Solving Satisfiability and Implication Problems in Database Systems

SHA GUO, WEI SUN, and MARK A. WEISS
Florida International University

Satisfiability, implication, and equivalence problems involving conjunctive inequalities are important and widely encountered database problems that need to be efficiently and effectively processed. In this article we consider two popular types of arithmetic inequalities, (X **op** Y) and (X **op** C), where X and Y are attributes, C is a constant of the domain or X, and **op** $\in \{<, \leq, =, \neq, >, \geq\}$. These inequalities are most frequently used in a database system, inasmuch as the former type of inequality represents a θ—join, and the latter is a selection. We study the satisfiability and implication problems under the integer domain and the real domain, as well as under two different operator sets ($\{<, \leq, =, \geq, >\}$ and $\{<, \leq, =, \neq, \geq, >\}$). Our results show that solutions under different domains and/or different operator sets are quite different. Out of these eight cases, excluding two cases that had been shown to be NP-hard, we either report the first necessary and sufficient conditions for these problems as well as their efficient algorithms with complexity analysis (for four cases), or provide an improved algorithm (for two cases). These iff conditions and algorithms are essential to database designers, practitioners, and researchers. These algorithms have been implemented and an experimental study comparing the proposed algorithms and those previously known is conducted. Our experiments show that the proposed algorithms are more efficient than previously known algorithms even for small input.

Categories and Subject Descriptors: H.2.4 [**Database Management**]: Systems—*query processing*; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*deduction*; I.1.2. [**Algebraic Manipulation**]: Algorithms—*analysis of algorithms*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*complexity of proof procedure*

General Terms: Algorithms, Theory, Language

Additional Key Words and Phrases: Deduction, reasoning, satisfiabilty, implication, equivalence

1. INTRODUCTION

Efficiently and effectively solving the satisfiability problem (whether there exists a contradiction in a formula consisting of conjunctive inequalities)

This research is supported in part by NASA under NAGW-4080, USDA/FL DOE under 131-30440-40001 and 131-30450-50041, Florida High Technology and Industrial Council, and FIU Foundation.
Authors' addresses: School of Computer Science, Florida International University, Miami, FL 33199; Please address all correspondence to: W. Sun, email: ⟨weisun@cs.fiu.edu⟩.

and the implication problem (whether one formula *implies* another formula) is central to many database problems. Examples of the database application areas involving such problems are distributed query processing where relations are horizontally fragmented based on selections [Yu and Chang 1984; Ceri and Pelagatti 1984; Meghini and Thanos 1991; Shasha and Wang 1991; Ullman 1989], global query optimization (optimizing a group of queries by identifying common subexpressions) [Jarke and Koch 1984; Kim 1984; Sellis 1986], enforcing inference or data consistency (testing the applicability of rules, constraints, or triggers) [Chakravarthy et al. 1990; King 1981; Sun and Yu 1994; Yu and Sun 1989], updating databases through views (computing queries from derived relations and views) [Astrahan et al. 1976; Blakeley et al. 1986a, 1986b], and evaluating queries in deductive database or logic-based systems [Ullman 1989], among many others.

The following definitions formally define the problems.

Definition 1.1 An *inequality* is of the form either (X **op** Y) or (X **op** C), where C is a constant of the domain of X, X and Y are attributes/variables of the integer domain or the real domain, and **op** $\in \{<, \leq, =, >, \geq, \neq\}$. An *unequality* is an inequality involving the operator $\neq$.

These two basic types of inequalities are quite representative in relational and deductive database systems. In fact, the inequalities of the first type specify θ—joins, and the inequalities of the second type are selections.

Definition 1.2 A *formula* can be recursively defined as follows: *true* and *false* are formulae; an inequality is a formula; $A \wedge B$ (i.e., A and B) is a formula if A and B are formulae; $A \vee B$ (i.e., A or B) is a formula if A and B are formulae; $\neg A$ (i.e., the *negation* of A) is a formula if A is a formula; (A) is a formula if A is a formula; nothing else is a formula. "$\vee$" and "$\wedge$" are called *connectives*. A formula is *normalized* if it contains neither parentheses nor negations.

It is clear that a formula can always be equivalently transformed into normalized form. In this study we therefore assume a formula is normalized without losing generality.

Definition 1.3 $\{C_1/X_1, C_2/X_2, \ldots, C_n/X_n\}$ is said to be an *assignment* for a formula if every occurrence of X_i in the formula is simultaneously replaced by C_i, $1 \leq i \leq n$.

An assignment *satisfies* a formula if and only if the formula evaluates *true* under the assigned values. There exists a *contradiction* in a formula if and only if there does not exist an assignment that satisfies the formula. In the latter case, we also say the formula is *unsatisfiable*.

Example 1.1 The formula $((X < 9) \wedge (X > 3) \wedge (Y = 1))$ is satisfiable, because the assignment $\{4/X, 1/Y\}$ satisfies the formula; the formula $((X < 3) \wedge (X > 1) \wedge (X \neq 2))$ is satisfiable if the domain is real, but unsatisfiable if the domain of X is integer; and the formula $((X < 3) \vee (X > 1))$ is a tautology, because it evaluates *true* under any assignment.

In this article we only focus on conjunctive formulae. Solutions to problems involving disjunctive formulae are generally more difficult. Throughout this study, let S and T be two conjunctive formulae. S and T are also considered to be two sets of inequalities in order to simplify our presentation, where conjunctions among set members (inequalities) are assumed. There are three problems to be studied: *implication*, *satisfiability*, and *equivalence* as defined in the following.

Definition 1.4 *Implication.* *S implies T*, denoted as $S \Rightarrow T$, if and only if every assignment that satisfies S also satisfies T.

Satisfiability/Contradiction. S is *satisfiable* if and only if there exists at least one assignment for S that satisfies S. Otherwise, there is a *contradiction* in S.

Equivalence. S and T are *equivalent* if and only if S implies T and T implies S.

Because the equivalence between S and T is defined on the basis of implication between S and T, it is sufficient for us to study only the implication and satisfiability problems. We are interested in identifying the most efficient algorithms for these problems. We study these problems under different restrictions, and the following are the major types of restrictions that can be placed:

Operator Set. Operators of inequalities can be put into two different groups: $OP_{\neg\neq} \overset{\text{def}}{\equiv} \{<, \leq, =, \geq, >\}$ or $OP_{all} \overset{\text{def}}{\equiv} \{<, \leq, =, \neq, \geq, >\}$, where $\overset{\text{def}}{\equiv}$ stands for "defined as."

Problems with inequalities involving $OP_{\neg\neq}$ are likely to be solved more efficiently, whereas problems with inequalities involving OP_{all} are usually more difficult to solve. The intuition as to why there exists such a difference is that $\neq$ is basically a sort of negation, whereas in logic, negations introduce complexity of solutions. By applying the following equivalence transformations, the operator set can be further reduced.

$$(X = C) \equiv (X \leq C) \wedge (X \geq C); \qquad (X < C) \equiv (X \leq C) \wedge (X \neq C);$$

$$(X > C) \equiv (X \geq C) \wedge (X \neq C); \qquad (X = Y) \equiv (X \leq Y) \wedge (X \geq Y);$$

$$(X < Y) \equiv (X \leq Y) \wedge (X \neq Y); \qquad (X > Y) \equiv (X \geq Y) \wedge (X \neq Y).$$

Due to the conjunction of inequalities, $OP_{\neg\neq}$ and OP_{all} can be equivalently reduced to: $OP_{\neg\neq} \overset{\text{def}}{\equiv} \{<, \leq, \geq, >\}$, and $OP_{all} \overset{\text{def}}{\equiv} \{\leq, \neq, \geq\}$. The equivalence is in the sense that applying these transformations does not change the size of a formula in big-O notation.

Variables/Constants Domains. We assume that all variables and constants will have the same type of domains (either integer or real). The general satisfiability and implication problems in the integer domain have been shown to be NP-hard [Rosenkrantz and Hunt 1980]. However, the study of these problems under the real domain has not been addressed in the literature. As we found in this study, solutions to the problems under the real domain differ significantly from those under the integer domain.

In this article, we study "Is S satisfiable?" and "Does S imply T?" We assume that any variable in T, say X, is also included in the variables of S, unless X appears in inequalities (X **op** X) $\in T$ only (in this case, the inequality must be either a tautology that can be removed or a contradiction that makes the formula unsatisfiable). The sizes of S and T (the number of inequalities of S and T, denoted as $|S|$ and $|T|$, respectively) are used as the basic measurement of the problem size. We also assume that accessing a variable only takes constant time. The similar assumption has been widely adopted in analyses of algorithms.[1] In addressing the implication problem, we assume that the same restriction or assumption is applied to both S and T simultaneously. We elect not to discuss situations where S and T have different restrictions (because it is rather unlikely that such a situation exists in practice) and the explosion of the combinations of different situations.

The rest of this article is organized as follows: in Section 2 a comparison with previous results is made after we briefly review the previous work. Our contributions are pointed out. The relationship among algorithms for solving satisfiability and implications problems is also provided. Section 3 discusses the satisfiability problem, and Section 4 addresses the implication problem. In Section 5, we provide a C++ implementation and experimental results. Finally, we conclude this study in Section 6.

2. RELATED WORK AND OUR CONTRIBUTIONS

Because solving the satisfiability and implication problems is central to database systems, these problems have received fairly intensive studies.[2]

Rosenkrantz and Hunt [1980] discuss the satisfiability and equivalence of conjunctive mixed predicates containing inequalities of the form (X **op** C), (X **op** Y), or (X **op** $Y + C$), where only the integer domain is assumed. In that paper, the general satisfiability and implication problems (**op** is OP_{all}) are shown to be NP-hard in the integer domain. An $O(|S|^3)$ algorithm is also given for the restrictive satisfiability and implications problems where "$\neq$" is not allowed (**op** is $OP_{\neg\neq}$) in inequalities. Later, Sun et al. [1989] discuss the same implication problems (Does S imply T), and extend the results by providing an algorithm with the same complexity to solve the problem even if "$\neq$" is allowed in inequalities in T (but not in S).

[1]Please see Aho et al. [1983], Cormen et al. [1990], Tarjan [1972, 1983], and Weiss [1995].
[2]Please see Aho et al. [1979a, 1979b], Johnson and Klug [1983, 1984], Klug [1988], Rosenkrantz and Hunt [1980], Sun et al. [1989], Sun and Weiss [1994], and Ullman [1989].

Table I. Comparison of Our Results with Previous Best Known Results

Problems	Domains	Operators	Our Results	Previous Results
Is S Satisfiable (satisfiability problem)	integer	$\mathbf{OP}_{\neg\neq}$	|S|(*Sec.* 3.1)	$\|S\|^3$ [Rosenkrantz and Hunt 1980]
		$\mathbf{OP}_{all}$		NP-hard [Rosenkrantz and Hunt 1980]
	real	$\mathbf{OP}_{\neg\neq}$	$\|S\|$(*Sec.* 3.1)	
		$\mathbf{OP}_{all}$	$\|S\|$(*Sec.* 3.2)	
Does S Imply T (implication problem)	integer	$\mathbf{OP}_{\neg\neq}$	$\|S\|^2 + \|T\|$ (*Sec.* 4.1)	$\|S\|^3 + \|T\|$ [Sun et al. 1989]
		$\mathbf{OP}_{all}$		NP-hard [Sun et al. 1989]
	real	$\mathbf{OP}_{\neg\neq}$	$\|S\|^2 + \|T\|$ (*Sec.* 4.1)	
		$\mathbf{OP}_{all}$	$\min(\|S\|^{2.376} + \|T\|, \|S\|*\|T\|)$ (*Sec.* 4.2)	

The restrictive implication of conjunctive queries involving inequalities of the form (X $\mathbf{OP}_{\neg\neq}$ Y) where both integer and real domains are considered has been studied in Aho et al. [1979a, 1979b], and Johnson and Klug [1983, 1984]. Klug [1988] generalizes the implication problem by including inequalities of the form (X $\mathbf{OP}_{\neg\neq}$ C) as well as (X $\mathbf{OP}_{\neg\neq}$ Y), whereas it only assumes the dense totally ordered domains (e.g., the real domain) and does not allow the integer domain.

Ullman [1989] gives an algorithm with the complexity of $O(|S|^3 + |T|)$ for the implication of conjunctive predicates containing only inequalities of the form (X $\mathbf{OP_{all}}$ Y) for both integer and real domains (selections are left out). Sun and Weiss [1994] provide an improved algorithm with the complexity $O(|S|^{2.376} + |T|)$ for this situation.

In this article we study the satisfiability and implication problems involving inequalities of the form either (X **op** Y) or (X **op** C) under both integer and real domains and under $\mathbf{OP}_{\neg\neq}$ and $\mathbf{OP}_{all}$, respectively. As shown in Table I, we have either reported the first efficient algorithms or improved the previous best known algorithms for various situations (implication versus satisfiability problem, integer versus real domain, $\mathbf{OP}_{\neg\neq}$ versus $\mathbf{OP}_{all}$). More precisely, the following contributions are significant to this study (also see Table I):

—We report the first linear algorithm for the general satisfiability in the real domain involving $\mathbf{OP_{all}}$.

—We improve the previous best known $O(|S|^3)$ [Rosenkrantz and Hunt 1980; Sun et al. 1989] to $O(|S|)$ for the restricted satisfiability in the integer domain involving $\mathbf{OP}_{\neg\neq}$.

—We improve the previous best known $O(|S|^3 + |T|)$ [Rosenkrantz and Hunt 1980; Sun et al. 1989] to $O(|S|^2 + |T|)$ for the restricted implication in the integer domain involving $\mathbf{OP}_{\neg\neq}$.

—We report the first $O(|S|^2 + |T|)$ algorithm for the restricted implication in the real domain.

—An $O(|S|^{2.376} + |T|)$ algorithm is proposed to solve the general implication in the real domain involving $\mathbf{OP_{all}}$. Our algorithm extends the previous best known $O(|S|^{2.376} + |T|)$ algorithm [Sun and Weiss 1994] which only allow the inequalities of the form $(X \ \mathbf{op} \ Y)$.

As a summary, Table I gives the results and comparisons with previous best known results.

Before we proceed to detailed discussions, we first present the following two theorems that relate time complexity bounds of algorithms for satisfiability and implication problems.

THEOREM 2.1 *If the time complexity for solving the implication "Does S imply T?" is bounded by $O(f(n, |S|, |T|))$, then the time complexity for solving the satisfiability "Is S satisfiable?" is bounded by $O(f(n, |S|, 1))$, where n is the number of distinct variables used in S and T, and f is a function.*

Theorem 2.1 merely states that the algorithm that solves an implication problem can be used to solve the satisfiability problem. The following theorem states that the algorithm used to solve the satisfiability problem also can be used to solve the implication problem.

THEOREM 2.2. *Let S and T be two formulae involving conjunctive inequalities. S implies T if and only if for each inequality $t \in T$, $S \wedge \neg t$ is unsatisfiable. If the time complexity for solving the satisfiability* "Is S satisfiable?" *is bounded by $O(f(n, |S|))$, then the time complexity for solving the implication "Does S imply T?" is bounded by $O(|T| \cdot f(n, |S|))$.*

This theorem follows from the fact that S implies T if and only if $S \wedge \neg T = (S \wedge \neg t_1) \vee \ldots \vee (S \wedge \neg t_{|T|})$ is unsatisfiable.

We have implemented all these algorithms, and conducted a comparative study in Section 5. The C++ programs for the algorithms can be obtained by anonymous ftp from ⟨archive.fiu.edu⟩ under ⟨weisun⟩ directory, which may be of assistance to database designers, practitioners, and researchers.

3. IS S SATISFIABLE?

We first discuss the restricted satisfiability problem (i.e., $\mathbf{OP}_{\neg\neq}$ is used for inequalities). In Section 3.2 the general satisfiability problem is discussed (i.e., $\mathbf{OP_{all}}$ is used for inequalities).

3.1 The Restricted Satisfiability Problem

We first consider the integer domain. Then we extend to the real domain.

With the integer domain, it is sufficient to consider $\mathbf{op} \in \{<, \leq\}$ for inequalities of the type $(X \ \mathbf{op} \ Y) \in S$ because of symmetry, and $\mathbf{op} \in \{\leq, \geq\}$ for inequalities of the type $(X \ \mathbf{op} \ C)$, because $(X < C)$ is equivalent to $(X \leq C - 1)$ and $(X > C)$ is equivalent to $(X \geq C + 1)$.

Definition 3.1.1 A *labeled directed graph* for an inequality set S, denoted as $G_S = (V_S, E_S)$, is a *directed graph* with each node X in V_S

one-to-one corresponding to a distinct variable X in S, and each edge from node X to node Y and labeled with $\otimes$, denoted as $(X, Y, \otimes) \in E_S$, one-to-one corresponding to an inequality $(X \otimes Y) \in S$, where the label $\otimes$ is either $<$ or $\leq$.

G_S captures some important properties that are useful in the construction of our algorithms. For any two nodes X and Y in G_S, if there exists a directed path from X to Y, then, clearly, $(X < Y)$ or $(X \leq Y)$ is implied by S. The "$<$" or "$\leq$" is determined by the labels of the edge(s) in the path. Furthermore, if $(X \geq C) \in S$, then either $(Y > C)$ or $(Y \geq C)$ is implied by S; if $(Y \leq C) \in S$, then either $(X < C)$ or $(X \leq C)$ is implied by S. These observations are used later to decide if S is satisfiable.

Based on G_S, we can also deduce if $(X = Y)$ for two distinct nodes X and Y. It is clear that if X and Y are reachable via paths from each other, $(X = Y)$ is implied by S by transitivity. We say that all such variables as well as the edges among them form a *strongly connected component*, or *SCC*. Clearly, if an edge labeled with "$<$" is found in an *SCC*, S is unsatisfiable. Otherwise, we "collapse" each *SCC* into a single node. This collapsing is in the fact the use of a single variable in the *SCC*, called the *representative variable/node* of the *SCC*, to represent all variables in the *SCC*. The purpose of the collapsing is to obtain an acyclic graph for an easier subsequent computation and presentation.

Let $G_{S_{collapsed}}$ be the acyclic graph after the preceding collapsing, and $S_{collapsed}$ be the set of inequalities whose variables are replaced by the representative variables of *SCC*s. Clearly, S and $S_{collapsed}$ are equivalent.

For all $(X \leq C_i)$ and $(X \geq C'_i) \in S_{collapsed}$, let $C^X_{up} = \min(C_i)$ and $C^X_{low} = \max(C'_i)$ for variable X. The closed range $[C^X_{low}, C^X_{up}]$, called the (*closed*) *minimum range* for X, indicates the domain range qualified by the set of inequalities of the form $(X\{\leq, \geq\}C)$ for variable X. However, this range is not the "real" minimum range due to the restrictions imposed by other inequalities. For example, given $\{(X \leq 6), (X \leq Y), (Y \leq 5), (Y \geq 2), (Y \geq 0)\}$, the minimum ranges for X and Y are $(-\infty, 6]$ and $[2, 5]$, respectively. Because $(X \leq Y) \in S$, the "real" minimum range for X should be $(-\infty, 5]$.

So we should further compute the *"real" minimum range* for each variable X, denoted as $[A^X_{low}, A^X_{up}]$. We construct the "real" range for each variable X in $S_{collapsed}$ as follows: X is selected one by one according to its topological ordering in $G_{S_{collapsed}}$. $A^X_{low} = \max(C_i, C^X_{low})$ for all C_i, where $C_i = A^{X_i}_{low}$ is the assigned value for X's parent X_i[3] if the edge from X_i to X is labeled with "$\leq$", or $C_i = A^{X_i}_{low} + 1$ if the edge is labeled with "$<$." Then we select X one by one according to its *inverse* topological ordering in $G_{S_{collapsed}}$. $A^X_{up} = \min(C_j, C^X_{up})$ for all C_j, where C_j is equal to $A^{X_i}_{up}$, the assigned value for X's child X_j if the edge from X to X_j is labeled with "$\leq$,"

[3]A node X is called a *parent* of node Y if there exists a directed edge from X to Y, and Y is called a *child* of X.

or $C_j = A_{up}^{X_j} - 1$ if the edge is labeled with "<." Example 3.1.1 helps explain the procedure.

From the construction of A_{low}^X and A_{up}^X, we know that they have the property of minimality in the sense that given any assignment that satisfies S, the assigned value for any X, say A_X in the assignment, falls in $[A_{low}^X, A_{up}^X]$. Note that $C_{low}^X \le A_{low}^X$ and $A_{up}^X \le C_{up}^X$. We use $°A_{\min}$ and $°A_{\max}$ to denote two *"minimum" assignments* in which X is assigned A_{low}^X and A_{up}^X, respectively.

LEMMA 3.1.1 *In the integer domain involving* $\mathbf{OP}_{\neg\neq}$, *a conjunctive formula S is satisfiable if and only if*

(1) *every SCC in* G_S *does not contain an edge labeled* "<", *and*
(2) *for each variable X in* $S_{collapsed}$, $A_{low}^X \le A_{up}^X$.

PROOF. Condition (1) is obvious. Thus we focus on Condition (2).

Necessity: Let $°A_S$ be a satisfiable assignment for S, and the assigned value for any variable X in $°A_S$ is C_X. C_X must fall in the "real" minimum range of X, namely, $A_{low}^X \le C_X \le A_{up}^X$.

Sufficiency: We only need to prove that $°A_{min}$ or $°A_{max}$ satisfies S. Because $C_{low}^X \le A_{low}^X \le A_{up}^X \le C_{up}^X$, each $(X\{\le, \ge\}C) \in S_{collapsed}$ is satisfied by $°A_{\min}$. For any $(X < Y) \in S$, there exists an edge in $G_{S_{collapsed}}$ labeled with "<" from X to Y (X is a *parent* of Y). Because $A_{low}^Y = \max(A_{low}^{Y_i}+1, C_{low}^Y)$ for all parents Y_i of Y, we have $A_{low}^Y > A_{low}^X$. In other words, $(X < Y)$ is satisfied by $°A_{\min}$. Similar reasoning applies for $(X \le Y) \in S$. As a result, all inequalities in S are satisfied by $°A_{\min}$. □

The algorithm to determine if S is satisfiable is given in the following steps.

Algorithm 1

Step 1: Perform the transformation on each inequality in S such that only $(X\{<, \le\}Y)$ and $(X\{\le, \ge\}C)$ exist in S. If $(X \ne Y)$ or $(X \ne C)$ is found, report NP-hard problem and exit. At the same time, eliminate the trivial inequalities, namely, $(X\ \mathbf{op}\ X)$ and $(C_1\ \mathbf{op}\ C_2)$ in S. If $(X\{<, >, \ne\}X)$, or $(C\{<, >, \ne\}C)$, or $(C_1\{<, \le, =\}C_2)$ with $C_1 > C_2$, or $(C_1\{>, \ge, =\}C_2)$ with $C_1 < C_2$, then report that S is unsatisfiable, and exit.

Step 2: Construct the minimum range $[C_{low}^X, C_{up}^X]$ for each X by scanning all $(X\{\le, \ge\}C)$.

Step 3: Construct the labeled directed graph G_S; detect all *SCC*s. If any "<" is found in any *SCC*, then S is unsatisfiable; exit. Otherwise, collapse *SCC*s and obtain an acyclic graph $G_{S_{collapsed}}$.

Step 4: Topologically sort all nodes of the graph. Then the "real" minimum ranges $[A_{low}^X, A_{up}^X]$ are computed.

Step 5: If any $A_{low}^X > A_{up}^X$, S is unsatisfiable. Otherwise, report that S is satisfiable.

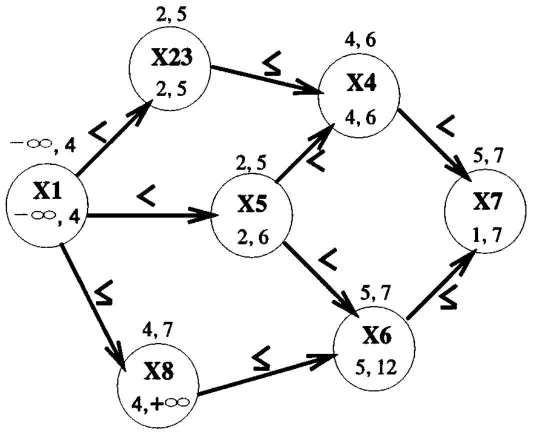

Fig. 1. Construction of assignment satisfying G_S in the integer domain.

THEOREM 3.1.1. *In the integer domain involving* $\mathbf{OP}_{\neg\neq}$, the satisfiability problem can be solved in $O(|S|)$ time.

PROOF. Steps 1 and 2 can be done in one scan of each inequality in S; *SCC*s can be found in two depth-first-searches of G_S. Finding the topological ordering of all variables of $G_{S_{collapsed}}$ takes $O(|S|)$ time. Finding all the "real" minimum ranges requires two traversals of $G_{S_{collapsed}}$ (one in topological order and one in inverse-topological order of nodes) and the cost is $O(|S|)$. Testing if all $A^X_{low} \leq A^X_{up}$ takes $O(|S|)$ time. □

Example 3.1.1. Consider the set of inequalities S:

$$\begin{array}{llllll} S = \{ & X_1 < X_2, & X_1 < X_5, & X_1 \leq X_8, & X_1 < 5, \\ & X_2 \leq X_3, & X_2 \leq 5, & X_2 > 1, & \\ & X_3 \leq X_2, & X_3 \leq X_4, & X_3 \leq 7, & X_3 \geq 1, \\ & X_4 < X_7, & X_4 \leq 6, & X_4 \geq 4, & \\ & X_5 < X_4, & X_5 < X_6, & X_5 \leq 6, & X_5 \geq 2, \\ & X_6 \leq X_7, & X_6 \geq 5, & X_6 \leq 12, & \\ & X_7 < 8, & X_7 \geq 1, & & \\ & X_8 \leq X_6, & X_8 > 3\} & & & . \end{array}$$

Figure 1 gives the construction of $°A_{\min}$. We note that $(X_1 < 5)$ is equivalent to $(X_1 \leq 4)$ in the integer domain, and a similar transformation is applied to other selections. X_2 and X_3 constitute an *SCC*, thus they are collapsed into a single node, denoted as X_{23}. The pair inside each node denotes the minimum range for that node. The pair outside each node denotes the "real" minimum range. For node X_1, $A^{X_1}_{low} = -\infty$, because $C^{X_1}_{low} = -\infty$ and X_1 does not have any parent. For node X_8, $A^{X_8}_{low} = 4$, because X_1 is the only parent of X_8, $A^{X_1}_{low} = -\infty$ and $C^{X_8}_{low} = 4$. For node X_4, $A^{X_4}_{low} = 4$, because $C^{X_4}_{low} = 4$ (note that although X_5 and X_{23} are X_4's

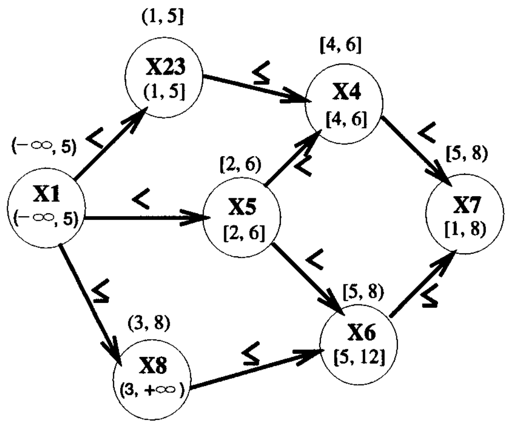

Fig. 2. Construction of assignment satisfying G_S in the real domain.

parents, they will yield smaller $A_{low}^{X_4} = 3$ and $A_{low}^{X_4} = 2$, thus it is ignored). We note that $A_{low}^{X_7} = (5)$ is different from $C_{low}^{X_7}$ (1) because $A_{low}^{X_4}$ is 4 with a "<" label on the edge, which will yield $A_{low}^{X_7} = 5$ (in fact, $A_{low}^{X_7} = 5$ is also a result of $A_{low}^{X_6} = 5$ and its "≤" label on the edge).

We proceed to solve the satisfiability problem in the real domain using a similar strategy. We construct the same G_S and $G_{S_{collapsed}}$ as we did in the previous integer domain case.

We note that in the real domain, we can not transform $(X < C)$ to $(X \leq C - 1)$. Instead, we need to mark the minimum ranges to be "<" ("open bound") or "≤" ("closed bound").

Consider a set of inequalities for variable X: $\{(X \textbf{ op } C_1), (X \textbf{ op } C_2), \ldots, (X \textbf{ op } C_k)\}$ in S, $\textbf{op} \in \{<, \leq\}$. We identify $C_{up}^X = \min_{i=1}^k (C_i)$. If $(X < C_{up}^X) \in S$, then C_{up}^X is *open* (i.e., the end point of the bound is not qualified by the inequalities); otherwise, C_{up}^X is *closed* (i.e., the end point of the bound is qualified by the inequalities). Similarly, either an open or a closed C_{low}^X can be obtained. In other words, the difference between this case and the corresponding case in the integer domain lies in the openness and/or the closedness of the bounds.

Similarly, the "real" minimum ranges are also computed with "open" or "closed" markers on the bounds. $A_{low}^X = \max(C_i, C_{low}^X)$, $C_i = A_{low}^{X_i}$ for all parents X_i of X. First we need to decide the marker of C_i before that of A_{low}^X. C_i is "closed" if the edge between X_i and X is labeled with "≤" and $A_{low}^{X_i}$ is "closed"; otherwise C_i is "open." A_{low}^X is "open" if one of $\max(C_i,$ $C_{low}^X)$ *is "open;" otherwise* A_{low}^X is "closed." $A_{up}^X = \min(C_j, C_{up}^X)$, $C_j = A_{up}^{X_j}$ for all X's children X_j. The marking of C_j and A_{up}^X is similar.

Example 3.1.2 Using Example 3.1.1, suppose that all variables and constants are in the real domain. Figure 2 gives the construction of $°A_{\min}$ and $°A_{\max}$.

LEMMA 3.1.2 *In the real domain involving* $\textbf{OP}_{\neg\neq}$, *a conjunctive formula S is satisfiable if and only if,*

(1) *every SCC in G_S does not contain an edge labeled "<", and*
(2) *for each variable X in $S_{collapsed}$, $A^X_{low} < A^X_{up}$, or $A^X_{low} = A^X_{up}$ and both A^X_{up} and A^X_{low} are "closed".*

A proof can be easily constructed from that of Lemma 3.1.1. An algorithm to evaluate if S is satisfiable in the real domain is provided in the followup.

Algorithm 2

Step 1: Perform the transformation on each inequality in S such that only $(X\{<, \le\}Y)$ and $(X\{<, >, \le, \ge\}C)$ exists in S. If $(X \ne Y)$ or $(X \ne C)$ is found, the satisfiability will be handled in another algorithm as described in the next section, so exit. At the same time, eliminate the trivial inequalities, namely, (X **op** X) and (C_1 **op** C_2) in S. If $(X\{<, >, \ne\}X)$, or $(C\{<, >, \ne\}C)$, or $(C_1\{<, \le, =\}C_2)$ with $C_1 > C_2$, or $(C_1\{>, \ge, =\}C_2)$ with $C_1 < C_2$, then report that S is unsatisfiable, and exit.

Step 2: Construct the minimum range C^X_{low}, C^X_{up} for each X by scanning all $(X\{<, >, \le, \ge\}C)$.

Step 3: Construct the labeled directed graph G_S, detecting all *SCC*s. If any "<" is found in any *SCC*, then S is unsatisfiable; exit. Otherwise, collapse *SCC*s and obtain an acyclic graph $G_{S_{collapsed}}$.

Step 4: Topologically sort the graph. Then the "real" minimum ranges A^X_{low}, A^X_{up} are computed.

Step 5: If any $A^X_{low} > A^X_{up}$ or $A^X_{low} = A^X_{up}$ with either bound is "open," S is unsatisfiable. Otherwise, report that S is satisfiable.

3.2 The General Satisfiability Problem

In the integer domain, this problem has been proved to be NP-hard [Rosenkrantz and Hunt 1980]. The known NP-complete problem of determining whether an undirected graph is three colorable [Cormen et al. 1990; Garey et al. 1976] is reduced to an instance of this satisfiability problem.

In the remainder of this subsection we consider only the real domain. It is sufficient to consider **op** $\in \{\le, \ne\}$ for inequalities of the type (X **op** Y) $\in S$ and **op** $\in \{\le, \ge, \ne\}$ for inequalities of the type (X **op** C) $\in S$. For all $(X \le Y)$ and $(X\{\le, \ge\}C)$ in S, we construct G_S, $G_{S_{collapsed}}$, the minimum ranges, and the "real" minimum ranges of S as we did in Section 3.1. Clearly, if any *SCC* of G_S contains nodes X and Y, and $(X \ne Y)$ is in S, then S is unsatisfiable.

LEMMA 3.2.1 *In the real domain involving* $\mathbf{OP_{all}}$, *a conjunctive formula S is satisfiable if and only if,*

(1) *for each $(X \ne Y) \in S$, X and Y do not belong to the same SCC, and*
(2) *for each variable X of S, either $(A^X_{low} < A^X_{up})$ or ($A^X_{low} = A^X_{up}$ and $(X \ne A^X_{low}) \in S$).*

PROOF.

Necessity: Necessity is proved by constructing a satisfiable assignment of S. We first collapse S to $S_{collapsed}$. The collapsing will not affect $(X \neq Y)$ inasmuch as X and Y are in different *SCC*s. For each $X \in S_{collapsed}$ with $A^X_{low} = A^X_{up}$, $X = A^X_{low}$ is implied. We further replace all occurrences of X by A^X_{low} for all inequalities of $S_{collapsed}$. The replacement will not result in trivial inequalities of the form $(C \neq C)$ and $C = A^X_{low}$, due to $(X \neq A^X_{low}) \notin S$. Let $S'_{collapsed}$ be the set after the replacement. The collapsing and the replacement do not change the satisfiability of S, namely, if an assignment satisfies $S_{collapsed}$, it must satisfy $S'_{collapsed}$, and the inverse is true. Let $G'_{S_{collapsed}}$ be the labeled directed graph of $S'_{collapsed}$ (all the labels are "$\leq$"). For each variable X, we have $A^X_{low} < A^X_{up}$. Let $°A$ be an assignment of $S'_{collapsed}$ with each X assigned the value $A^X_{low} + \delta_X$. δ_X is a positive real small enough to satisfy $A^X_{low} + \delta_X < A^X_{up}$. Furthermore, for each parent X_i of X, we assure that $\delta_{X_i} < \delta_X$. These δ can always be constructed because the real domain is a dense domain. Clearly, $°A$ satisfies $G'_{S_{collapsed}}$.

Because the unequalities of S comprise a finite set, we can modify the δ (each of them has infinite choices), to ensure that $G'_{S_{collapsed}}$ is satisfied and the unequalities are also satisfied.

Sufficiency: If S is satisfiable, it is easy to verify Condition (1) and $A^X_{low} \leq A^X_{up}$ by contradictions. If $A^X_{low} = A^X_{up}$ and $(X \neq A^X_{low}) \in S$, it is impossible that an assignment exists that satisfies $(X = A^X_{low})$ *and* $(X \neq A^X_{low})$ concurrently. This contradicts the hypothesis that S is satisfiable. □

An algorithm to evaluate if S is satisfiable in the real domain is provided in the following.

Algorithm 3

Step 1: Perform the transformation on each inequality in S such that only $(X\{\leq, \neq\}Y)$ and $(X\{\leq, \geq, \neq\}C)$ exist in S. At the same time, eliminate the trivial inequalities, namely, $(X$ **op** $X)$ and $(C_1$ **op** $C_2)$ in S. If $(X\{<, >, \neq\}X)$, or $(C\{<, >, \neq\}C)$, or $(C_1\{<, \leq, =\}C_2)$ with $C_1 > C_2$, or $(C_1\{>, \geq, =\}C_2)$ with $C_1 < C_2$, then report that S is unsatisfiable; exit.

Step 2: Construct the minimum range $[C^X_{low}, C^X_{up}]$ for each X by scanning all $(X\{\leq, \geq\}C)$.

Step 3: Construct the labeled directed graph G_S; detect all *SCC*s. If any "$\neq$" is found in any *SCC*, then S is unsatisfiable; exit. Otherwise, collapse *SCC*s and obtain an acyclic graph $G_{S_{collapsed}}$.

Step 4: Topologically sort all nodes on the graph. Then, the "real" minimum ranges A^X_{low}, A^X_{up} are computed.

Step 5: If any $(A^X_{low} > A^X_{up})$ or $(A^X_{low} = A^X_{up}$ and $(X \neq A^X_{low}) \notin S)$, S is unsatisfiable. Otherwise, report that S is satisfiable.

THEOREM 3.2.1 *In the real domain involving* $\mathbf{OP_{all}}$, *the satisfiability problem is solvable in* $O(|S|)$ time.

PROOF. Steps 1 and 2 can be done in one scan of S. *SCC*s can be detected by two depth-first-searches in $O(|S|)$ time. The unequalities in *SCC*s can be checked by one scan of the unequalities in S. Steps 4 and 5 can be done in $O(|S|)$ time. As a result, the total time complexity is $O(|S|)$. □

4. DOES S IMPLY T?

We first discuss the restricted implication problem (i.e., $\mathbf{OP}_{\neg\neq}$ is used for inequalities). In Section 4.2, the general implication problem is discussed (i.e., $\mathbf{OP}_{\mathbf{all}}$ is used for inequalities).

4.1. The Restricted Implication Problem

We consider the integer domain first.

LEMMA 4.1.1 *In the integer domain involving $\mathbf{OP}_{\neg\neq}$, S implies T if and only if S is unsatisfiable, or*

(1) *for any $(X \le Y) \in T$ there exists a path from X to Y in $G_{S_{collapsed}}$, or $A^X_{up} \le A^X_{low}$;*
(2) *for any $(X < Y) \in T$ there exists a path from X to Y in $G_{S_{collapsed}}$ with at least one edge of the path labeled with "<", or $A^X_{up} < A^Y_{low}$;*
(3) *for any $(X \le C) \in T$, $C \ge A^X_{up}$; and*
(4) *for any $(X \ge C) \in T$, $C \le A^X_{low}$.*

PROOF.
Sufficiency: If S is unsatisfiable, then S implies T. Here we only consider the situation that S is satisfiable. It is clear that if Conditions (1)–(4) are true, then any $(X \le Y)$, $(X < Y)$, $(X \le C)$, or $(X \ge C)$ in T are implied by S; thus S implies T.
Necessity: We show that if S implies T, then Conditions (1)–(4) are true. For any $(X \le Y) \in T$, if there exists a path from X to Y, then we are done. Otherwise, we prove that $A^X_{up} \le A^Y_{low}$. Suppose $A^X_{up} > A^Y_{low}$. We want to construct an assignment that satisfies S and has X and Y assigned A^X_{up} and A^Y_{low}, respectively. Let A denote the set of variables reachable from X (including X) and B the set of variables that have path(s) to Y (including Y). Note that A and B have no common variable(s) because by assumption a path does not exist from X to Y. Each variable in A, say X_A, is assigned $A^{X_A}_{up}$; each variable in B, say X_B, is assigned $A^{X_B}_{low}$. Clearly, this assignment satisfies S and contradicts $(X \le Y) \in T$. A similar argument is applicable to Condition (2). For Condition (3), if $C < A^X_{up}$, then $°A_{\max}$ satisfies S but does not satisfy $(X \le C) \in T$ as X is assigned A^X_{up}. This is a contradiction. The proof for Condition (4) is similar. □

An algorithm to evaluate if S implies T in the integer domain is provided next.

Algorithm 4

Step 1: Use Algorithm 1 to evaluate S. If S is unsatisfiable, then the implication is true; exit.

Step 2: Perform the transformation on each inequality in T such that only ($X\{<, \leq\}Y$) and ($X\{\leq, \geq\}C$) exist in T. Each variable is also replaced by its representative variable. If ($X \neq Y$) or ($X \neq C$) is found, report NP-hard problem and exit. At the same time, eliminate the trivial inequalities, namely, (X **op** X) and C_1 **op** C_2) in T. If ($X\{<, >, \neq\}X$), or ($C\{<, >, \neq\}C$), or ($C_1\{<, \leq, =\}C_2$) with $C_1 > C_2$, or ($C_1\{>, \geq, =\}C_2$) with $C_1 < C_2$, then report that the implication is not true; exit.

Step 3: Compute the transitive closure of $G_{S_{collapsed}}$ and remember (in each depth-first search) if there exists a path containing "<" edge(s) from the starting node to each child.

Step 4: For each ($X \leq Y$) $\in T$, if there neither exists a path from X to Y nor $A^X_{up} \leq A^Y_{low}$, then the implication is not true; exit.

Step 5: For each ($X < Y$) $\in T$, if there neither exists a path containing "<" edge(s) from X to Y nor $A^X_{up} < A^Y_{low}$, then the implication is not true; exit.

Step 6: For each ($X \leq C$) $\in T$, if $C < A^X_{up}$, then the implication is not true; exit.

Step 7: For each ($X \geq C$) $\in T$, if $C > A^X_{low}$, then the implication is not true; exit. Otherwise, report that the implication is true.

THEOREM 4.1.1 *In the integer domain involving* $\mathbf{OP}_{\neg\neq}$, *the implication problem can be solved in* $O(|S|^2 + |T|)$ *time.*

PROOF. Steps 1 and 2 take $O(|S|)$ time. The transitive closure of $G_{S_{collapsed}}$ costs $O(|S|)$. Steps 4–7 take $O(|T|)$ time. As a result, the total complexity is $O(|S|^2 + |T$

For the real domain, it is sufficient to consider **op** $\in \{<, \leq\}$ for inequalities of the type (X **op** Y) $\in S$ and **op** $\in \{<, >, \leq, \geq\}$ for inequalities of the type (X **op** C) $\in S$.

LEMMA 4.1.2 *In the real domain involving* $\mathbf{OP}_{\neg\neq}$, *S implies T if and only if S is unsatisfiable, or*

(1) for any ($X \leq Y$) $\in T$ *there exists a path from X to Y in* G_S, *or* $A^X_{up} \leq A^Y_{low}$;

(2) *for any* ($X < Y$) $\in T$ *there exists a path from X to Y in* G_S *with at least one edge of the path labeled with "<," or either* $A^X_{up} < A^Y_{low}$ *or* $A^X_{up} = A^Y_{low}$ *and one of them is "open";*

(3) *for any* ($X \leq C$) $\in T$, $C \geq A^X_{up}$;

(4) *for any* ($X < C$) $\in T$, $C > A^X_{up}$ *or* $C = A^X_{up}$ *and* A^X_{up} *is "open";*

(5) *for any* ($X \geq C$) $\in T$, $C \leq A^X_{low}$; *and*

(6) for any ($X > C$) $\in T$, $C < A^X_{low}$ *or* $C = A^X_{low}$ *and* A^X_{low} *is "open."*

A proof can be constructed using the same reasoning as in Lemma 4.1.1. The algorithm to evaluate if S implies T in the real domain is provided in the following.

Algorithm 5

Step 1: Use Algorithm 2 to evaluate S. If S is unsatisfiable, then the implication is true; exit.

Step 2: Perform the transformation on each inequality in T such that only $(X\{<, \leq\}Y)$ and $(X\{<, >, \leq, \geq\}C)$ exist in T. Each variable is also replaced by its representative variable. If $(X \neq Y)$ or $(X \neq C)$ is found, the implication is handled in Algorithm 6; exit. At the same time, eliminate the trivial inequalities, namely, $(X \textbf{ op } X)$ and $(C_1 \textbf{ op } C_2)$ in T. If $(X\{<, >, \neq\}X)$, or $(C\{<, >, \neq\}C)$, or $(C_1\{<, \leq, =\}C_2)$ with $C_1 > C_2$, or $(C_1\{>, \geq, =\}C_2)$ with $C_1 < C_2$, then report that the implication is not true; exit.

Step 3: Compute the transitive closure of $G_{S_{collapsed}}$ and remember if there exists a path containing one or more "<" edges from X to Y.

Step 4: For each $(X \leq Y) \in T$, if there neither exists a path from X to Y nor $A^X_{up} \leq A^Y_{low}$, then the implication is not true; exit.

Step 5: For each $(X < Y) \in T$, if there neither exists a path containing "<" edge(s) from X to Y nor $A^X_{up} < A^Y_{low}$, then the implication is not true; exit.

Step 6: For each $(X \leq C) \in T$, if $C < A^X_{up}$, then the implication is not true; exit.

Step 7: For each $(X < C) \in T$, if $C < A^X_{up}$, or $C = A^X_{up}$ and A^X_{up} is "closed", then the implication is not true; exit.

Step 8: For each $(X \geq C) \in T$, if $C > A^X_{low}$, then the implication is not true; exit.

Step 9: For each $(X > C) \in T$, if $C > A^X_{low}$ and A^X_{low} is "closed," then the implication is not true; exit. Otherwise, report that the implication is true.

The complexity is deducted by Algorithm 4, namely $O(|S|^2 + |T|)$.

THEOREM 4.1.2 *In the real domain involving* $\mathbf{OP}_{\neg\neq}$, *the implication problem can be solved in* $O(|S|^2 + |T|)$ *time.*

4.2 The General Implication Problem

For the integer domain, this implication problem is NP-hard by Theorem 2.1 and the corresponding satisfiability problem is NP-hard. In the following discussion, we only consider the real domain.

A. Klug [1988] and J. Ullman [1989] proposed an $O(|S|^3 + |T|)$ algorithm to solve the implication problem involving inequalities of the form $(X \textbf{ op } Y)$.[4] This is a special case of our situation because we consider the case that the two allowed forms of inequalities are $(X \textbf{ op } Y)$ and $(X \textbf{ op } C)$, where C is a constant in the domain of X. We first briefly discuss the Klug–Ullman algorithm on which our approach is based.

Klug and Ullman's [1989] approach uses an idea similar to the way that functional dependencies in a relational database system are handled, where a collection of axioms, *Klug–Ullman Axioms*, is used [Ullman 1989]. The following eight axioms for inequalities are then shown to be sound

[4]In fact, this algorithm is also directly applicable to the integer domain for this restricted type of inequalities.

(only inferring correct inequalities) and complete (inferring all correct inequalities) [Ullman 1989].

A1: $(X \leq X)$

A2: $(X < Y)$ implies $(X \leq Y)$

A3: $(X < Y)$ implies $(X \neq Y)$

A4: $(X \leq Y)$ and $(X \neq Y)$ imply $(X < Y)$

A5: $(X \neq Y)$ implies $(Y \neq X)$

A6: $(X < Y)$ and $(Y < Z)$ imply $(X < Z)$

A7: $(X \leq Y)$ and $(Y \leq Z)$ imply $(X < Z)$

A8: $(X \leq Z)$, $(Z \leq Y)$, $(X \leq W)$, $(W \leq Y)$ and $(W \neq Z)$ imply $(X \neq Y)$.

Then Ullman's algorithm first computes the inequality closure, denoted as S^+, by applying Axioms A1–A8 until they no longer generate any new inequalities. The closure computation procedure is as follows:

(1) Convert each $<$ relationship, say $(X < Y)$, into $(X \leq Y)$ and $(X \neq Y)$.
(2) Compute the transitive closure of the $\leq$ relationships.
(3) Apply Axiom A8 to infer additional $\neq$ relationships.
(4) Reconstruct the $<$ relationships using Axiom A4; that is, $(X < Y)$ if $(X \leq Y)$ and $(X \neq Y)$.

Step (1) only takes $O(|S|)$ time. Step (2) can be done in $O(|S|^2)$ time by performing a depth-first search, from each variable, of the graph $G_S = (V_S, E_S)$ whose nodes are the variables and whose arcs are the $\leq$ relationships [Aho et al. 1983]. Step (3) can be done in $O(|S|^3)$ time: for each pair X and Y, we find all those Z such that $(X \leq Z \leq Y)$, in $O(|S|)$ time, just by enumerating the Zs and checking whether $(X \leq Z)$ and $(Z \leq Y)$ are both known. Then we check if any two such Zs are related by $\neq$. That takes $O(|S|)$ time. Note that it is sufficient to check the original $\neq$ pairs, that is, those given in S and those added in Step (1). It is not necessary to check the new pairs added in Step (3). The total time for Step (3) is $O(|S|)$ times the number of pairs of variables, which is bounded by $|S|^2$. Thus the time complexity of the algorithm is $O(S|S|^3)$. The total time complexity of Ullman's algorithm to test whether S implies T is $O(|S|^3 + |T|)$.

For our strategy, it is sufficient to consider **op** $\in \{\leq, \neq\}$ for inequalities of the type $(X$ **op** $Y) \in S$ and **op** $\in \{\leq, \geq, \neq\}$ for inequalities of the type $(X$ **op** $C) \in S$. We first construct an inequality set S_{NEW} from S as follows: let $(C_1, C_2, \ldots, C_k)$ be all the distinct constants in ascending order of their values, which are used in all inequalities of the form $(X$ **op** $C_i)$ in S. We introduce k dummy variables (W_1, W_2, ..., W_k) to represent these k distinct constants, and $2k - 2$ inequalities, $S_{ADDED} = \{(W_1 \leq W_2), (W_2 \leq W_3), \ldots, (W_{k-1} \leq W_k), (W_1 \neq W_2), (W_2 \neq W_3), \ldots, (W_{k-1} \neq W_k)\}$ to represent the relationships among the newly introduced variables. For each $(X \textbf{ op } C_i) \in S$, we transform it into $(X \textbf{ op } W_i)$, where W_i represents the dummy variable for C_i. Let S' be the inequality set after the preceding transformation, and $S_{NEW} = S' \cup S_{ADDED}$. $G_{S_{NEW}}$ is constructed for the new inequality set S_{NEW} in the same way that G_S is constructed from S. Now S_{NEW} only contains inequalities of the form $(X \textbf{ op } Y)$; therefore the closure of S_{NEW}, denoted as S^+_{NEW}, is computed [Ullman 1989]. After S^+_{NEW} is computed, dummy variables are replaced back with corresponding constants. It can be directly observed that after the preceding transformation, the size of S_{NEW} is still bounded by $O(|S|)$. Note that in the satisfiability evaluation of S, we already have computed the minimum and "real" minimum ranges.

LEMMA 4.2.1 *In the real domain involving* $\mathbf{OP}_{all}$, *S implies T if and only if S is unsatisfiable, or*

(1) *for any* $(X \leq Y) \in T$, $(X \leq Y) \in S^+_{NEW}$, *or* $A^X_{up} \leq A^Y_{low}$;
(2) *for any* $(X \neq Y) \in T$, $(X \neq Y) \in S^+_{NEW}$;
(3) *for any* $(X \leq C) \in T$, $C \geq C^X_{up}$;
(4) *for any* $(X \geq C) \in T$, $C \leq C^X_{low}$; *and*
(5) *for any* $(X \neq C) \in T$, $(X \neq C) \in S^+_{NEW}$ *or* $C < C^X_{low}$ *or* $C > C^X_{up}$.

PROOF. Conditions (1) and (2) directly follow the soundness and completeness of the Klug–Ullman Axioms. The proof for other conditions can be done easily. □

However, the time complexity by directly applying Lemma 4.2.1 is exactly the same as that of Ullman's algorithm. We now provide a more efficient algorithm with the complexity $O(|S|^{2.376} + |T|)$. The following algorithm is based on the one given in Sun and Weiss [1994] which handles only inequalities of the form $(X \textbf{ op } Y)$.

First of all, we can see from Ullman's algorithm that the test whether an inequality of $(X \leq Y) \in T$ is implied by S can be decided easily in $O(|S|^2)$ time. The more difficult part is to test whether an inequality of $(X \neq Y) \in T$ is implied by S (due to the Axiom 8). Thus we focus on solving this problem. An example is also provided at the end of this subsection to illustrate our approach.

Given S_{NEW} (it is sufficient to consider the operators $\in \{\leq, \neq\}$), we first construct a *directed graph* $G'_{S_{NEW}} = (V_S, E_S)$, where $V_S = A \cup B \cup C$ and $E_S = E_{AB} \cup E_{BC}$; $v_W \in A$ and $v_W \in C$ iff W is a variable of S_{NEW}; $v_{XY} \in B$ iff $(X \neq Y) \in S_{NEW}$; $(v_W, v_{XY}) \in E_{AB}$ iff $(W \leq X)$ and $(W \leq Y)$ are implied by S_{NEW}; and $(v_{XY}, v_Z) \in E_{BC}$ iff $(X \leq Z)$ and $(Y \leq Z)$ are implied by S_{NEW}. This graph is used exclusively to deduce unequalities explicitly (in S) and implicitly implied by S. Example 4.2.1 shows how such a graph can be constructed.

$|A|$, $|B|$, and $|C|$ are bounded by $O(|S_{NEW}|)$, thus $|V_S|$ is bounded by $O(|S_{NEW}|)$. We can easily observe that the in-degree of any node in A is zero, the out-degree of any node in C is zero, and B may contain nodes with nonzero in-degree and/or nonzero out-degree. Clearly $|E_S|$ is bounded by $O(|S_{NEW}|)$ and can be computed in $O(|S_{NEW}|^2)$ time as follows: we first compute the transitive closure (wrt $\leq$) of S_{NEW} in $O(|S_{NEW}|^2)$, then for each $(X \neq Y) \in S_{NEW}$, we examine each variable, say W, in S_{NEW}. If $(W \leq X)$ and $(W \leq Y)$ are implied by S_{NEW} (it takes constant time after the transitive closure is computed), we add the edge $(v_W, v_{XY}) \in E_{AB}$. E_{AB} can be constructed in $O(|S_{NEW}|^2)$. In a similar manner, E_{BC} can be constructed. The following lemma is essential for testing whether $(X \neq Y)$ is implied by S_{NEW} [Sun and Weiss 1994].

LEMMA 4.2.2 *In the real domain involving* $\mathbf{OP}_{all}$, $(X \neq Y)$ *is implied by* S_{NEW} *if and only if*

—$(X \neq Y) \in S_{NEW}$, *or*

—*there exists a path from X to Y of length 2 in* $G'_{S_{NEW}}$, *namely,* $v \in V_S$, *such that* (v_X, v), $(v, v_Y) \in E_S$.

The algorithm to evaluate if S implies T in the real domain is provided.

Algorithm 6

Step 1: Use Algorithm 3 to evaluate S. If S is unsatisfiable, then the implication is true; exit.

Step 2: Perform the transformation on each inequality in T such that only $(X\{\leq, \neq\}Y)$ and $(X\{\leq, \geq, \neq\}C)$ exist in T. Each variable is also replaced by its representative variable. At the same time, eliminate the trivial inequalities, namely, $(X$ **op** $X)$ and C_1 **op** $C_2)$ in T. If $(X\{<, >, \neq\}X)$, or $(C\{<, >, \neq\}C)$, or $(C_1\{<, \leq, =\}C_2)$ with $C_1 > C_2$, or $(C_1\{>, \geq, =\}C_2)$ with $C_1 < C_2$, then report that the implication is not true; exit.

Step 3: Construct $G'_{S_{NEW}}$ and its adjacency matrices. Perform the matrix multiplication algorithm. For the deduced unequalities, replace the dummy variables back with the corresponding constants.

Step 4: Construct $G_{S_{collapsed}}$ and compute the transitive closure of $G_{S_{collapsed}}$.

Step 5: For each $(X \leq Y) \in T$, if there neither exists a path from X to Y nor $A^X_{up} \leq A^Y_{low}$, then the implication is not true; exit.

Step 6: For each $(X \neq Y) \in T$, if it is not in S or deduced in Step 3, the implication is not true; exit.

Step 7: For each $(X \leq C) \in T$, if $C < A^X_{up}$, then the implication is not true; exit.

Step 8: For each $(X \geq C) \in T$, if $C > A^X_{low}$, then the implication is not true; exit.

Step 9: For each $(X \neq C) \in T$, if it neither is in S nor deduced in Step 3, nor $C < C^X_{low}$ or $C > C^X_{up}$, then the implication is not true; exit. Otherwise, report that the implication is true.

THEOREM 4.2.1. *In the real domain involving* $\mathbf{OP}_{all}$, *the implication problem can be solved in* $O(min(|S|^{2.376} + |T|, |S|*|T|))$ *time.*

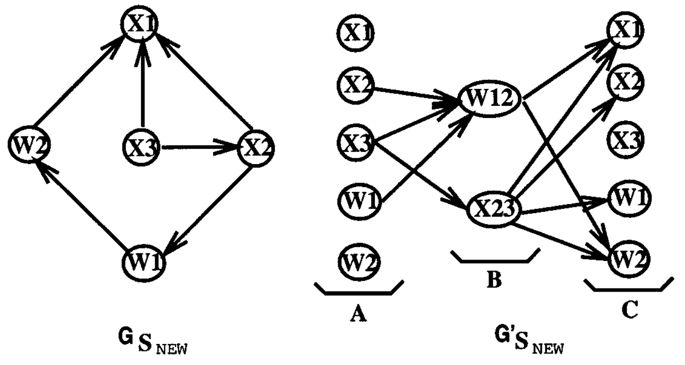

Fig. 3. Deduction of unequalities.

Proof. The second component of $O(|S|*|T|)$ directly follows from Theorem 2.2. Identifying all paths of length 2 can be obtained by multiplying two $|S| \times |S|$ adjacency matrices. It has been shown that the multiplication of two $n \times n$ matrices can be computed in $O(n^{2.376})$ time [Coppersmith and Winograd 1987]. After the number of paths of length 2 for all pairs of nodes in $G'_{S_{NEW}}$ have been computed, it only takes constant time to test whether $(X \neq Y)$ is implied by S_{NEW}. Note that $|S_{NEW}|$ is bounded by $O(|S|)$. Constructing $G'_{S_{NEW}}$ takes $O(|S_{NEW}|^2)$ time, computing the transitive closure (wrt $\leq$) for $G'_{S_{NEW}}$ can be done in $O(|S_{NEW}|^2)$ time, and computing all paths of length 2 in $G'_{S_{NEW}}$ needs $O(|S_{NEW}|^{2.376})$. And for each $(X \leq Y)$ or $(X \neq Y) \in T$, it only takes constant time to test whether it is implied by S_{NEW}. Overall, the algorithm takes $O(|S_{NEW}|^{2.376} + |T|)$. □

Example 4.2.1 Consider $S = \{(X_1 \geq X_2), (X_2 \geq X_3), (X_1 \geq 5), (X_1 \geq X_3), (X_2 \leq 3), (X_2 \neq X_3)\}$. S is clearly satisfiable (e.g., $\{6/X_1, 2/X_2, 1/X_3\}$ is an assignment that satisfies S). $S_{NEW} = \{(X_1 \geq X_2), (X_2 \geq X_3), (X_1 \geq W_2), (X_1 \geq X_3), (X_2 \leq W_1), (X_2 \neq X_3), (W_1 \leq W_2), (W_1 \neq W_2)\}$, where W_1 and W_2 represent integers 3 and 5, respectively. After computing all transitive relationships of $G_{S_{NEW}}$, the following can be obtained.

$X_1 \leq$	X_1				
$X_2 \leq$	X_1	X_2		W_1	W_2
$X_3 \leq$	X_1	X_2	X_3	W_1	W_2
$W_1 \leq$	X_1			W_1	W_2
$W_2 \leq$	X_1				W_2

$G_{S_{NEW}}$ and $G'_{S_{NEW}}$ of S are shown in Figure 3.

There are ten paths in G' of length 2: $((X_3, X_{23}), (X_{23}, X_1))$, $((X_3, X_{23}), (X_{23}, X_2))$, $((X_3, X_{23}), (X_{23}, W_1))$, $((X_3, X_{23}), (X_{23}, W_2))$, $((X_2, W_{12}), (W_{12}, X_1))$, $((X_2, W_{12}), (W_{12}, W_2))$, $((X_3, W_{12}), (W_{12}, X_1))$, $((X_3, W_{12}),$

$(W_{12}, W_2))$, $((W_1, W_{12}), (W_{12}, X_1))$, $((W_1, W_{12}), (W_{12}, W_2))$. Thus the deduced $\neq$ inequalities are: $(X_3 \neq X_1)$, $(X_3 \neq X_2)$, $(X_3 \neq W_1)$, $(X_3 \neq W_2)$, $(X_2 \neq X_1)$, $(X_2 \neq W_2)$, $(W_1 \neq X_1)$, and $(W_1 \neq W_2)$.

5. IMPLEMENTATION AND EXPERIMENT

We have implemented all the algorithms proposed in Sections 3 and 4. The implementation is integrated into a Microsoft Windows-based graphic user interface front-end. The C++ programs can be obtained by anonymous ftp from ⟨archive.fiu.edu⟩ under ⟨weisun⟩ directory.

In our implementation, we further extended the algorithms to solve the restricted problems with unequalities allowed in the inequality set in some situations. In the processing of a restricted satisfiability problem "Is S satisfiable?" in the integer domain, the unequalities in S are not touched until the satisfiability of S without the unequalities is decided. If the answer is "unsatisfiable," then the answer to the original problem is unsatisfiable. For the situation of a "satisfiable" answer, if the "real" minimum ranges of X and Y for each $(X \neq Y) \in S$ have an empty intersection, $(X \neq Y)$ is satisfied by any assignment that satisfies S without the unequalities. For each $(X \neq C)$ in S, if C does not fall in the "real" minimum range of X, $(X \neq C)$ is satisfied. If all the unequalities are satisfied, the final answer is "satisfiable," otherwise, the algorithm reports NP-hard.

For the restricted implication "S implies T" in the integer domain, our algorithm can handle the unequalities in some cases. The satisfiability of S is decided by the preceding algorithm. If the result is NP-hard, then NP-hard is reported for the corresponding implication problem. If the result is "unsatisfiable," the implication is true. Otherwise, we consider all the unequalities in T: for each $(X \neq Y)$ in T, if it is in S, or the "real" minimum ranges of X and Y have empty intersection, the unequality is implied. For each $(X \neq C)$ in T, if it is in S, or C does not fall in the "real" minimum range of X, then the unequality is implied. If all the unequalities in T are implied by S, the final answer to the implication is true; otherwise, the algorithm reports NP-hard.

It is noted here that a query may have hundreds of joins in a database system, especially in a deductive or logic-based system, because these database systems typically contain hundreds or thousands of rules in ever increasingly complex database applications and they are normally implemented on the underlying relational technology. This basically translates into hundreds or thousands of relational joins [Krishnamurthy et al. 1986; Gallaire et al. 1984; Ullman 1989]. In addition, query sizes could be huge in the ever increasingly complex database applications.

We have also implemented the previously known $O(|S|^3)$ algorithms for the restricted satisfiability and implication problems. Based on our implementation, we have conducted experiments to compare the performance of our algorithms with that of the previously known ones based on the computation costs of sample problems whose sizes vary from 3 to 300. We

Table II. Performance Comparison of Solving Satisfiability Problems

Sample Problems	$S03_1$	$S03_2$	$S03_3$	$S03_4$	$S03_5$	$S03_6$	$S03_7$	$S03_8$	$S03_9$	$S03_{10}$
No. of Variables	6	5	4	3	3	2	2	2	2	0
Previous Algorithm	29	21	16	11	12	7	7	4	4	2
Our Algorithm	3	3	3	2	2	2	2	3	3	1
Improvements (times)	8.7	6.0	4.3	4.5	5.0	2.5	2.5	0.33	0.33	1.0

(a): Problem Size 3: Average Improvement = 3.5 (times)

Sample Problems	$S05_1$	$S05_2$	$S05_3$	$S05_4$	$S05_5$	$S05_6$	$S05_7$	$S05_8$	$S05_9$	$S05_{10}$
No. of Variables	10	7	6	4	4	3	2	2	2	0
Previous Algorithm	88	41	35	23	22	13	11	4	4	2
Our Algorithm	4	3	4	3	3	3	4	3	2	2
Improvements (times)	21.0	12.7	7.8	6.7	6.3	3.3	1.8	0.3	1.0	0.0

(b): Problem Size 5: Average Improvement = 6.1 (times)

Sample Problems	$S10_1$	$S10_2$	$S10_3$	$S10_4$	$S10_5$	$S10_6$	$S10_7$	$S10_8$	$S10_9$	$S10_{10}$
No. of Variables	20	14	10	8	7	4	4	3	1	0
Previous Algorithm	501	222	113	84	87	31	32	22	6	1
Our Algorithm	9	8	8	6	7	8	8	8	1	1
Improvements (times)	54.7	26.8	13.1	13.0	11.4	2.9	3.0	1.8	5.0	0.0

(c): Problem Size 10: Average Improvement = 13.2 (times)

Sample Problems	SAT30	SAT50	SAT100	SAT150	SAT300
Problem Sizes	30	50	100	150	300
No. of Variables	6	10	25	30	40
Previous Algorithm	208	378	4245	7245	5458
Our Algorithm	20	38	85	132	266
Improvements (times)	9.4	8.9	48.9	53.9	19.5

(d): Problem Sizes ≥ 30 (including 30, 50, 100, 150, 300)

measured the computation costs by the CPU time (milliseconds) used to solve the sample problems as shown in Tables II and III.[5]

These results clearly show the superiority of the proposed strategies: our algorithms have good to excellent improvement over previous best known results for even small-size problems. For medium and large-size problems, the improvement is obvious and its trend can be expected.

From Table III, a similar observation can be drawn as to the performance of our algorithm in solving the implication problems.

6. CONCLUSIONS

In this article we have provided a comprehensive and systematic study of these problems for conjunctive inequalities of the form (X **op** Y) or (X **op**

[5]Our computation data are obtained by running our program on solving the sample problems on a relatively slow Intel 486 SX-25 standalone IBM-compatible PC; due to that the basic CPU time measure is 1 MS.

Table III. Performance Comparison of Solving Implication Problems

Sample Problems	$I03_1$	$I03_2$	$I03_3$	$I03_4$	$I03_5$	$I03_6$	$I03_7$	$I03_8$	$I03_9$	$I03_{10}$
No. of Variables	6	6	4	2	3	3	2	1	1	0
Previous Algorithm	28	21	16	8	11	9	7	6	5	2
Our Algorithm	4	3	3	3	4	4	2	4	3	1
Improvements (times)	6.0	6.0	4.3	1.7	1.8	1.3	2.5	0.5	0.7	1.0

(a): Problem Size 3: Average Improvement = 2.6 (times)

Sample Problems	$I05_1$	$I05_2$	$I05_3$	$I05_4$	$I05_5$	$I05_6$	$I05_7$	$I05_8$	$I05_9$	$I05_{10}$
No. of Variables	10	7	6	4	4	3	2	1	1	0
Previous Algorithm	87	43	36	27	20	14	11	5	5	1
Our Algorithm	5	5	5	5	3	5	6	4	3	1
Improvements (times)	16.4	7.6	6.2	4.4	5.7	1.8	0.8	0.3	0.7	0.0

(b): Problem Size 5: Average Improvement = 4.4 (times)

Sample Problems	$I10_1$	$I10_2$	$I10_3$	$I10_4$	$I10_5$	$I10_6$	$I10_7$	$I10_8$	$I10_9$	$I10_{10}$
No. of Variables	20	14	10	8	7	4	4	3	1	0
Previous Algorithm	500	222	126	84	87	31	38	23	6	2
Our Algorithm	12	9	8	6	7	8	10	7	3	1
Improvements (times)	40.7	23.7	14.8	13.0	11.4	2.9	2.8	2.3	1.0	1.0

(c): Problem Size 10: Average Improvement = 11.4 (times)

Sample Problems	IMP30	IMP50	IMP100	IMP150	IMP300
Problem Sizes	30	50	100	150	300
No. of Variables	6	10	25	30	40
Previous Algorithm	212	382	4204	7249	5478
Our Algorithm	24	42	95	135	288
Improvements (times)	7.8	8.1	43.3	52.7	18.0

(d): Problem Sizes ≥30 (including 30, 50, 100, 150, 300)

C). For each of the cases (the integer domain and the real domain, $\mathbf{OP}_{\neg\neq}$ and $\mathbf{OP}_{all}$, satisfiability versus implication problems), excluding the two cases that have been shown NP-hard, we have either reported the first necessary and sufficient conditions together with their efficient algorithms with complexity analysis or provided improved algorithms. These results are extremely essential for designers and researchers of database and logic-based systems.

ACKNOWLEDGMENTS

We thank the anonymous referees who provided very helpful comments on this paper. In particular, we would like to thank one anonymous referee who pointed out a better bound for the implication problem under real domain using OP_{all}: $\min(|S|^{2.376} + |T|, |S|*|T|)$ (from our original $(|S|^{2.376} + |T|)$) following Theorem 2.2.

REFERENCES

AHO, A. V., HOPCROFT, J. E., AND ULLMAN, J. D. 1983. *Data Structures and Algorithms.* Addison-Wesley, Reading MA.

AHO, A. V., SAGIV, Y., AND ULLMAN, J. D. 1979a. Equivalences among relational expressions. *SIAM J. Comput. 8,* 2 (May), 218–246.

AHO, A. V., SAGIV, Y., AND ULLMAN, J. D. 1979b. Efficient optimization of a class of relational expressions. *ACM Trans. Database Syst. 4,* 4 (Dec.), 435–454.

ASTRAHAN, M. M. ET AL. 1976. System R: Relational approach to database management. *ACM Trans. Database Syst. 1,* 2 (June), 97–137.

BLAKELEY, J. A., COBURN, N., AND LARSON, P. A. 1986a. Updating derived relations: Detecting irrelevant and autonomously computable updates. In *Proceedings of the Eleventh International Conference on Very Large Databases* (Kyoto, Japan), 457–466.

BLAKELEY, J. A., LARSON, P. A., AND TOMPA, F. W. 1986b. Efficiently updating materialized views. In *Proceedings of the ACM SIGMOD International Conference on Management of Data,* 61–71.

CERI, S. AND PELAGATTI, G. 1984. *Distributed Databases—Principles and Systems.* McGraw-Hill, New York.

CHAKRAVARTHY, U., GRANT, J., AND MINKER, J. 1990. Logic-based approach to semantic query optimization. *ACM Trans. Database Syst. 15,* 2 (June), 162–207.

COPPERSMITH, D. AND WINOGRAD, S. 1987. Matrix multiplication via arithmetic progressions. In *Proceedings of the Nineteenth Annual ACM Symposium on the Theory of Computing,* 1–6.

CORMEN, T., LEISERSON, C., AND RIVEST, R. 1990. *Introduction to Algorithms.* MIT Press, Cambridge, MA.

FLOYD, R. W. 1962. Algorithm 97: Shortest path. *Comm. ACM 5,* 6 (June), 345.

GALLAIRE, H., MINKER, J., AND NICHOLAS, J. 1984. *Logic Databases: Deductive Approach 16,* 2.

GAREY, M., JOHNSON, D., AND STOCKMEYER, L. 1976. Some simplified NP-complete problems. *Theor. Comput. Sci. 1,* 237–267.

GUO, S., SUN, W., AND WEISS, A. M. 1996. On satisfiability, equivalence, and implication problems involving conjunctive queries in database systems. *IEEE Trans. Knowl. Data Eng.*

JARKE, M. AND KOCH, J. 1984. Query optimization in database systems. *ACM Comput. Surv. 16,* 2 (June), 111–152.

JOHNSON, D. AND KLUG, A. 1984. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. Comput. Syst. Sci. 28,* 1 (Feb.), 167–189.

JOHNSON, D. AND KLUG, A. 1983. Optimizing conjunctive queries that contain untyped variables. *SIAM J. Comput. 12,* 4 (Nov.), 616–640.

KIM, W. 1984. Global optimization of relational queries: A first step. In *Query Processing in Database Systems,* W. Kim, D. Reiner, and D. Batory, Eds., Springer, New York.

KING, J. J. 1986. Quist: A system for semantic query optimization in relational databases. In *Proceedings of the Seventh International Conference on Very Large Databases* (Kyoto, Japan), 510–517.

KLUG, A. 1988. On conjunctive queries containing inequalities. *J. ACM 35,* 1 (Jan.), 146–160.

KRISHNAMURTHY, R., BORAL, H., AND ZANIOLO, C. 1986. Optimization of nonrecursive queries. In *Proceedings of the Twelfth International Conference on Very Large Databases* (Kyoto, Japan), 128–137.

LOBO, J., MINKER, J., AND RAJASEKAR, A. 1992. *Foundations of Disjunctive Logic Programming,* MIT Press, Cambridge, MA.

MEGHINI, C. AND THANOS, C. 1991. The complexity of operations on a fragmented relation. *ACM Trans. Database Syst. 16,* 1 (March), 56–87.

ROSENKRANTZ, D. J. AND HUNT, H. B. III 1980. Processing conjunctive predicates and queries. In *Proceedings of the Sixth International Conference on Very Large Databases,* 64–72.

SELLIS, T. 1986. Global query optimization. In *Proceedings of the ACM SIGMOD International Conference on Management of Data,* 191–205.

SHASHA, D. AND WANG, T. 1991. Optimizing equijoin queries in distributed databases where relations are hash partitioned. *ACM Trans. Database Syst. 16,* 2 (June), 279–308.

STONEBRAKER, M. AND NEUHOLD, E. 1977. A distributed database version of INGRES. In *Proceedings of the Second Berkeley Workshop on Distributed Data Management and Computer Networks,* (May), 19–36.

SUN, X., KAMEL, N. N., AND NI, L. M. 1989. Processing implication on queries. *IEEE Trans. Softw. Eng. 15,* 10 (Oct.), 1168–1175.

SUN, W. AND WEISS, M. A. 1994. An efficient algorithm for testing implication involving arithmetic inequalities. *IEEE Trans. Knowl. Data Eng. 6,* 6 (Dec.), 997–1001.

SUN, W. AND YU, C. 1994. Semantic query optimization for chain and tree queries. *IEEE Trans. Knowl. Data Eng. 6,* 1 (Feb.), 136–151.

TARJAN, R. 1983. *Data Structures and Network Algorithms.* SIAM, Philadelphia, PA.

TARJAN, R. 1972. Depth-first search and linear graph algorithms. *SIAM J. Comput. 1,* 2 (June), 146–160.

ULLMAN, J. D. 1989. *Principles of Database and Knowledge-Base Systems, Volumes I & II.* Computer Science Press, New York, 885–892.

WEISS, M. A. 1995. *Data Structures and Algorithm Analysis* (second edition). Benjamin-Cummings, Redwood City, CA.

WONG, E. AND YOUSSEF, K. 1976. Decomposition—a strategy for query processing. *ACM Trans. Database Syst. 1,* 3 (Sept.).

YU, C. AND CHANG, C. 1984. Distributed query processing. *ACM Comput. Surv. 16,* 3.

YU, C. AND SUN, W. 1989. Automatic knowledge acquisition for semantic query optimization. *IEEE Trans. Knowl. Data Eng. 1, 3* (Sept.), 362–375.

Received February 1994; revised May 1995; accepted November 1995