Testing Containment of Conjunctive Queries Under Functional and Inclusion Dependencies

(Extended Abstract)

D. S. Johnson
Bell Laboratories
Murray Hill, NJ 07974

A. Klug*
University of Wisconsin
Madison, WI 53706

ABSTRACT. We consider the problem of optimizing conjunctive queries in the presence of inclusion and functional dependencies. We show that the problem of containment (and hence those of equivalence and non-minimality) is in NP when either (a) there are no functional dependencies or (b) the set of dependencies is what we call *key-based*. These results assume that infinite databases are allowed. If only *finite* databases are allowed, new containments may arise, as we illustrate by an example. We also prove a "compactness" theorem that shows that no such examples can exist for case (b).

1. INTRODUCTION

The concept of an *inclusion dependency* in a relational database is a generalization of the Codd's notion of a "foreign key," as introduced in his seminal paper on the relational data model [6]. Together with other types of dependencies, such as functional dependencies, these dependencies provide a formal device for ensuring that databases model real-world entities and relationships.

Inclusion dependencies have been previously discussed (under various names) in [7,8,10,12,13] -- we take our terminology from Fagin [8]. An inclusion dependency (ID) says that values in columns of one relation must also appear as values in columns of some other relation. For example, suppose we are given relations:

EMP (employee number, salary, department)

DEP (department, location)

Then the ID "*EMP*(department) $\subseteq$ *DEP*(department)" says that the values in the department column of the *EMP* relation are all values in the department column of the *DEP* relation, i.e., that every department that has an employee also has a location. In this paper, we show how ID's affect the containment, equivalence, and minimization of conjunctive queries.

Conjunctive queries form a large and well-studied class of queries, containing a large proportion of those questions one might wish to ask in practice. When there are no constraints to consider, or when there are only functional dependencies, it has been shown that the containment, equivalence, and minimization problems are all NP-complete [5]. Such results are usually considered to be negative, since they show that the problems are as hard as problems for which it is widely conjectured no polynomial time algorithms exist. In this paper we shall emphasize a more positive point of view: Since so many problems in the area of relational databases are undecidable, provably exponential, or at least PSPACE-hard (e.g., see [3,4]), there is at least a bit of hope for a problem when it can be shown to be no *harder* than NP-complete.

This is especially true for problems concerned with query optimization, since queries tend to be very much smaller than the databases to which they are to be applied, and queries may be applied repeatedly over time. Thus an algorithm whose running time is exponential in query length may still pay for itself even if it only yields a relatively small improvement in the query. Furthermore, suppose, say, that the equivalence problem were in NP. Then it would be possible to give "short proofs" of equivalence, and a knowledge of the intended meaning of a query might on occasion help us to find such proofs quickly.

In this paper we consider the extent to which the "NP-easyness" of the above problems generalizes to the case where inclusion dependencies are present. It is conceivable that ID's make the problems much more difficult. As shown by Casanova, Fagin, and Papadimitriou [3], the combination of FD's with ID's is not finitely axiomatizable, and the inference problem for ID's alone is PSPACE-complete (whereas the inference problem for FD's by themselves is solvable in polynomial time). The potential effect of ID's on questions of query optimization is easily illustrated. Consider the following two conjunctive queries addressed to databases of the form specified above.

$$Q_1 = \{(e) : \text{There exist } s, d, l \text{ such that } EMP(e,s,d) \text{ and } DEP(d,l)\}$$

$$Q_2 = \{(e) : \text{There exist } s, d \text{ such that } EMP(e,s,d)\}$$

These two queries are equivalent if the ID "*EMP*(department) $\subseteq$ *DEP*(department)" holds, although they can give quite distinct answers if it does not.

Thus the presence of ID's gives rise to new and possibly more difficult problems. Attempts to extend the techniques and lemmas of [1,2,5] to handle ID's run into immediate difficulties since they seem to require the construction of infinite objects (as part of "chase" procedures). New methods would seem to be required.

In this paper we concentrate on the question of containment, since it is easy to show that whenever the containment problem is in NP, then so are the equivalence problem and the problem of determining whether a query is *not* minimal. We show that, assuming that the ID's and FD's satisfy certain reasonable conditions, a certain "compactness" result holds and the

*Work of this author partially supported by U.S. Army Contract #DAAG29-79-C-0165 and NSF Grant #MCS8102864

containment problem for conjunctive queries remains in NP. The compactness result can fail for sets of dependencies not satisfying the conditions, as is illustrated by a simple example containing just one FD and one ID. If one is willing to settle for the more restrictive notion of containment for *all* databases, including infinite ones, then the compactness result is not needed, and less stringent conditions on the dependencies will suffice. For instance, when there are no FD's, containment (over all databases) is in NP for *any* fixed finite set of ID's. (The fact that the problem is harder when restricted to finite databases is analogous to results on the inference problem for embedded implicational dependencies in [4]).

In Section 2 we review the basic definitions of the relational model, along with those for conjunctive queries, and present formal definitions of inclusion and functional dependencies. In Section 3 we show how to test for query containment (over *all* databases) in nondeterministic polynomial time, if the set of dependencies is fixed and either contains no FD's or is "key-based." In Section 4 we discuss the issue of containment for finite databases versus containment for *all* databases, and prove our compactness result, showing that for key-based dependencies, the two notions are the same. Section 5 then concludes with a discussion of open problems and directions for future research.

2. DEFINITIONS AND NOTATION

A *relation* R is a two-dimensional table with columns labeled by distinct *attributes* and a possibly infinite number of rows (or *tuples*). Each attribute A has a *domain* $D(A)$, and entries in a column labeled by by A must be elements of the domain $D(A)$. The *relation scheme* for a relation R is the sequence of attributes labelling its columns, and may be viewed as an ordered subset of the set of all attributes. Since the order of the rows in a relation has no significance, we can view a relation with scheme $(A_1,A_2,...,A_k)$ as a subset of $D(A_1) \times D(A_2) \times \cdots \times D(A_k)$. A *database* is a finite sequence of relations. The *relation schema* for a database is the sequence of relation schemes for the tables it contains.

A query Q can be viewed as a mapping from databases to relations. Given any database D that satisfies a specified *input relation schema*, it produces a set of tuples $Q(D)$ fitting an associated *output scheme*. If Q and Q' are queries with the same input schemas and output schemes, we say that Q is *infinitely contained* in Q', written $Q \subseteq_\infty Q'$, if for all appropriately formatted (and not necessarily finite) databases D, $Q(D) \subseteq Q'(D)$. We say that Q is *finitely* contained in Q', written $Q \subseteq_f Q'$, if the above holds for all appropriately formatted *finite* databases. It is, of course, this latter and presumably weaker form of containment that would be relevant in practice. Two queries Q and Q' would normally be considered to be equivalent if $Q \subseteq_f Q'$ and $Q' \subseteq_f Q$.

We formalize the way dependencies affect containment as follows: Suppose Σ is a set of dependencies. We then write $\Sigma \models Q \subseteq_\infty Q'$ ($\Sigma \models Q \subseteq_f Q'$), if $Q(D) \subseteq Q'(D)$ for all appropriately formatted (finite) databases D that obey the dependencies in Σ.

Our notation for functional and inclusion dependencies will be as follows: A *functional dependency* is a formal statement of the form $R: Z \rightarrow A$, where R is a relation name, Z is a set of attributes of R, and A is an attribute. A database *obeys* this FD if there are no tuples in relation R with identical Z-values and different A-values. An *inclusion dependency* is a formal statement of the form $R[X] \subseteq S[Y]$, where R and S are relation names, X is an ordered list of attributes of R, and Y is an ordered list of attributes of S of the same length as X. A database *obeys* the ID $R[J_1,...,J_j] \subseteq S[K_1,...,K_j]$ if for every subtuple $<a_1, \ldots, a_j>$ that occurs in columns $J_1,...,J_j$ of some tuple in relation R, there is a tuple of relation S that contains $<a_1,...,a_j>$ in columns $<K_1,...,K_j>$.

Conjunctive queries are discussed in detail in Chandra and Merlin [5], Aho, Sagiv, and Ullman [1,2], and Johnson and Klug [9]. In general we shall specify conjunctive queries informally, as we did for Q_1 and Q_2 in the previous section. However, it will also be useful to be able to view such a query Q as a formal object, with the following parts: (1) an input schema I_Q, (2) an output scheme $O_Q = (A_1, \ldots, A_p)$, (3) a set $X_Q = \{x_1,...,x_p\}$ of *distinguished variables* (DVs), (4) a set $Y_Q = \{y_1,...,y_q\}$ of *nondistinguished variables* (NDVs), and (5) a set $C_Q = \{c_1, \ldots, c_r\}$ of distinct *conjuncts*, each conjunct c_i being associated with a relation $R(c_i)$ of the input schema I_Q and having the form $(c_i[1],...,c_i[m])$, where $m = Length(c)$ is the number of columns (attributes) in $R(c_i)$ and each $c_i[j]$ is either a DV, an NDV, or a *constant*, i.e., an element of the domain of the j^{th} attribute of $R(c_i)$. The tuple $(x_1,...,x_p)$ is called the *summary row* of Q.

As an illustration of these definitions, recall the query Q_1 from the previous section:

$$Q_1 = \{(e): \text{There exist } s,d,l \text{ such that } EMP(e,s,d) \text{ and } DEP(d,l)\}$$

For this query the input schema is $<EMP,DEP>$, the output scheme is (employee number), the set of DV's is $\{e\}$, the set of NDV's is $\{s,d,l\}$, and the conjuncts are (e,s,d) and (d,l), with $R(e,s,d) = EMP$ and $R(d,l) = DEP$. The summary row is "(e)". There are no constants in this particular query.

Given a database B and a conjunctive query Q, the relation constructed when Q is applied to B is

$$Q(B) = \{(x_1,...,x_p) : x_i \in D(A_i) \text{ and there exist } y_1,...,y_q \text{ such that for all } c_i \in C_Q, \text{ the tuple } (c_i[1],...,c_i[m]), \text{ with the valuations of the DVs and NDVs substituted in, is a row in relation } R(c_i) \text{ of } B\}$$

An equivalent definition of $Q(B)$, which will prove useful in what follows, involves the notion of a *homomorphism* from a query Q to a database B. Let U_Q be the set of all symbols (DV's, NDV's, and constants) occuring in Q, and let D_B be the union of the domains of all the attributes occurring in B. A homomorphism from Q to B is a function $f: U_Q \rightarrow D_B$ that sends each constant to itself and induces a well-defined map from the conjuncts of Q to rows in the corresponding relations of B. A tuple $\bar{a} = (a_1, \ldots, a_p)$ is in $Q(B)$ if and only if there is a homomorphism f from Q to B under which the image of the summary row of Q summary row is $\bar{a}$, i.e., under which $f(x_i) = a_i$, $1 \le i \le p$.

3. CHASES AND CONTAINMENTS

A key concept in our containment algorithm is that of the *chase* of a conjunctive query with respect to a set of dependencies Σ [11]. The conjuncts of a query Q can be viewed as tuples in a database satisfying the query's input schema, where each variable is interpreted as a unique new constant. However, this collection of tuples may not qualify as a proper database if there is a set Σ of dependencies that proper databases must obey. The idea of the chase is to convert the conjuncts of a query into a proper database, possibly by coalescing distinct conjuncts, or adding new ones.

In the case when there are only FD's in Σ, each step of the chase consists of an application of the following *chase rule* for FD's:

> If $R: Z \rightarrow A$ is an FD in Σ and $c_1 = <z_1, \cdots, z_n>$ and $c_2 = <w_1, \cdots, w_n>$ are conjuncts with $R(c_1) = R(c_2) = R$, $c_1[Z] = c_2[Z]$, and $c_1[A] \neq c_2[A]$, we say that the given FD is *applicable* and identify the symbols $c_1[A]$ and $c_2[A]$, wherever they occur in the conjuncts and summary row of Q. The value for the combined symbol is determined as follows: If both were constants, delete all conjuncts from Q and halt (this query cannot be chased to an equivalent query obeying the given FD). If one is a constant, let the combined symbol be that constant. If both are variables, choose either variable to represent the combined symbol, except that a DV is preferred to an NDV.

After a finite amount of work, there will be no more applicable FD's and the chase will terminate. Depending on the choices made along the way, any one of a number of syntactically distinct queries might result from this process (although Maier et. al. [11] have shown that the result *is* unique up to a renaming of the variables). In what follows, we shall assume that we have specified an explicit, deterministic method for making the above choices, so that the resulting query is well-defined, even up to the names of its variables. We shall call this query $chase_\Sigma(Q)$ (Our results will hold no matter what method is specified).

In the case where Σ contains only FD's, it has been shown that $\Sigma \models Q \subseteq_f Q'$ if and only if there is a *homomorphism* of Q' to $chase_\Sigma(Q)$, i.e., a map h of the symbols of Q' to the symbols of $chase_\Sigma(Q)$ that leaves constants fixed, induces a well-defined map from the conjuncts of Q to the conjuncts of $chase_\Sigma(Q)$, and sends the summary row of Q' to the summary row of $chase_\Sigma(Q)$ (see [1,2,5,9]). This result relies on the fact that if there is a homomorphism from Q to a database D, then there is a homomorphism from $chase_\Sigma(Q)$ to D. In order to have this latter fact hold true when ID's are allowed, we are more-or-less forced to define the following chase rule for ID's:

> If $R[X] \subseteq S[Y]$ is an ID of Σ, c is a conjunct of Q with $R(c)=R$, and there is no conjunct c' such that $R(c')=S$ and $c'[Y]=c[X]$, we say the ID is *applicable* and add a new conjunct c'' to Q where $R(c'')=S$, $c''[Y]=c[X]$ and where $c''[A]$ is a distinct new symbol for each attribute A not in Y.

To construct $chase_\Sigma(Q)$ when Σ contains both FD's and ID's, one applies applicable FD's and ID's as follows: (1) If there is an applicable FD, apply it, (2) If a number of conjuncts have applicable ID's, choose one of minimum level, where the level of a conjunct from Q is 0 and the level of a conjunct generated by applying an ID to a level-i conjunct has level $i+1$. (Other ties are less significant and, as before, will be broken according to some arbitrarily chosen but prespecified rule).

It is easy to see that even such simple Σ's as the single ID "$\{R[2] \subset R[1]\}$" can give rise to infinite chases under the above rules. However, we do have the desired lemma and theorem (proofs are left to the reader):

Lemma 1. If D satisfies the dependencies of Σ and f is a homomorphism from a conjunctive query Q to D, then f can be extended to a homomorphism from $chase_\Sigma(Q)$ to D.

Theorem 1. If Q and Q' are conjunctive queries, then $\Sigma \models Q \subseteq_\infty Q'$ if and only if there is a homomorphism from Q' to $chase_\Sigma(Q)$.

Note that, since $chase_\Sigma(Q)$ might be infinite, Theorem 1 does not immediately yield an algorithm for testing containment, although it does show that the set $\{(\Sigma,Q,Q'): \Sigma \models Q \subseteq_\infty Q'\}$ is recursively enumerable. Our goal is to show that this set is actually in NP for fixed values of Σ. We have been able to show that this is true for two important types of Σ.

Let us call a set Σ of FD's and ID's *key-based* if (a) For a given relation R, the FD's $R: Z \rightarrow A$ all have the same left-hand-side Z, and every attribute of relation R which is not in Z is the right-hand-side of some FD for R, and (b) Each ID $R[X] \subseteq S[Y]$ has its right-hand-side Y contained in the left-hand-side of an FD for the relation S, and its left-hand-side X *disjoint* from the left-hand-sides of the FD's for the relation R. Note that property (a) implies that Z is a *key* for relation R [6].

Theorem 2. If Σ (i) consists entirely of ID's or (ii) is a set of key-based dependencies, then the problem "Given conjunctive queries Q, Q', does $\Sigma \models Q \subseteq_\infty Q'$?" is in NP.

Proof. We shall concentrate on case (i). The result for (ii) follows from that for (i) because of two easy observations:

(1) If $\Sigma[F]$ is a set of FD's, then for all conjunctive queries Q, $chase_{\Sigma[F]}(Q)$ is equivalent to Q for databases satisfying Σ [11], and

(2) If $\Sigma[I]$ is a set of ID's and $\Sigma = \Sigma[F] \cup \Sigma[I]$ is key-based, then for all conjunctive queries Q, $chase_\Sigma(Q) = chase_{\Sigma[I]}(chase_{\Sigma[F]}(Q))$.

Our plan-of-attack for case (i) will be to show that if there is a homomorphism $Q' \rightarrow chase_\Sigma(Q)$, then there is also a homomorphism from Q' to some initial segment of $chase_\Sigma(Q)$ whose depth is bounded by a polynomial function of the size of Q, the polynomial itself being determined by Σ. A nondeterministic polynomial time algorithm for testing containment would then consist of (1) Guessing the image of Q' under this bounded homomorphism, (2) Guessing enough of $chase_\Sigma(Q)$ to prove that the image is indeed part of $chase_\Sigma(Q)$, and (3) verifying that there is indeed a homomorphism from Q' to the guessed image.

To realize our plan-of-attack, we prove a series of three lemmas. These lemmas view $chase_\Sigma(Q)$ as a directed graph, with a vertex for each conjunct. The *roots* of this graph are the original conjuncts of Q. If an ID was applied to conjunct c_1 and generated c_2, there is an *ordinary* arc from c_1 to c_2. If $R(c_1)=R$ but the ID "$R[X] \subseteq S[Y]$" generated no new conjunct because the required conjunct was already present in S, there is a *cross* arc from c_1 to the already-present conjunct. In both cases the arc is labelled by the relevant ID. Note that by our chase construction procedure, all ordinary arcs (c,c') have $level(c') = level(c)+1$, and all cross arcs (c,c') have $level(c') \leq level(c)+1$. See Figure 1.

Lemma 2. Suppose c_1 and c_2 are conjuncts of $chase_\Sigma(Q)$ and there is a directed path from c_1 to c_2 using only ordinary arcs of $chase_\Sigma(Q)$ and having length L. Suppose further that c_3 is a conjunct in $chase_\Sigma(Q)$ with $R(c_3)=R(c_1)$ and there is a homomorphism from $\{c_1\}$ to $chase_\Sigma(Q)$ that sends c_1 to c_3. Then there is a homomorphism h of $\{c_1,c_2\}$ to $chase_\Sigma(Q)$ that has $h(c_1)=c_3$ and $level(h(c_2)) \leq level(c_3)+L$.

Proof of Lemma. The proof is by induction on the length of the path. Suppose the lemma holds for all paths of length N or less, where $N \geq 0$, and consider the case where the path from c_1 to c_2 is of length $N+1$. Let c_1' be the immediate predecessor of c_2 in the path ($c_1'=c_1$ if $N=0$). By the lemma for paths of length N, there is a conjunct c_3' in $chase_\Sigma(Q)$ with $level(c_3') \leq level(c_3)+N$ and a homomorphism h' from $\{c_1,c_1'\}$ to $chase_\Sigma(Q)$ with $h'(c_1)=c_3$ and $h'(c_1')=c_3'$. Suppose that the arc from c_1' to c_2 is labelled with the ID "$R[J_1,\ldots,J_k] \subseteq S[K_1,\ldots,K_k]$," where $R=R(c_1)$. By our construction of $chase_\Sigma(Q)$, there must be some arc (c_3',c_4) with this ID as its label, although it may be a cross arc. We thus must have $R(c_2) = R(c_4) = S$. Moreover, we must have $level(c_4) \leq level(c_3')+1 \leq level(c_3)+(N+1)$. Let h_1 be the homomorphism from $\{c_1,c_1'\}$ to $chase_\Sigma(Q)$, and let h_2 be the partial homomorphism that sends each symbol of c_2 to the corresponding symbol of c_4 (h_2 must exist, as it can be shown that repeated symbols in c_2 must be matched by repeated symbols in c_4, and similarly for constants in c_2: all are inherited from c_1'). The only symbols in the domains of both h_1 and h_2 are those in columns $K_1,\ldots,K_k$ of c_2. Consider the symbol $z=c_2[K_1]$. By the definition of $chase_\Sigma(Q)$, $c_2[K_i]=c_1'[J_i]$ and $c_4[K_i]=c_3'[J_i]$ Thus $h_2(z) = h_2(c_2[K_i]) = c_4[K_i] = c_3'[J_i] = h_1(c_1'[J_i]) = h_1(z)$. Hence h_1 and h_2 are consistent, and together they yield the desired homomorphism from $\{c_1,c_2\}$ to $chase_\Sigma(Q)$. []

Lemma 3. Suppose c_1, c_2, and c_3 are as in Lemma 2. Let N_Σ be the maximum number of attributes involved on one side of an ID from Σ. Then there is a homomorphism h from $\{c_1,c_2\}$ to $chase_\Sigma(Q)$ that has $h(c_1)=c_3$ and satisfies

$$level(h(c_2)) \leq level(c_3) + |\Sigma|(N_\Sigma+1)^{N_\Sigma}$$

Proof of Lemma. If the length N of the path from c_1 to c_2 is no more than $M=|\Sigma|(N_\Sigma+1)^{N_\Sigma}$, we are done by Lemma 2. Otherwise, consider the conjuncts along this path. Two of these conjuncts c and c' will be judged *equivalent* if the arc entering each is labelled by the same ID "$R[J_1,\ldots,J_k] \subseteq S[K_1,\ldots,K_k]$" and if, for each i, $1 \leq i \leq k$, $c[K_i]=c'[K_i]$ whenever either symbol occurs in c_1. Note that these are the only columns in c and c' that *can* contain symbols from c_1. Note also that if the path is longer than

M, there must be at least two equivalent conjuncts on it.

The idea of the proof is to excise the portions of the path between equivalent conjuncts, using Lemma 2 to glue the end-pieces together. For instance, suppose c is the first occurrence of a conjunct that has an equivalent conjunct later in the path, and suppose that c' is the last conjunct in the path that is equivalent to c. Let L_1, L_2, and L_3 be the lengths of the paths from c_1 to c, c to c', and c' to c_2, respectively (so that $L = L_1 + L_2 + L_3$), and let "$R[J_1,...,J_k] \subseteq S[K_1,...,K_k]$" be the ID labelling the arcs entering c and c'. By Lemma 2, there is a homomorphism h_1 from $\{c_1, c\}$ to $chase_\Sigma(Q)$ that has $h_1(c_1) = c_3$ and has $level(h_1(c)) \leq level(c_3) + L_1$. By the equivalence of c and c', there must also be a homomorphism h_2 from $\{c_1, c'\}$ to $chase_\Sigma(Q)$ that has $h_2(c_1) = c_3$ and $h_2(c') = h_1(c)$. By Lemma 2 we can now construct a homomorphism h_3 from $\{c', c_2\}$ to $chase_\Sigma(Q)$ with $h_3(c') = h_1(c)$ and $level(h_3(c_2)) \leq level(h_1(c)) + L_3 \leq level(c_3) + L_1 + L_3$. Homomorphisms h_2 and h_3 must be consistent, because they agree on the one conjunct (c') in both domains, and all symbols common to c_1 and c_2 must occur in c'. Thus we get an induced homomorphism h from $\{c_1, c_2\}$ to $chase_\Sigma(Q)$ that leaves the symbols in $c_1[X]$ fixed and has $level(h(c_2)) \leq level(c_3) + (L - L_2)$.

By performing repeated excisions of the above form we can obtain the desired result. []

Lemma 4. If Σ is a set if ID's, Q is a conjunctive query, and C is a set of conjuncts in $chase_\Sigma(Q)$, then there is a homomorphism of C to $chase_\Sigma(Q)$ that preserves the summary row of $chase_\Sigma(Q)$ and such that no conjunct in the image has level exceeding

$$|C| \cdot |\Sigma| (N_\Sigma + 1)^{N_\Sigma}$$

Proof of Lemma.

For each conjunct c in C there is a unique path made up of ordinary arcs in $chase_\Sigma(Q)$ connecting c to a conjunct from Q. Let H be the subgraph of $chase_\Sigma(Q)$ made up of the union of these paths for all c in C, and let C' be the union of C with all those conjuncts in H that belong to Q or have out-degree two or more in H. We then can derive an induced forest F with the conjuncts in C' for vertices and an arc from c to c' if there is a path in H from c to c' all of whose interior vertices have out-degree equal to one, and are not members of C'. See Figure 2.

Note that the maximum length of a path in F from a root to a leaf is simply C. We prove Lemma 4 by induction on this maximum length, using Lemma 3 and the fact that if $c \in C'$, the only symbols shared by descendants of c in F are symbols occurring in c. Details omitted. []

The Theorem is derived from Lemma 4 as follows: Suppose there is a homomorphism h from Q' to $chase_\Sigma(Q)$. Let h' be the homomorphism derived from Lemma 4 by taking $C = h(Q')$. Our desired "bounded-depth" homomorphism from Q' to $chase_\Sigma(Q)$ is then simply the composition of h with h'. []

4. CONTAINMENT FOR FINITE DATABASES

While the last section dealt with containment with respect to arbitrary databases, in practice we are concerned only with finite databases, and hence in $\subseteq_f$ rather than $\subseteq_\infty$. It would be nice if the two notions were equivalent, but they are not. Consider the set Σ consisting of the FD $R: \{2\} \rightarrow 1$ and the ID $R[2] \subseteq R[1]$. The following two conjunctive queries are equivalent for all finite databases obeying Σ but not for all infinite ones:

$$Q_1 = \{(x) : (\exists y) R(x,y)\}$$

$$Q_2 = \{(x) : (\exists y)(\exists y')(R(x,y) \,\&\, R(y',x))\}$$

Thus $\subseteq_\infty$ implies $\subseteq_f$, but $\subseteq_f$ does not imply $\subseteq_\infty$. Our final result shows that for the class of key-based dependencies, the two notions *are* equivalent.

Theorem 3. If Σ is a set of key-based dependencies and Q and Q' are conjunctive queries, then $\Sigma \models Q \subseteq_f Q'$ implies $\Sigma \models Q \subseteq_\infty Q'$.

Proof. Suppose $\Sigma \models Q \subseteq_f Q'$. By the definition of $\subseteq_f$, this means that for each finite database B satisfying the input schema for Q and Q', and each tuple t in B, if there exists a homomorphism from Q to B that sends the summary row of Q to t, then there exists a homomorphism of Q' to B that sends the summary row of Q' to t. By Theorem 1, we must show that there is a homomorphism of Q' to $chase_\Sigma(Q)$ that sends the summary row of Q' to that of $chase_\Sigma(Q)$.

We shall rely on two properties of key-based dependencies. Property (P1) was mentioned in the previous section: If the set of FD's in Σ is $\Sigma[F]$ and the set of ID's is $\Sigma[I]$, then $chase_\Sigma(Q) = chase_{\Sigma[I]}(chase_{\Sigma[F]}(Q))$. Because of (P1) we may assume without loss of generality that $\Sigma[F] = \emptyset$. Property (P2) follows from part (b) of the definition of *key-based* and provides a limit on the propagation of symbols in $chase_\Sigma(Q)$: For all $i \geq 0$, if z is a symbol occurring in a conjunct at level i of $chase_\Sigma(Q)$, then z occurs in no conjunct with level exceeding $i+1$.

The idea of the proof is to construct a finite database B^* that contains all of Q and yet obeys the dependencies in Σ. The identity homomorphism sends Q to B^*, and so there must be a homomorphism h from Q' to B^*. If B^* is chosen so as to look like $chase_\Sigma(Q)$, at least locally, then perhaps h can be modified slightly so that it becomes our desired homomorphism from Q' to $chase_\Sigma(Q)$.

Consider the graph $G_{Q'}$ that has a vertex for the summary row and each conjunct in Q', and an edge between two vertices if the corresponding conjuncts and/or summary row share a symbol. If $G_{Q'}$ is connected, the construction of B^* is easy:

Let d be the diameter of $G_{Q'}$, and construct the first $d+1$ levels of $chase_\Sigma(Q)$. Then choose a new special symbol z_A for each attribute A and modify the chase rule for ID's so that whenever a new conjunct c is created because of an ID "$R[X] \subseteq S[Y]$," the entry in each column of c labelled by an attribute A *not* in Y is the special symbol z_A. This will insure that the chase procedure will terminate: After one more level the only symbols left will be the special symbols, and thereafter the chase can only proceed until a conjunct c_R, completely filled with special symbols, has been constructed for each relation R that occurs on the right-hand-side of some ID.

We claim that if there is a homomorphism h from Q' to B^* that sends summary row to summary row, then the conjuncts of $h(Q')$ must all lie within the first $d+1$ levels of B^*, and since these levels are identical to the first $d+1$ levels of $chase_\Sigma(Q)$, h gives us our desired homomorphism of Q' to $chase_\Sigma(Q)$. First of all, since h sends summary row to summary row, each DV of Q' must go to a symbol of Q. Since $G_{Q'}$ is connected, some conjunct of Q' must contain a DV. Thus, by (P2), some conjunct c of G' must satisfy $level(h(c)) \leq 1$. Now observe that if c and c' are adjacent in $G_{Q'}$, we must, again by (P2), have $|level(h(c)) - level(h(c'))| \leq 1$. Thus, by definition of *diameter*, we must have, for all pairs c, c' of conjuncts in $G_{Q'}$, $|level(h(c)) - level(h(c'))| \leq d$, and the claim follows.

The problem of constructing the desired database B^* becomes a bit trickier when $G_{Q'}$ is *not* connected. This corresponds to the case when the query Q' has a boolean part, i.e., when Q' is of the form "If $Q_1(B)$ is true, return $Q_2(B)$, else return nothing." If $G_{Q'}$ is not connected, it may have a component C that contains no occurences of DV's, and whose image under h is hence not constrained to lie close to level 0. The policy used above, that is, constructing B^* by generating the first few levels of $chase_\Sigma(Q)$ and then "closing off" the structure into a finite database, might not work because C might map to the special structure used for the closing off. This could happen no matter how deep we go in mimicking $chase_\Sigma(Q)$ before closing it off.

The above dilemma is what keeps us from proving Theorem 3 for more general classes of dependencies than just the key-based ones. However, for the case at hand we are still in luck, because of (P2). The idea is to do the closing off, not with *new*

special symbols, but with symbols that have already been used at a level at least $d+1$ earlier, where d is the maximum diameter of any connected component of G_Q. Thus the image of a connected component C might look as schematized in Figure 3a if it were to straddle the closing-off portion of B^*, but the two parts of this image can be put back together and mapped to $chase_\Sigma(Q)$, using techniques similar to those used for proving Theorem 2. Since distinct connected components share no common symbols, we can change the homomorphism for one component without affecting any of the others, so each component can be handled separately in this way and an overall homomorphism constructed that has the desired properties.

We omit the remaining details. []

5. CONCLUDING REMARKS

Theorems 2 and 3 can be slightly generalized. By our argument used at the beginning of the proof of Theorem 2 to reduce the case of key-based dependencies to that of a Σ consisting only of ID's, we can see that Theorem 2 will in fact hold for any set of FD's and ID's satisfying property (P2) of the previous section, a considerably more general class than just those Σ that are key-based (It includes, for instance, any Σ in which all ID's involve only one attribute per side).

Similarly, Theorem 3 will hold for any set of FD's and ID's satisfying both (P1) and (P2). (Theorem 3 can also be generalized to handle any Σ in which all ID's involve only one attribute per side, since these can be guaranteed to satisfy a weakened version of (P2): if a symbol occurs in a conjunct c of $chase_\Sigma(Q)$, it cannot occur in any conjuncts with level exceeding $level(c)+|\Sigma|$). We suspect that Theorem 3 may well also hold for any Σ that contains only ID's. This is because if a database B must not only obey a set of ID's but also be finite, the finiteness condition only seems to have the effect of forcing cycles (perhaps arbitrarily long ones) to occur in the graph G_B, which has a vertex for each tuple in B and an edge between to vertices if the corresponding tuples share a symbol. Conjunctive queries seem only to be able to talk about cycles of bounded length, and so if there is some infinite database that distinguishes between two queries, there is probably some very large finite one that also makes the distinction.

The above intuitive argument is, however, a long way from a formal proof, and it appears that the techniques in our proof of Theorem 3 do not readily extend to this case. Thus the question of whether our "compactness" theorem extends to the case where Σ contains only ID's might well be a good topic for further research. We also wonder whether Theorem 2 might not extend to *all* sets Σ made up of FD's and ID's (or is there, say, a Σ containing only FD's and ID's for which the containment problem for conjunctive queries is PSPACE-hard, or worse?).

Another possible direction for further work is to try our approach on problems involving embedded multivalued dependencies (EMVD's) [11]. Chases involving EMVD's also introduce new symbols and so do not terminate. Which sets of EMVD's give rise to containment problems that are "only" as hard as NP? And what about arbitrary first-order queries? Here the containment problem is PSPACE-easy when there are no dependencies [5]. What happens when ID's are introduced?

REFERENCES

1. A. V. Aho, Y. Sagiv, and J. D. Ullman, "Equivalences among relational expressions," *SIAM J. Comput.* **8** (1979), 218-246.
2. A. V. Aho, Y. Sagiv, and J. D. Ullman, "Efficient optimization of a class of relational expressions," *ACM Trans. Database Syst.* **4** (1979), 435-454.
3. M. A. Casanova, R. Fagin, and C. H. Papadimitriou, "Inclusion dependencies and their interaction with functional dependencies," this proceedings.
4. A. K. Chandra, H. R. Lewis, and J. A. Makowsky, "Embedded implicational dependencies and their inference problem," *Proc. 13th Ann. ACM Symp. on Theory of Computing*, Assoc. Comp. Mach., New York, 1981, 342-354.
5. A. K. Chandra and P. M. Merlin, "Optimal implementation of conjunctive queries in relational data bases," *Proc. 9th Ann. ACM Symp. on Theory of Computing*, Assoc. Comput. Mach., New York, 1977, 77-90.
6. E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM* **13** (1970), 377-387.
7. E. F. Codd, "Extending the database relational model to obtain more meaning," *ACM Trans. Database Syst.* **4** (1979), 397-434.
8. R. Fagin, "A normal form for relational databases that is based on domains and keys," *ACM Trans. Database Syst.* **6** (1981), 387-415.
9. D. S. Johnson and A. Klug, "Optimizing conjunctive queries when attribute domains are not disjoint," *Proc. 22nd Ann. Symp. on Foundations of Computer Science*, IEEE Computer Society, Long Beach, CA, 1981.
10. A. Klug, "Entity relationship views over uninterpreted enterprise schemes," *in* P. P. Chen (ed.) *Entity-Relationship Approach to Systems Analysis and Design*, North-Holland, Amsterdam, 1980, 35-59.
11. D. Maier, A. O. Mendelzon, and Y. Sagiv, "Testing implications of data dependencies," *ACM Trans. Database Syst.* **4** (1979), 455-469.
12. J. M. Smith and D. C. P. Smith, "Database abstractions: aggregation," *Commum. ACM* **20** (1977), 405-413.
13. C. Zaniolo, "Design of relational views over network schemas," *Proc. ACM SIGMOD Conf.*, Assoc. Comp. Mach., 1979, 179-180.

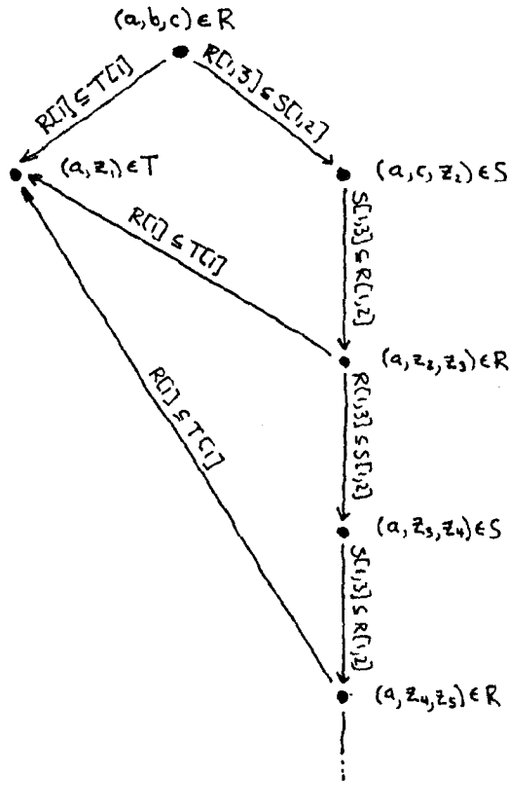

FIGURE 1. $\text{CHASE}_\Sigma(Q)$ AS A DIRECTED GRAPH

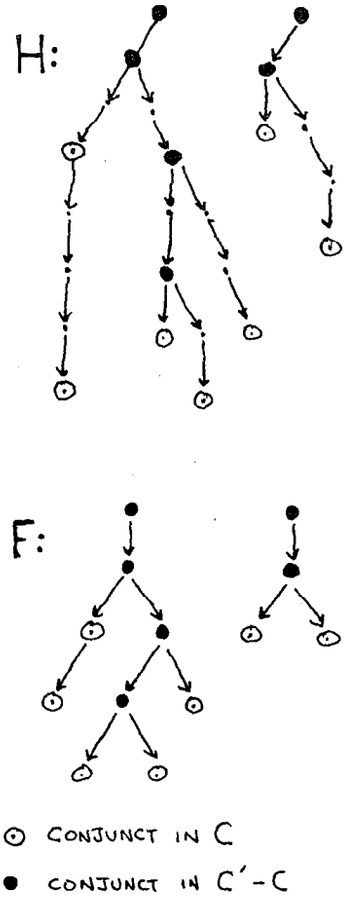

FIGURE 2. SUBGRAPH H OF $\text{CHASE}_\Sigma(Q)$ AND INDUCED FOREST F (FROM PROOF OF LEMMA 4)

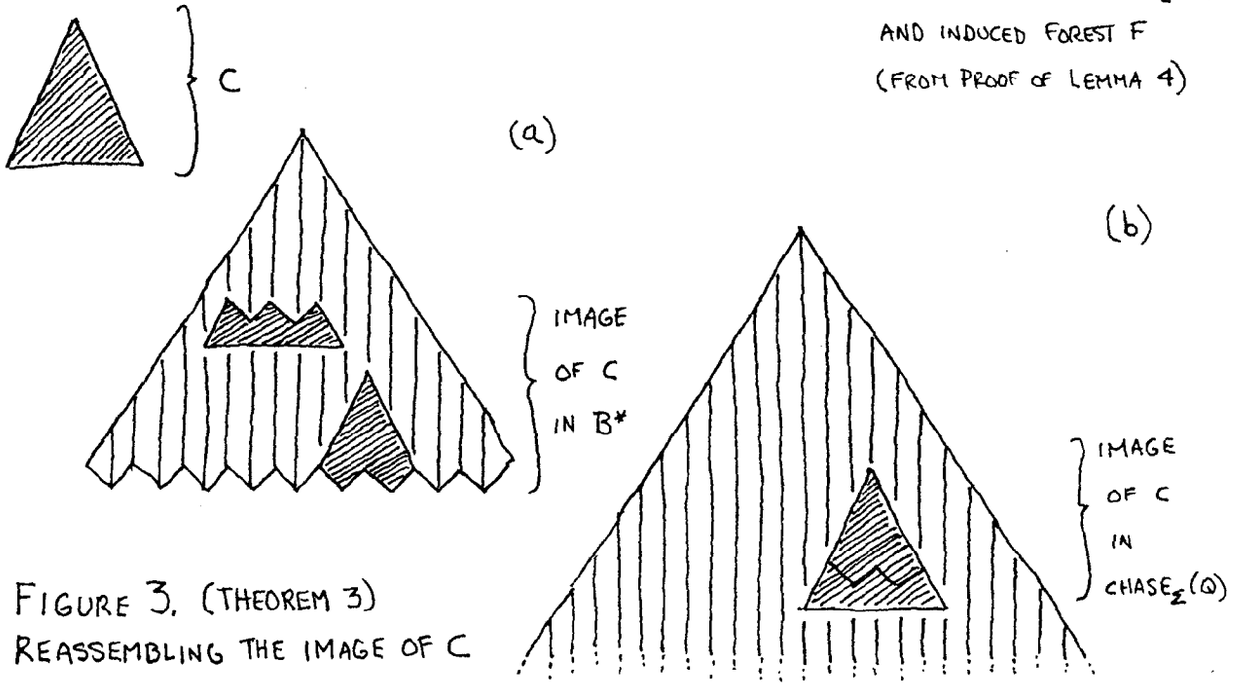

FIGURE 3. (THEOREM 3) REASSEMBLING THE IMAGE OF C