferences. In *ACM SIGMOD International Conference on Man-*

agement 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] 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.

[5] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-Based Constraint Logic Programming: Syntax and Semantics. *ACM Transactions on Programming Languages and Systems*, 23(1):1–29, 2001.

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

[7] 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.

[8] 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.

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

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

[11] J. Chomicki. Semantic Optimization of Preference Queries. In *International Symposium on Constraint Databases*, pages 133–148, Paris, France, June 2004. Springer-Verlag, LNCS 3074.

[12] J. Chomicki. Iterative Modification and Incremental Evaluation of Preference Queries. In *International Symposium on Foundations of Information and Knowledge Systems (FOIKS)*, pages 63–82. Springer, LNCS 3861, 2006.

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

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

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

[16] P. Godfrey, R. Shipley, and J. Gryz. Maximal Vector Computation in Large Data Sets. In *International Conference on Very Large Data Bases (VLDB)*, pages 229–240, 2005.

[17] K. Govindarajan, B. Jayaraman, and S. Mantha. Preference Logic Pro-

gramming. In *International Conference on Logic Programming (ICLP)*, pages 731–745, 1995.

[18] K. Govindarajan, B. Jayaraman, and S. Mantha. Preference Queries in Deductive Databases. *New Generation Computing*, 19(1):57–86, 2000.

[19] 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.

[20] V. Hristidis and Y. Papakonstantinou. Algorithms and Applications for Answering Ranked Queries using Ranked Views. *VLDB Journal*, 13(1):49–70, 2004.

[21] W. Kießling. Foundations of Preferences in Database Systems. In *International Conference on Very Large Data Bases (VLDB)*, pages 311–322, 2002.

[22] 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.

[23] 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.

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

[25] W. Kießling and G. Köstler. Preference SQL - Design, Implementation, Experience. In *International Conference on Very Large Data Bases (VLDB)*, pages 990–1001, 2002.

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

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

[28] 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)*, pages 275–286, 2002.

[29] 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.

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

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

[32] 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.

[33] K. Marriott and P. J. Stuckey. Semantics of Constraint Logic Programs with Optimization. *ACM Letters on Programming Languages and Systems*, 2(1-4):197–212, 1993.

[34] 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.

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

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

[37] 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.