ON CONJUNCTIVE QUERIES CONTAINING INEQUALITIES

by

Anthony Klug

Computer Sciences Technical Report #477

May 1982

On Conjunctive Queries Containing Inequalities

Anthony Klug

Computer Science Department

University of Wisconsin

Abstract

Conjunctive queries are generalized so that inequality comparisons can be made between elements of the query. Algorithms for containment and equivalence of such "inequality queries" are given. In general, containment does not imply the existence of homomorphisms (containment mappings), but the homomorphism property does exist for subclasses of inequality queries.

A minimization algorithm is defined using the equivalence algorithm. It is first shown that the constants appearing in a query can be divided into "essential" and "nonessential" subgroups. The minimal query can be nondeterministically guessed using only the essential constants of the original query.

Categories and Subject Descriptors: H.2.3[Database Management]: Languages - query languages

General Terms: Algorithms, Languages, Theory

Additional Keywords and Phrases: relational model, relational calculus, conjunctive query, tableau, inequality, equivalence, containment, minimization

---

Author's address: Computer Sciences Dept., University of Wisconsin-Madison, Madison, WI 53706. This work was supported in part by NSF grant MCS-8102864

1. Introduction

The relational data model has been widely accepted over the last decade as a way for looking at and interacting with databases. Codd's relational algebra and calculus [Codd] have been used as the formal basis for many real query languages (e.g., [SWKH], [DaGK]). At the same time, there has been much work in the theoretical aspects of relational database systems (see, e.g., [Ullm], [BeBG]). In particular, the problems of containment, equivalence, and minimization of queries have received considerable attention [AhSU78], [ChMe], [JoKl], [Sagi]. The equivalence problem is that of deciding whether two queries always retrieve the same data. The containment problem is to determine if the results of one query are always a subset of the results of another query. The minimization problem is the problem of finding a query equivalent to a given one and having the fewest possible number of joins.

These problems are important for a number of reasons:

Joins are among the most expensive relational operations to perform, and unnecessary joins may be present in a query due to the presence of views or the inexperience of the user, and so minimizing the number of joins can greatly decrease query execution cost.

Equivalence algorithms can be used for minimization (see section 4).

Containment can be used for dependency verification [YaPa].

For arbitrary relational calculus or relation algebra queries, these problems are undecidable [Solo]. However, a more restricted class of queries, the class of conjunctive queries, (or tableaux) has been studied by Chandra and Merlin [ChMe], by Aho, Sagiv, and Ullman [AhSU78] [AhSU79], by Sagiv [Sagi], by Sagiv and Yannakakis [SaYa], and by Johnson and Klug [JoKl1] [JoKl2]. Chandra and Merlin give an algorithm for the containment, equivalence, and minimization problems and show that the problems are NP-complete. Aho, Sagiv and Ullman give a polynomial-time algorithm for minimization and equivalence of "simple tableaux". Sagiv gives several polynomial-time algorithms for a number of other subclasses of tableaux. Johnson and Klug [JoKl1] look at another class, the "fanout free" queries and give polynomial-time algorithms for minimization and equivalence.

None of the classes of conjunctive queries studied so far have allowed inequality comparisons between data values. For example, a simple query such as, "get names of employees whose salaries are less than $18,000" is not included in any of the previously studied classes. Since such "inequality queries" are important and occur frequently in practice, it is desirable to consider the containment, equivalence and minimization problems for them; and this is what we do in this paper.

This paper is structured as follows: Section 2 contains the necessary definitions for relations, databases, and queries. Section 3 considers the containment problem, and Section 4 treats the minimization problem. Section 5 provides a summary and lists some problems for future consideration.

2. Basic Definitions

This section contains the basic definitions for relations, databases, and queries.

A relation R can be viewed as a finite two-dimensional table with columns labeled by distinct attributes. Each attribute A_i has a domain $D(A_i)$ and entries in a column labeled by by A_i must be elements of the domain $D(A_i)$. The relation scheme for a relation consists of the relation's name along with the sequence of attributes labeling its columns. Since the order of the rows in a relation has no significance, we can view a relation with scheme $R(A_1,A_2,\ldots,A_k)$ as a finite subset of the Cartesian product $D(A_1) \times D(A_2) \times \ldots \times D(A_k)$. For simplicity, we will assume that each domain $D(A_i)$ is the set $\mathbf{Q}$ of rational numbers. The rational numbers form a suitable generic data domain with respect to studying queries containing inequalities because they are totally ordered and dense. (String spaces also have these properties.) A database D is a finite sequence of tables, and the table in D for relation R is

denoted D_R. The relation schema for a database is the sequence of relation schemes for the tables it contains. In what follows we shall assume that all databases under consideration have the same relation schema[1].

A conjunctive query Q (of degree p) can be specified by the following: (1) A set $X_Q = \{x_1, \ldots, x_p\}$ of distinguished variables, the sequence $\langle x_1, \ldots, x_p \rangle$ being called the summary row, or just the summary; (2) a set $Y_Q = \{y_1, \ldots, y_q\}$ of non-distinguished (existentially quantified) variables; (3) a set $C_Q = \{c_1, \ldots, c_r\}$ of distinct conjuncts, each conjunct c_i being an atomic formula of the form $R(z_1, \ldots, z_m)$, where R is a relation name, and each z_i is a variable (either distinguished or nondistinguished); and (4) a set $L_Q = \{l_1, \ldots, l_u\}$ of inequalities, each inequality l_i being an atomic formula of the form $z_1 \ \theta \ z_2$, where z_1 and z_2 are either variables or constants (but not both constants), and θ is $=$, $<$, or $\leq$.

For later convenience, we will let K_Q denote the set of constants appearing in Q and CK_Q the set of constants of Q appearing in equalities of the form x = c. ("CK" stands for "conjunct constants" because such equalities can be eliminated by placing the constant c directly in the conjuncts

[1] As in [JoK11], we shall also assume that there are no data dependencies to be considered.

where x appears.)

Example 1. Given the schema

emp(eno,sal,dno)
dept(dno,status),

the query "give the names of employees who earn less than $25000 and who work in departments with status 10" corresponds to the following conjunctive query:

$$X = \{e\}$$
$$Y = \{s,d,t\}$$
$$C = \{emp(e,s,d),\ dept(d,t)\}$$
$$L = \{s<25000,\ t=10\}$$

From now on we will use a more intuitive set-theoretic notation for queries, and this example becomes:

$$\{e : (\exists s,d,t)(emp(e,s,d)\ \&\ dept(d,t)\ \&\ s < 25000\ \&\ t = 10)\}$$

□

When we wish to distinguish between the class of queries defined above and the original class of conjunctive queries, we will refer to the former as "inequality queries" and the latter as "equality queries." "Query" by itself will mean inequality query.

Inequality queries are equivalent to relational algebra expressions containing the operators selection (with inequality comparisons), projection, and join (including ine-

quality joins) [Codd] [ChMe]. To get the equivalent of the union operator, we use sets (or unions) of queries: A query set Y is a finite set $\{Q_1,\ldots,Q_k\}$ of queries of the same degree.

A query can be viewed as a function from databases to relations. To define the value of a query on a database we use the idea of a valuation assigning data values to variables in the query [Shoe]. A valuation ρ for a query Q is a function from $X_Q \cup Y_Q$ to **Q**. For convenience, valuations are also considered to be defined on constants c with $\rho(c)=c$ and on tuples $\langle z_1,\ldots,z_n\rangle$ with $\rho(\langle z_1,\ldots,z_n\rangle) = \langle\rho(z_1),\ldots,\rho(z_n)\rangle$. A valuation ρ is order preserving with respect to a query Q if for all inequalities $z_1 \Theta z_2$ in L_Q, $\rho(z_1) \Theta \rho(z_2)$ holds (under the usual ordering of **Q**). Then, given a database D and a conjunctive query Q, the relation constructed when Q is applied to D is

$$Q(D) = \{\rho(\langle x_1,\ldots,x_p\rangle) : \rho \text{ is an order preserving valuation wrt } Q \text{ such that for all conjuncts } R(z_1,\ldots,z_m) \text{ in } C_Q, \rho(\langle z_1,\ldots,z_m\rangle) \text{ is a tuple of } D_R\}$$

We will say that ρ gets the tuple $\langle\rho(x_1),\ldots,\rho(x_p)\rangle$ into Q(D).

Given a database D and a query set $Y = \{Q_1,\ldots,Q_k\}$, the relation constructed when Y is applied to D is

$$Y(D) = Q_1(D) \cup \ldots \cup Q_k(D).$$

Finally in this section we describe without proof a simple method for inferring new inequalities from a given set of inequalities:

Let L be a set of inequalities as above. We define the graph of L, denoted G(L), as follows:

(1) The nodes of G(L) are the variables and constants appearing in L.

(2) There is a directed arc from node z_1 to node z_2 if the inequality $z_1 < z_2$ or $z_1 \leq z_2$ is in L, or if z_1 and z_2 are both constants and z_1 is less than z_2. Such an arc is labeled "<" or "$\leq$" according to whether the inequality is strict (<) or nonstrict ($\leq$).

(3) If an equality $z_1 = z_2$ is in L, G(L) has a $\leq$-arc in each direction between z_1 and z_2.

The set L of inequalities derives an inequality l as follows: L derives the inequality $x \leq y$, where x and y are both variables, if G(L) has a directed path from x to y. The inequality $x < y$ is derived if there is a directed path from x to y having at least one <-arc. An inequality $c \leq x$, where c is a constant and x is a variable, is derived if there is a directed path from some constant c' to x and c is less than or equal to c'. An equality $c < x$ is derived if

there is a path from some constant c' to x, where $c \leq c'$, and the path has at least one <-arc or c is less than c'. Inequalities of the form $x < c$ and $x \leq c$ are derived in a similar fashion. An equality $z_1 = z_2$ is derived if both $z_1 \leq z_2$ and $z_2 \leq z_1$ are derived.

Using the above derivation rules, we can assume that the inequality sets for all queries are reduced. That is, all inequalities $z_1 \Theta z_2$ such that $L-\{z_1 \Theta z_2\}$ derives $z_1 \Theta z_2$ have been removed, and all pairs of inequalities $z_1 \leq z_2$, $z_2 \leq z_1$ have been replaced by equalities $z_1 = z_2$. We will also assume that all queries are consistent, that is, that they cannot derive contradictory inequalities such as $x < 2$ and $x > 3$. (The value of an inconsistent query is always the empty set.)

3. Containment and Equivalence

In this section we consider the containment and equivalence problems for inequality queries. Formally, given two queries Q_1 and Q_2, we say Q_1 is contained in Q_2, written $Q_1 \subseteq Q_2$, if for all databases D, $Q_1(D)$ is a subset of $Q_2(D)$. We say that Q_1 is equivalent to Q_2 if $Q_1(D) = Q_2(D)$ for all databases D. Our goal in this section is to find an algorithm for testing containment (which will also yield an algorithm for equivalence).

For equality queries Q_1 and Q_2, it has been shown that the containment $Q_1 \subseteq Q_2$ holds if and only if there is a homomorphism from Q_2 to Q_1 [ChMe], a homomorphism being a function on the symbols of Q_2 to those of Q_1 that maps distinguished variables to corresponding distinguished variables of Q_1, that is the identity on constants, and that induces a mapping from the conjuncts of Q_2 to those of Q_1. Proving the "if" part is quite simple. To prove the "only if" part, we are required to consider the query Q_1 itself to be a database. Formally, this is done by applying a one-to-one valuation ρ to Q_1. More precisely, $\rho(Q_1)$ is the database D such that for each relation R, D_R is the set of tuples $<\rho(z_1),\ldots,\rho(z_n)>$, for all conjuncts $R(z_1,\ldots,z_n)$ in C_{Q_1}. If $Q_1 \subseteq Q_2$, then $Q_1(\rho(Q_1)) \subseteq Q_2(\rho(Q_1))$. Let s_1 be the summary of Q_1. We always have $\rho(s_1) \in Q_1(\rho(Q_1))$ (ρ itself gets $\rho(s_1)$ into $Q_1(\rho(Q_1))$), so $\rho(s_1) \in Q_2(\rho(Q_1))$. The valuation which gets $\rho(s_1)$ into $Q_2(\rho(Q_1))$ becomes a homomorphism $Q_2 \rightarrow Q_1$ by composing it with ρ^{-1}.

If we were to attempt this same procedure for inequality queries, the one-to-one valuation ρ would assign unique rational numbers to the variables in Q_1. Since the set of rational numbers is totally ordered, every pair of distinct values in the database will be involved in a less-than relationship. That is, if $\rho(x) \neq \rho(y)$, then $\rho(x) < \rho(y)$ or $\rho(y) < \rho(x)$ regardless of what inequalities are in L_Q. These spurious inequalities can cause $\rho(s_1)$ to appear in

$Q_2(\rho(Q_1))$ even though containment fails:

Example 2. Consider the following two queries and a valuation on the first:

$$Q_1 = \{x : (\exists y,z) R(x,y,z)\}$$
$$Q_2 = \{x : (\exists y,z)(R(x,y,z) \ \& \ y < z)\}$$
$$\rho = \{x \to 0,\ y \to 1,\ z \to 2\}$$

We see that $\rho(s_1) \in Q_2(\rho(Q_1))$. But the containment $Q_1 \subseteq Q_2$ does not, of course, hold. □

Thus a single "canonical" database does not provide a correct test of containment for inequality queries. The correct approach is to use several (exponentially many, in fact) representative databases in place of the single database $\rho(Q_1)$. Each representative database will represent a different allowable arrangement of variable assignments along the rational number line.

We begin the development of representative databases with some definitions and lemmas:

Two valuations ρ_1, ρ_2 are order equivalent with respect to a query Q if for all symbols z_1, z_2 in Q (including constants), $\rho_1(z_1) < \rho_1(z_2)$ if and only if $\rho_2(z_1) < \rho_2(z_2)$. Order equivalence (with respect to a given query) is clearly an equivalence relation, and we let W_Q be the set of all equivalence classes of order preserving valuations on Q. W_Q is finite. Also note that for order equivalent valuations

ρ_1 and ρ_2, $\rho_1(z_1)\ \Theta\ \rho_1(z_2)$ iff $\rho_2(z_1)\ \Theta\ \rho_2(z_2)$ for all symbols z_1 and z_2 and for Θ equal to $=$ or $\leq$.

Now, given query Q with $W_Q = \{E_1,\ldots,E_k\}$, let $\{r_1,\ldots,r_k\}$ be a set of representatives, one from each respective equivalence class. A representative database set (r.d.s.) for Q is the set of databases $\{r_1(Q),\ldots,r_k(Q)\}$. We will assume there is some arbitrary selection rule which picks the representatives $\{r_1,\ldots,r_k\}$ for a given query, and unless other special choices are needed, we will refer to the r.d.s. and denote it D_Q.

The next lemmas provide some tools for the main theorem to follow which relates containment and r.d.s's.

Lemma 1. Let Q be a query, and let ρ_1, ρ_2 be order preserving and order equivalent valuations with respect to Q. Then there exists a strictly increasing function f on **Q** such that $f \circ \rho_1 = \rho_2$ and f is the identity on the constants in Q.

Proof. Note that the images $\rho_1(X_Q \cup Y_Q \cup K_Q)$ and $\rho_2(X_Q \cup Y_Q \cup K_Q)$ have the same number of elements; say the former is $A = \{a_1,\ldots,a_m\}$, and the latter is $B = \{b_1,\ldots,b_m\}$, where the indices are such that $i < j$ implies $a_i < a_j$ and $b_i < b_j$. Because order equivalence takes into account the constants in the query, if c is a constant in Q and $c = a_j$, then $c = b_j$. Begin the definition of f by specifying $f(a_i) = b_i$ for $i=1,\ldots,m$. Now simply extend the domain of f to all of **Q** in a piecewise-linear fashion. Each

piece of f has a positive slope. Also, since all constants c in Q appear in the same position in A and B, f(c) = c. □

Lemma 2. Let Q be a query, D a database, and f a strictly increasing function on **Q** which is the identity on constants in Q. Then f(Q(D)) = Q(f(D)).

Proof. If $t \in f(Q(D))$, then $t=f(u)$ for some $u \in Q(D)$. Let ρ be the valuation which gets u into Q(D). Then we have $f(u) = f(\rho(s))$, $f(\rho(<z_1,\ldots,z_n>)) \in f(D_R)$ for all conjuncts $R(z_1,\ldots,z_n)$ in C_Q, and $f(\rho(x)) \ \theta \ f(\rho(y))$ for all $(x\theta y)$ in L_Q, the last condition holding because f is strictly increasing. But this means that $f\circ\rho$ gets f(u) into Q(f(D)).

Conversely, if $t \in Q(f(D))$, then there is a valuation ρ such that $t = \rho(s)$, $\rho(<z_1,\ldots,z_n>) \in f(D_R)$ for all conjuncts $R(z_1,\ldots,z_n)$ in C_Q, and $\rho(x) \ \theta \ \rho(y)$ for all $(x\theta y)$ in L_Q. Since f is strictly increasing, f^{-1} is also strictly increasing. Thus we get $f^{-1}(t) = f^{-1}(\rho(s))$, $f^{-1}(\rho(<z_1,\ldots,z_n>)) \in D_R$ for all conjuncts $R(z_1,\ldots,z_n)$ in C_R, and $f^{-1}(\rho(x)) \ \theta \ f^{-1}(\rho(y))$ for all $(x\theta y)$ in L_Q. In addition, $f^{-1}\circ\rho$ is a legal valuation since ρ and f^{-1} both map constants of Q_1 to themselves. Hence $f^{-1}(t) \in Q(D)$, and $t = f(f^{-1}(t)) \in f(Q(D))$. □

Lemma 3. All inequality queries are monotonic. That is, if database D_1 is contained in database D_2 (relation by relation), then for any query Q, $Q(D_1) \subseteq Q(D_2)$.

Proof. Easy.

We are now ready to give the fundamental theorem relating containment of inequality queries to a (testable) property of r.d.s.'s:

Theorem 1. Let Q_1, Q_2 be queries. Then $Q_1 \subseteq Q_2$ if and only if for each $D_i = r_i(Q_1)$ in D_{Q_1}, $r_i(s_1) \in Q_2(D_i)$, where s_1 is the summary of Q_1.

Proof. (=>) If $Q_1 \subseteq Q_2$, then for each D_i in D_Q, $Q_1(D_i) \subseteq Q_2(D_i)$. Since it is also the case that $r_i(s_1) \in Q_1(D_i)$ by using the identity valuation, we get $r_i(s_1) \in Q_2(D_i)$.
(<=) Let D be any database, and suppose $t \in Q_1(D)$. Let ρ be a valuation which gets t into $Q_1(D)$. We may consider the image $\rho(Q_1)$ itself to be a database D' (a subset of D), and we still have $t \in Q_1(D')$ by the same valuation. Since ρ is order preserving wrt Q_1, $\rho \in E_i$ for some equivalence class E_i in W_{Q_1}. Lemma 1 gives us a strictly increasing function f such that $\rho = f \circ r_i$. By our hypothesis, $r_i(s_1) \in Q_2(D_i)$, so by the lemmas above, $t = \rho(s_1) = f(r_i(s_1)) \in f(Q_2(D_i)) = Q_2(f(D_i)) = Q_2(f(r_i(Q_1))) = Q_2(\rho(Q_1)) = Q_2(D') \subseteq Q_2(D)$. □

Theorem 1 provides us with an algorithm for containment and equivalence between inequality queries. However, since there can be exponentially many representative databases in D_Q to check, this algorithm is not in the complexity class NP. (We do know it is NP-hard since it generalizes an NP-hard problem: containment for equality conjunctive queries [ChMe].) The containment problem could be shown to be in NP

if we had the following property holding: $Q_1 \subseteq Q_2$ if and only if there exists a homomorphism $h:Q_2 \rightarrow Q_1$, a homomorphism in this case being a function on the symbols of Q_2 to those of Q_1 that maps distinguished variables to corresponding distinguished variables of Q_1, that induces a mapping from the conjuncts of Q_2 to those of Q_1, and that sends inequalities in L_{Q_2} to inequalities derivable from L_{Q_1}. Such a symbol mapping can be guessed in polynomial time, and each of the required properties can be checked in polynomial time. While this homomorphism property holds for equality queries (see the beginning of this section), only the "if" part holds in general for inequality queries:

Lemma 4. For any queries Q_1 and Q_2, if there is a homomorphism $h:Q_2 \rightarrow Q_1$, then $Q_1 \subseteq Q_2$.

Proof. The idea is that if a valuation ρ gets a tuple into $Q_1(D)$, then the composition $\rho \circ h$ will be a valuation getting t into $Q_2(D)$. □

A counterexample to the converse of this lemma is given by the next example.

Example 3. Consider the queries Q_1 and Q_2 as follows:

$$Q_1: \{u : (\exists x,y,z)(S(u) \ \& \ R(x,y) \ \& \ R(y,z) \ \& \ x < z)\}$$
$$Q_2: \{u : (\exists x,y)(S(u) \ \& \ R(x,y) \ \& \ x < y)\}$$

If there are two tuples (a,b) and (b,c) in R such that $a < c$, then either $a < b$ or else $a \geq b$ and $b \leq a < c$.

Hence, $Q_1 \subseteq Q_2$, but every mapping of the variables of Q_2 to those of Q_1 will either fail to be a conjunct mapping or will not map the inequality in Q_2 to an equality derivable from L_{Q_1}. We must sometimes map the conjunct of Q_2 to the first conjunct of Q_1 and sometimes to the second. □

The question now arises of which subclasses of inequality queries possess the homomorphism property. We know the equality queries have it. We will show that the homomorphism property also holds for two other subclasses: the left semi-interval queries, where inequalities are all of the form $x \Theta c$, where x is a variable and c is a constant, and the right semi-interval queries, where inequalities are all of the form $c \Theta x$, where x is a variable and c is a constant. In addition, it will be convenient to assume that a query in one of these classes does not contain two equalities $x = c$, $y = c$ (or $c = x$, $c = y$) where x and y are distinct variables.

Theorem 2. The homomorphism property holds for left and right semi-interval queries.

Proof. We will give a proof only for left semi-interval queries since the proof for right semi-interval queries is analogous. In light of the previous lemma, we also only need to show that containment implies the existence of a homomorphism.

Suppose Q_1 and Q_2 are left semi-interval queries and $Q_1 \subseteq Q_2$. Since L_{Q_1} is reduced, any variable will occur in at most one inequality of L_{Q_1}. We can therefore find a valuation ρ on Q_1 with the following properties: (a) $\rho(x) \ \theta \ c$ holds for all inequalities $(x \ \theta \ c)$ in L_{Q_1}, and $\rho(x) > \max\{c' \in K_{Q_1} \cup K_{Q_2} : c' < c\}$, (b) ρ is one-to-one, and (c) for all variables x not appearing in L_{Q_1}, $\rho(x) > \max(K_{Q_1} \cup K_{Q_2})$.

Note that for any variable x of Q_1 and constant c of Q_1 or Q_2, $\rho(x) \ \theta \ c$ holds if and only if x appears in an inequality $x \ \theta \ c'$ of L_{Q_1} and $c' \leq c$. In other words, $\rho(x \theta c)$ is true if and only if $(x \ \theta \ c)$ is derivable in L_{Q_1}.

Now let $D = \rho(Q)$. We have $\rho(s_1) \in Q_1(D)$, so $\rho(s_1) \in Q_2(D)$. Let r be a valuation getting $\rho(s_1)$ into $Q_2(D)$, and consider the composition $\rho^{-1} \circ r$. We claim that this is the desired homomorphism. That $\rho^{-1} \circ r$ maps s_2 to s_1 and that it maps the conjuncts of Q_2 to those of Q_1 is easy to see. Let $(x \ \theta \ c)$ be in L_{Q_2}. We have $r(x) \ \theta \ c$ holding in D, but we can rewrite this inequality as $\rho(\rho^{-1}(r(x))) \ \theta \ c$. As the note above states, this means that $\rho^{-1}(r(x)) \ \theta \ c$ is derivable from L_{Q_1}. □

The following example shows how the homomorphism property fails for a class of queries somewhat more general

than the semi-interval queries.

Example 4. Consider the following queries (which are also represented more pictorially in Figure 1):

$$Q_1: \{u : (\exists a,b,c,d,e)(S(u) \ \&\ R(a,b) \ \&\ R(c,b) \ \&\ R(c,d) \ \&\ R(e,d) \ \&\ 0 \leq a \ \&\ a \leq 3 \ \&\ 8 \leq e \ \&\ e \leq 9 \ \&\ 4 \leq c \ \&\ c \leq 6)\}$$

$$Q_2: \{u : (\exists x,y,z)(S(u) \ \&\ R(x,y) \ \&\ R(z,y) \ \&\ 0 \leq x \ \&\ x \leq 5 \ \&\ 5 < z \ \&\ z \leq 9)\}$$

These queries are typed [JoKl1]: each variable appears under just one attribute of a relation; the inequalities only specify intervals (there are no inequalities of the form $x\Theta y$, where x and y are variables); and the intervals in each query are disjoint. However, there are two symbol mappings $Q_2 \rightarrow Q_1$ that induce conjunct mappings:

h_1: $x \rightarrow a$, $y \rightarrow b$, $z \rightarrow c$, and
h_2: $x \rightarrow c$, $y \rightarrow d$, $z \rightarrow e$.

For h_1, the image of $(5 < z)$ is $(5 < c)$ which is not derivable in L_1; and for h_2, the image of $(x \leq 5)$ is $(c \leq 5)$ which is also not derivable. However, the containment $Q_1 \subseteq Q_2$ does hold. □

Of course, the failure of the homomorphism property does not imply that the problem is not in NP, but it does mean that a different approach must be taken if an NP algorithm is to be found for containment of inequality queries.

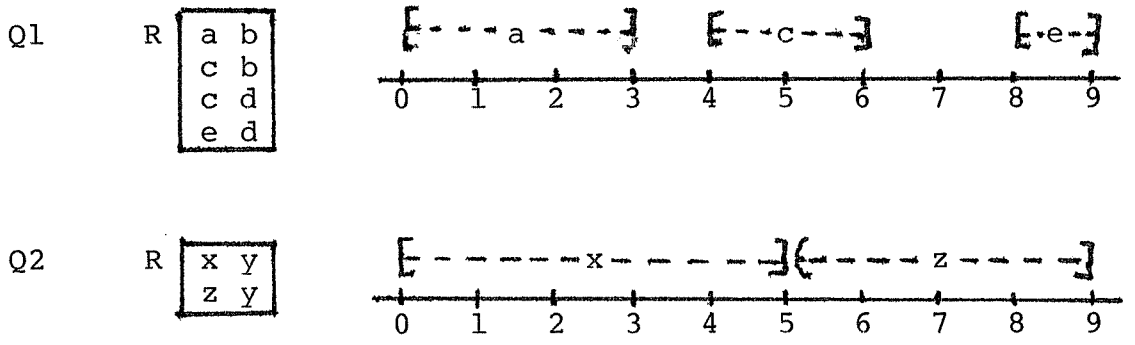

Figure 1.

Containment for Query Sets

The containment problem for query sets immediately reduces to the problem of containment of a query Q in a query set Y: $Y_1 \subseteq Y_2$ if and only if $Q \subseteq Y_2$ for each Q in Y_1. In [SaYa], it was shown that if Q is an equality query and Y is a set of equality queries, then $Q \subseteq Y$ if and only if there is a Q' in Y such that $Q \subseteq Q'$. We can think of this result as saying that equality queries "overlap" only in trivial ways. The result follows from the existence of a single canonical database for equality queries. Since inequality queries do not have single canonical databases (rather, representative database sets), we are not surprised that this containment property fails for them:

Example 5. Consider the queries Q_1,Q_2, and Q_3 defined as follows:

Q_1: $\{x : (\exists y)(R(x,y) \ \& \ S(y) \ \& \ T(y) \ \& \ 4<y \ \& \ y<9)\}$
Q_2: $\{x : (\exists y)(R(x,y) \ \& \ S(y) \ \& \ y<7)\}$
Q_3: $\{x : (\exists y)(R(x,y) \ \& \ T(y) \ \& \ 6<y)\}$

We have $Q_1 \subseteq Q_2 \cup Q_3$, but tuples in Q_1 are sometimes in Q_2 and sometimes in Q_3, i.e., neither the containment $Q_1 \subseteq Q_2$ nor the containment $Q_1 \subseteq Q_3$ holds. □

The containment problem for a query in a query set can still be solved, however, and the solution is a straightforward extension of Theorem 1:

Theorem 3. Let Q be a query and let Y be a query set. Then $Q \subseteq Y$ if and only if for each D_i in D_Q there is a Q' in Y such that $r_i(s) \in Q'(D_i)$.

Proof. Left to the reader. □

Since the homomorphism property does hold for left and right semi-interval queries, we might expect better behavior for query sets over these classes. This is correct:

Theorem 4. If $Q, Q_1, \ldots, Q_k$ are left (right) semi-interval queries, then $Q \subseteq Q_1 \cup \ldots \cup Q_k$ if and only if for some i, $Q \subseteq Q_i$.

Proof. Construct a valuation ρ and database D as in Theorem 2. In place of the constant set $K_{Q_1} \cup K_{Q_2}$, use the set $K_Q \cup K_{Q_1} \cup \ldots \cup K_{Q_k}$. □

4. Minimization

In the relational data model, it is easy to express queries whose implementation can be quite expensive. Thus it is worthwhile to find a method for converting queries into equivalent ones with more efficient implementations. For conjunctive (equality and inequality) queries, the main factor in the efficiency of the implementation is the number of "joins" contained in the query. The the query minimization problem therefore becomes the problem of finding an equivalent query involving the minimum possible number of joins. Since the number of joins is one less than the number of conjuncts, query minimization is the problem of removing unnecessary conjuncts.

The first result on minimization of conjunctive (equality) queries was given by Chandra and Merlin [ChMe]. They gave an algorithm for minimizing arbitrary equality queries, and they showed that the problem is NP-complete. Others ([AhSU78], [Sagi], [JoKl1]) have studied restricted classes of conjunctive equality queries and have derived polynomial time algorithms for the minimization problem. In other variations, the minimization problem is actually more difficult than the original problem. For example, Johnson and Klug [JoKl2] have looked at query minimization in the presence of functional and inclusion dependencies. For these harder problems, even non-deterministic or exponential time solutions for minimization are welcomed. One kind of

solution is to generate a minimization algorithm from an equivalence algorithm. The idea is that to find a minimal version of a query Q, we "guess" a minimal query Q′ from a set K of queries equivalent to Q and having no more conjuncts than Q. In order for this procedure to define a minimization algorithm, the search space K must be finite and we must know that a minimal query exists somewhere in K. If the original query Q has m total occurrences of variables and constants, then we will not need more than m distinct variables in our supply of "raw materials" from which to form K. If we can show that we also do not need any more constant symbols than those occurring in Q, then we add these constants to get a complete and finite set of raw materials for K. We get K itself by forming all possible queries from the raw materials having no more conjuncts than Q.

Showing the existence of an algorithm for inequality query minimization thus boils down to showing that new constant symbols are never needed in looking for a cheaper query. In the case of equality queries, this is trivial to show: If $Q \equiv Q'$, then there are homomorphisms $f:Q \rightarrow Q'$ and $f':Q' \rightarrow Q$. The first homomorphism shows that the constants in Q are also in Q′, and the second one shows that the constants in Q′ are also in Q. Thus equivalent equality queries always have the same set of constants. Another way of putting this is that every constant in an equality query

contributes in some way to the results defined by the query. With inequality queries, however, not only have we lost the nice homomorphism property of equality queries, but, in fact, constants in inequality queries may occur in ways which add nothing to the query:

Example 6. Consider the queries Q_1 and Q_2 defined as follows:

$$Q_1: \{x : (\exists y)(R(x,y) \ \& \ y<2 \ \& \ y<3\}$$
$$Q_2: \{x : (\exists y,z)(R(x,y) \ \& \ R(x,z) \ \& \ y<5 \ \& \ z<6)\}$$

In Q_1, the second inequality (and therefore the constant 3) is not needed. In Q_2, the constant 6 (and the inequality $z<6$ and the conjunct $R(x,z)$) do not contribute anything to the query. While the first case is handled by having removed redundancies from L_{Q_1}, the second query shows that the reason for a constant being unnecessary may involve the conjuncts rather than redundancies in L_Q. □

Because of these inconvenient ways in which constants may appear in inequality queries, most of this section does not deal with minimization per se but with the problem of determining what set of constants are necessary in guessing minimal queries. Once we have solved this problem, we will have a minimization algorithm as indicated above.

We will introduce the concepts of essential and nonessential constants. Essential constants will be ones

which cannot be varied or removed without altering the meaning of the query. Nonessential constants will be ones which can be varied within limits without changing the query at all. Our strategy is the following: We ignore constants which appear in equalities x = c (the members of CK_Q), since we will show that they are always essential. For a constant c which does not appear in CK_Q, we define an interval Δ_c of $\mathbb{Q}$ such that either c may replaced by all values in Δ_c without affecting Q, or any substitution of a value in Δ_c for c changes the meaning of Q. In the former case, c is nonessential, and in the latter case, c is essential. The endpoints of Δ_c are other constants appearing in Q (or $\pm\infty$), and we effect the removal of a nonessential constant c by replacing it by an endpoint of Δ_c.

This procedure for determining essential and nonessential constants may seem overly complicated when compared the more obvious one where a constant c is said to be nonessential in query Q when Q minus all inequalities containing c is still equivalent to Q. However, this simpler definition does not help to show that two equivalent queries must have the same set of essential constants. One may think of our approach as being more useful because it is more "gradual": we remove a nonessential constant by "sliding" it into an adjacent constant.

We now start with some definitions.

Given a query Q and constants c and c′, we write Q[c→c′] to denote the query obtained from Q by making the following replacements:

(1) If $c = c'$, do nothing.

(2) If $c' < c$, replace inequalities of the form $x < c$ or $x \leq c$ by $x \leq c'$, and inequalities of the form $c < x$ or $c \leq x$ by $c' < x$.

(3) If $c < c'$, replace inequalities of the form $c < x$ or $c \leq x$ by $x \leq c'$, and inequalities of the form $x < c$ or $x \leq c$ by $x < c'$.

(4) Replace equalities $x = c$ by $x = c'$.

Note that c need not appear in Q, in which case Q[c→c′] is the same as Q. In any case, c does not appear in Q[c→c′].

The reasons for replacing some strict inequalities by non-strict inequalities and vice versa will become apparent in the proofs.

We will write $Q_1 \approx> Q_2$ if Q_2 is of the form $Q_1[c \to c']$ for some constants c,c′ and if one of the following conditions holds:

(1) c does not appear in Q_1 or $c = c'$ (so that $Q_1 = Q_2$).

(2) $c' < c$ and there is no variable x such that $c' < x \leq c$ can be derived from L_1.

(3) $c < c'$ and there is no variable x such that $c \leq x < c'$ can be derived from L_1.

The canonical interval, $\Delta_{c,Q}$ for a constant c with respect to a query Q is defined as follows:

(1) If $\{c' \in K_Q : c'<c\}$ is empty, then the lower bound for $\Delta_{c,Q}$ is $-\infty$; otherwise the lower bound is $c_1 = \max\{c' \in Q : c'<c\}$. If c_1 exists and $Q \approx> Q[c \rightarrow c_1]$, then $\Delta_{c,Q}$ is closed on the left.

(2) If $\{c' \in K_Q : c<c'\}$ is empty, then the upper bound for $\Delta_{c,Q}$ is ∞; otherwise the upper bound is $c_2 = \min\{c' \in Q : c<c'\}$. If c_2 exists and $Q \approx> Q[c \rightarrow c_2]$, then $\Delta_{c,Q}$ is closed on the right.

When Q is understood, we will write Δ_c for $\Delta_{c,Q}$.

It is easy to see that $Q \approx> Q[c \rightarrow c']$ for every $c' \in \Delta_c$. We can think of going from Q to $Q[c \rightarrow c']$ as a perturbation of the query, and we will be determining those constants c whose values can be perturbed without affecting the query. For our results, it will be important to know that the same perturbation applied to a pair of equivalent queries will preserve the equivalence. The next lemma and theorem prove this property.

Lemma 5. Let c be a constant not in CK_Q (and not necessarily in K_Q), and let c′ be any constant in Δ_c. Let $Q' = Q[c \rightarrow c']$. We can choose the r.d.s.'s $D_Q = \{D_1,\ldots,D_m\}$ and $D_{Q'} = \{D_1',\ldots,D_k'\}$ so that for all i=1,...,k, $D_i' = D_i$, and D_i' contains no value in the semi-open interval [c,c′) (or (c′,c] if $c' \leq c$).

Proof. We will assume that $c' > c$. For $c' = c$, $Q = Q'$, the interval is empty, and there is nothing to prove; and for $c' < c$, the situation is analogous to what we do.

Let E_i' be an equivalence class in $W_{Q'}$, and let r be a member of E_i'. It is possible to obtain a valuation r_i from r by "sliding" all values in the range of r lying in the interval [c,c′) to the left out of the interval (see Figure 2). Note that, because there are no constants in K_Q between c′ and c, and because c does not occur in Q′, r_i is still in the same equivalence class E_i' and is order preserving with respect to Q′. We will show that r_i is also order preserving with respect to Q.

For an inequality (a Θ b) in Q where $a \neq c$ and $b \neq c$, the corresponding inequality in Q′ is still (a Θ b). The valuation r satisfies this inequality, and so r_i also satisfies it since the relative order of the r-values has not changed.

By assumption, equalities of the form (x = c) do not occur in Q.

Now consider an inequality (x Θ c) in Q, where Θ is <

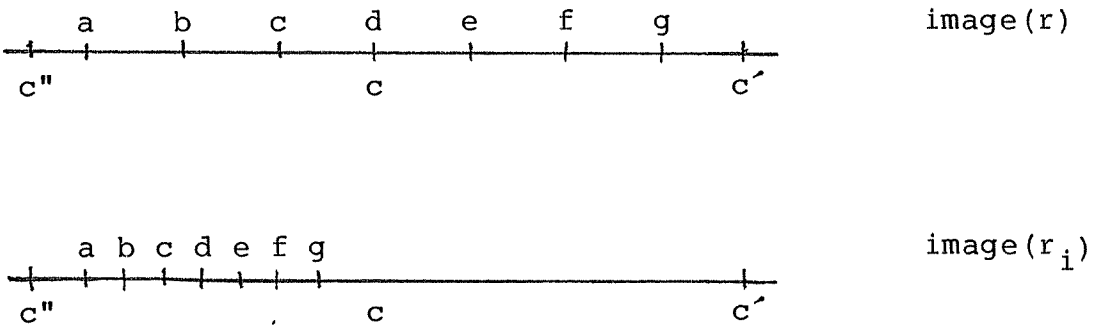

Figure 2.

or $\leq$. The corresponding inequality ($x < c'$) in Q' is satisfied by r, and $r(x) < c'$ implies that $r_i(x) < c$ since r_i has no value in $[c,c')$, so ($x \Theta c$) is satisfied by r_i.

Finally consider an inequality ($c \Theta x$) in Q, where Θ is $<$ or $\leq$. The corresponding inequality in Q' is ($c' \leq x$). Anything at c' or to the right of c' is also to the right of c and $r(x) = r_i(x)$, so $c \Theta r_i(x)$.

We have shown that in each E_i' in $W_{Q'}$ there is a valuation r_i which is also order preserving with respect to Q (i.e., which is in some E_j in W_Q) and whose image does not intersect the interval $[c,c')$. In fact, every E_j in W_Q has either zero or one of these special valuations r_i. This is because being order equivalent with respect to Q and having no values in the interval $[c,c')$ implies being order equivalent with respect to Q'. Thus we may assume that W_Q and $W_{Q'}$ are consistently indexed, i.e., that W_Q has n members, $W_{Q'}$ has $m \leq n$ members, and that $r_i = r_i'$ for

$i = 1,\ldots,m$. Using these representatives, we get $D_i = D_i'$ for $i=1,\ldots,m$, and by construction, these databases have no values in the interval $[c,c')$. □

Theorem 5. Let Q_1 and Q_2 be queries, and let c and c' be constants (not necessarily in Q_1 or Q_2) with c not in CK_{Q_1} or CK_{Q_2}. If $Q_1 \equiv Q_2$, $Q_1 \approx> Q_1[c\rightarrow c']$, and $Q_2 \approx> Q_2[c\rightarrow c']$, then $Q_1[c\rightarrow c'] \equiv Q_2[c\rightarrow c']$.

Proof. Let Q_1' be $Q_1[c\rightarrow c']$, and Q_2' be $Q_2[c\rightarrow c']$, and assume that $c < c'$. (The case $c = c'$ is trivial, and the case $c' < c$ is analogous to the one we consider.) We will only show the containment $Q_1' \subseteq Q_2'$; the other one is analogous.

We assume that D_{Q_1} and $D_{Q_1'}$ are as in the previous lemma. We show that $r_i(s_1') \in Q_2'(D_i')$ for each D_i' in $W_{Q_1'}$.

Since r_i also generates D_i for Q_1, and since $Q_1 \subseteq Q_2$, we have $r_i(s_1) \in Q_2(D_i)$. Let ρ be the valuation which gets $r_i(s_1)$ into $Q_2(D_i)$. We will show that ρ also gets $r_i(s_1)$ into $Q_2'(D_i)$. Since the conjuncts and the summary of Q_2' are the same as those of Q_2, ρ maps $C_{Q_2'}$ to tuples of D_i and s_2' to $r_i(s_1)$. We must only show that ρ is order preserving with respect to Q_2'.

Consider an inequality $(a \Theta b)$ of Q_2, where $a \neq c$ and $b \neq c$. The corresponding inequality in Q_2' is still $(a \Theta b)$ and is therefore satisfied by ρ.

For an inequality of Q_2 of the form $(x \Theta c)$, where Θ is

$<$ or $\leq$, $\rho(x) \leq c < c'$, so ρ also satisfies the corresponding inequality $(x < c')$ in Q_2'.

For an inequality of Q_2 of the form $(c \ \Theta \ x)$, where Θ is $<$ or $\leq$, $\rho(x) \geq c$. Since there are no values in the interval $[c,c')$, we know $\rho(x) \geq c'$. But the corresponding inequality in Q_2' is $(c' \leq x)$, so ρ satisfies it.

L_{Q_2} has no equalities of the form $x = c$.

All members of $L_{Q_2'}$ are generated by one of the above cases, so ρ is order preserving with respect to Q_2'. This verifies that $r_i(s_1') = r_i(s_1)$ is in $Q_2'(D_i')$. □

The first use of Theorem 5 is to show that when a constant c is replaced by other constants in Δ_c, either the query never changes in meaning, or else every substitution has a different meaning.

Theorem 6. Let Q be a query with c not in CK_Q.

(i) If there is a constant $c'' \neq c$ in Δ_c such that $Q \equiv Q[c \to c'']$, then for all c' in Δ_c, $Q \equiv Q[c \to c']$.

(ii) If there is a constant $c'' \neq c$ in Δ_c such that $Q \not\equiv Q[c \to c'']$, then for all $c' \neq c$ in Δ_c, $Q \not\equiv Q[c \to c']$.

Proof.

(i) Let c' be any other constant in Δ_c. Since $Q \equiv Q[c \to c'']$, and since $Q \approx\!> Q[c \to c']$ and $Q[c \to c''] \approx\!> Q[c \to c''][c \to c']$ (they are equal), we have by Theorem 5 $Q[c \to c'] \equiv Q[c \to c'']$.

(ii) This is just the contrapositive of (i). □

If we get condition (i) above holding, then we will say that constant c is non-essential. Any other constants in Q will be termed essential. Note that the theorem provides an algorithm for testing the property of being essential or nonessential, since we need only check if $Q \equiv Q[c \rightarrow c']$ for any one arbitrary $c' \neq c$ in the interval Δ_c.

It will be convenient to introduce some notation for these two classes of constants: EK_Q will denote the essential constants in query Q, and NK_Q will denote the nonessential constants. There is actually one detail to check regarding this definition: that EK_Q is a union of CK_Q and the constants described in (ii) of the previous theorem:

Lemma 6. For any query Q and constant $c \in CK_Q$, $Q \not\equiv Q[c \rightarrow c']$ for any $c' \neq c$ in Δ_c.

Proof. Let r be a valuation such that r(x) = c only for variables x such that (x = c) is in L, and let D be r(Q). Let c' be a constant lying between c and and the first image value r(y) greater than c (if any). Then c' is in Δ_c, Q(D) is not empty, but $Q[c \rightarrow c'](D)$ is empty because x = c' is in $L_{Q[c \rightarrow c']}$, but since c' does not appear in D, no valuation ρ can satisfy $\rho(x) = c$. □

We next show that if the endpoints of Δ_c are not in the interval, then case (ii) of the last theorem applies.

Theorem 7. If either end of Δ_c is bounded but not closed, then there is a constant c' in Δ_c with $Q \not\equiv Q[c \to c']$.

Proof. Assume it is the left end that is bounded but not closed. By definition, there are variables $x_1,\ldots,x_n$ such that for each i=1,...,n, the inequalities $c_1 < x_i \leq c$. are derivable from L_Q. The variables of Q can be divided into five groups as follows (see Figure 3), where θ denotes < or $\leq$:

(1) A set $\{u_i\}$ such that no inequalities $u_i \theta c$ or $c_1 \theta u_i$ are derivable.

(2) A set $\{y_i\}$ such that inequalities $y_i \theta c$ are derivable, but no inequalities $c_1 \theta y_i$ are derivable.

(3) A set $\{z_i\}$ such that inequalities $c_1 \theta z_i$ are derivable, but no inequalities $z_i \theta c$ are derivable.

(4) A set $\{w_i\}$ such that $c_1 \leq w_i$ and $w_i \leq c$ are derivable, but $w_i < c$ are not derivable.

(5) The set $\{x_i\}$ defined above.

There may be arcs between these groups of variables, but as a consequence of the definitions of the groups, only certain kinds of such cross arcs can occur as indicated in the following table:

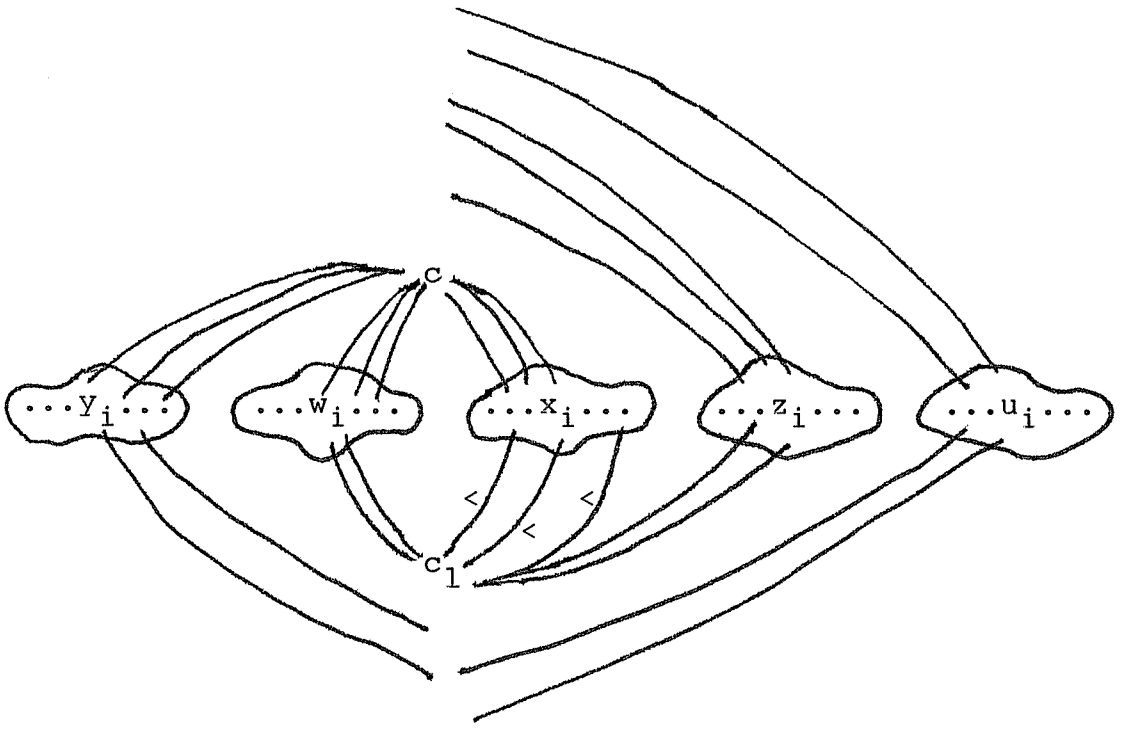

Figure 3.

Allowed	Not Allowed
$y_i \Theta w_j$	$w_j \Theta y_i$
$y_i \Theta x_j$	$x_j \Theta y_i$
$y_i \Theta z_j$	$z_j \Theta y_i$
$y_i \Theta u_j$	$u_j \Theta y_i$
$w_i \Theta x_j$	$x_j \Theta w_i$
$w_i \Theta z_j$	$z_j \Theta w_i$
	$w_i \Theta u_j$
	$u_j \Theta w_i$
$x_i \Theta z_j$	$z_j \Theta x_i$
	$x_i \Theta u_j$
	$u_j \Theta x_i$
$u_i \Theta z_j$	$z_j \Theta u_i$.

Thus it is possible to get an order preserving valuation ρ such that: $\rho(y_i)<c_1$ for all y_i; $\rho(w_i)=c_1$ for all w_i; $\rho(z_i)>c$ for all z_i; $\rho(u_i)>c$ for all u_i ($\rho(u_i)<c$ is also possible);

$p(x_i)<c$ for all x_i; $p(x_i)>k$ for all x_i, where k is a constant with $c > k > c_1$. Let D be $p(Q)$. Then $Q(D) \neq \emptyset$, but $Q[c\rightarrow k](D)$ is empty since each x_i in $Q[c\rightarrow k]$ has the constraint $c_1 < x_i \leq k$, but D does not even have any values in this interval. ⊟

Lemma 7. If c is the only constant in Q, then c is essential.

Proof. Left to the reader. ⊟

We now get to the main objectives of this section: The next theorem shows that nonessential constants can be eliminated by "sliding" them into adjacent constants. The theorem after that shows that equivalent queries always have the same set of essential constants.

Theorem 8.

(i) If c is nonessential in query Q, there is a query Q' with the same number of conjuncts as Q such that $Q' \equiv Q$ and $K_{Q'} = K_Q - \{c\}$.

(ii) For every query Q there is a query Q' such that $Q \equiv Q'$, Q' has no nonessential constants, and $|Q'| \leq |Q|$, where $|Q|$ denotes the number of conjuncts in Q.

Proof. (i) If c is nonessential, all queries in the set $\{Q[c\rightarrow c'] : c' \in \Delta_c\}$ are equivalent, and by the last theorem, both sides are either unbounded or closed. It is impossible for both sides to be unbounded (the previous lemma would

apply), so let c' be one of the endpoints. The Q' we want is $Q[c \to c']$.

(ii) Repeatedly apply part (i) until there are no more constants in NK_Q. ⊟

Theorem 9. If c is essential in Q and $Q \equiv Q'$, then c is essential in Q'. In other words, equivalent queries have the same set of essential constants.

Proof. Since c is essential in Q, no pair of queries in $\{Q[c \to c'] : c' \in \Delta_{c,Q}\}$ is equivalent. If c is nonessential in Q' or does not appear in Q', then all queries in $\{Q'[c \to c'] : c' \in \Delta_{c,Q'}\}$ are equivalent. Let c' be a constant in both $\Delta_{c,Q}$ and $\Delta_{c,Q'}$ and not equal to c. The equivalences $Q \equiv Q'$ (by assumption), $Q' \equiv Q'[c \to c']$ (by assumption), and $Q[c \to c'] \equiv Q'[c \to c']$ (by Theorem 5) contradict the assumption $Q \not\equiv Q[c \to c']$. ⊟

Putting the results above together, we get the following:

Theorem 10. For any query Q, a minimal query equivalent to Q lies within the set $\{Q' : Q' \equiv Q, |Q'| \leq |Q|, K_{Q'} = EK_Q\}$

Proof. Follows easily from the previous lemmas and theorems. ⊟

5. Summary and Future Work

Summary

In this paper we introduced the class of conjunctive inequality queries. These queries generalize the conjunctive queries of Chandra and Merlin [ChMe] and the tableaux of Aho, Sagiv, and Ullman [AhSU79] by including inequality comparisons among variables and constants. We showed how to test containment of a query Q_1 in another query Q_2 by a finite procedure over a finite set of representative databases. This containment problem is not known to be in NP, but we did identify some simple subclasses of inequality queries, the left and right semi-interval queries, whose containment problem is in NP. Containment for unions of conjunctive inequality queries can also be tested.

Minimization of a conjunctive query (of any kind) involves seeking to remove as many conjuncts (or tableau rows) as possible without changing the query's meaning. As long as we know that a minimal query will not need any constants other than those appearing in the original query, minimization can always be effected by simply "guessing" the minimal query and then using an equivalence algorithm to deterministically check that the meaning has not changed. For conjunctive queries having only equalities, it is trivial to show that all constants in the original query are needed in any minimal (or equivalent) query. Such is not the case for conjunctive queries having inequalities. In

section 4 we spent a considerable amount of time presenting a method for determining which constants in a query are essential to it. We also showed that in guessing a minimal query, only the essential constants are needed.

Future Work

There are a number of interesting problems left open by this work:

(1) What is the complexity of the containment problem? The algorithm we give does not run in even nondeterministic polynomial time. The problem is in PSPACE, however, and it is plausible that the containment problem is PSPACE-complete. First, there is no homomorphism property to provide short objects to guess. Secondly, it seems that all possible relative orderings of variable assignments (an exponential number) must be verified in order to verify containment.

(2) If the answer to (1) is "PSPACE-complete", i.e., "not likely to be in NP", then for what subclasses of the inequality queries is the containment problem in NP? (or in P?) We have shown that containment for left and right semi-interval queries is in NP by showing that the homomorphism property holds. For what other classes does the homomorphism property hold? Are there classes for which the homomorphism property does not hold but for which the containment problem is still in

NP?

(3) Given any query Q, can Q be minimized simply by removing conjuncts? That is, does a minimal query always exist within the "rows" of the given query? This problem is related to the existence of query homomorphisms. The property holds for equality queries [AhSU78] and the proof uses a composition of two homomorphisms. We have no homomorphism property for containment of inequality queries, but we also do not have examples of minimization which do not amount to simply removing rows. A related question is whether two minimizations of the same query are isomorphic. This has been shown to hold for equality queries [ChMe].

(4) What is an appropriate definition for minimization for unions of inequality queries? What are some algorithms for minimizing these unions? We can identify two reasonable definitions: (i) Since the number of joins needed to execute a query is directly proportional to the number of conjuncts, we should minimize the total number of the number of conjuncts in a union. (ii) We should minimize each component of the union separately and then minimize the number of components in the union. Are these definitions equivalent? Are there more appropriate concepts of minimality for unions of queries? For example, consider the union

$$\{x : R(x) \ \& \ x<5\} \cup \{x : R(x) \ \& \ x>7\}$$

One could think of evaluating this query by making one pass through relation R and evaluating the condition (x<5 OR x>7). Perhaps we should allow disjunctions of inequalities in inequality queries.

6. References

[ABCE] Astrahan M.M., Blasgen M.W., Chamberlin D.D., Eswaran K.P., Gray J.N., Griffiths P.P., King W.F., Lorie R.A., McJones P.R., Mehl J.W., Putzolu G.R., Traiger I.L., Wade B.W. and Watson V. "System R: Relational Approach to Database Management" ACM-TODS 1,#2, pp.97-137 (1976)

[AhSU78] Aho A.V., Sagiv Y. and Ullman J.D. "Efficient Optimization of a Class of Relational Expressions", ACM TODS, 4, 435-454 (1979)

[AhSu79] Aho A.V. Sagiv Y. and Ullman J.D. "Equivalences among Relational Expressions" SIAM J. Comptng. 8, 2, 218-246 (May 1979)

[BeBG] Beeri C., Bernstein P. and Goodman N. "A Sophisticate's Introduction to Database Normalization Theory" Proceedings VLDB Conf. 1978, West Berlin

[ChMe] Chandra A.K. and Merlin P.M. "Optimal Implementation of Conjunctive Queries in Relational Databases",

Proc. 9-th Annual Symp. on Theory of Computing, May, 1976, 77-90

[Codd] Codd E.F. "Relational Completeness of Data Base Sublanguages" Data Base Systems, R. Rustin (ed.), Prentice Hall, 1972

[DaGK] Dayal U., Goodman N., and Katz R. "An Extended Relational Algebra with Control over Duplicate Elimination", Proc. ACM Symp. on Principles of Database Systems, Los Angeles, 1982

[JoKl1] Johnson D. and Klug A. "Optimizing Conjunctive Queries When Attribute Domains are not Disjoint", Proc. 22nd Symp. on Foundations of Computer Science, Nashville, 1981

[JoKl2] Johnson D. and Klug A. "Testing Containment of Conjunctive Queries Under Functional and Inclusion Dependencies", Proc. ACM Symp. on Principles of Database Systems, Los Angeles, 1982

[Sagi] Sagiv Y. "Quadratic Algorithms for Minimizing Joins in Restricted Relational Expressions", Univ. of Ill. Computer Science Tech. Rep. UIUCDCS-R-79-992

[SaYa] Sagiv Y. and Yannakakis M. "Equivalences Among Relational Expressions with the Union and Difference Operators" Journal of the ACM, 27, pp. 633-655 (1980)

[Shoe] Shoenfield J.R. "Mathematical Logic", Addison-Wesley, Reading-London, 1967

[Solo] Solomon M.K. "Some Properties of Relational Expressions", ACM Southeast Regional Conference, April 1979, pp. 111-116.

[SWKH] Stonebraker M., Wong E., Kreps P. and Held G. "The Design and Implementation of INGRES", ACM-TODS 1, #3, 1976, pp.189-222

[Ullm] Ullman J.D. "A View of Directions in Relational Database Theory", Stanford Univ. Report STAN-CS-81-852

[YaPa] Yannakakis M. and Papadimitriou C. "Algebraic Dependencies", to appear