On Containment of Conjunctive Queries with Arithmetic Comparisons

Foto Afrati[1], Chen Li[2], and Prasenjit Mitra[3]

[1] National Technical University of Athens, 157 73 Athens, Greece
afrati@cs.ece.ntua.gr
[2] Information and Computer Science, University of California, Irvine, CA 92697, U.S.A
chenli@ics.uci.edu**
[3] School of Information Sciences and Techonology
The Pennsylvania State University, University Park, PA 16802, U.S.A
pmitra@ist.psu.edu

Abstract. We study the following problem: how to test if Q_2 is contained in Q_1, where Q_1 and Q_2 are conjunctive queries with arithmetic comparisons? This problem is fundamental in a large variety of database applications. Existing algorithms first normalize the queries, then test a logical implication using multiple containment mappings from Q_1 to Q_2. We are interested in cases where the containment can be tested more efficiently. This work aims to (a) reduce the problem complexity from Π_2^P-completeness to NP-completeness in these cases; (b) utilize the advantages of the homomorphism property (i.e., the containment test is based on a single containment mapping) in applications such as those of answering queries using views; and (c) observing that many real queries have the homomorphism property. The following are our results. (1) We show several cases where the normalization step is not needed, thus reducing the size of the queries and the number of containment mappings. (2) We develop an algorithm for checking various syntactic conditions on queries, under which the homomorphism property holds. (3) We further reduce the conditions of these classes using practical domain knowledge that is easily obtainable. (4) We conducted experiments on real queries, and show that most of the queries pass this test.

1 Introduction

The problem of testing query containment is as follows: how to test whether a query Q_2 is *contained* in a query Q_1, i.e., for any database D, is the set of answers to Q_2 a subset of the answers to Q_1? This problem arises in a large variety of database applications, such as query evaluation and optimization [1], data warehousing [2], and data integration using views [3]. For instance, an important problem in data integration is to decide how to answer a query using source views. Many existing algorithms are based on query containment [4].

** Supported by NSF CAREER award No. IIS-0238586.

A class of queries of great significance is conjunctive queries (select-project-join queries and Cartesian products). These queries are widely used in many database applications. Often, users need to pose queries with arithmetic comparisons (e.g., year > 2000, price ≤ 5000). Thus testing for containment of conjunctive queries with arithmetic comparisons becomes very important. Several algorithms have been proposed for testing containment in this case (e.g., [5, 6]). These algorithms first normalize the queries by replacing constants and shared variables, each with new unique variables and add arithmetic comparisons to equate those new variables to the original constants or shared variables. Then, they test the containment by checking a logical implication using *multiple* containment mappings. (See Section 2 for detail.)

We study how to test containment of conjunctive queries with arithmetic comparisons. In particular, we focus on the following two problems: (1) In what cases is the normalization step not needed? (2) In what cases does the *homomorphism property* hold, i.e., the containment test is based on a single containment mapping [6]?

We study these problems for three reasons. The first is the efficiency of this test procedure. Whereas the problem of containment of pure conjunctive queries is known to be NP-complete [7], the problem of containment of conjunctive queries with arithmetic comparisons is Π_2^P-complete [6, 8]. In the former case, the containment test is in NP, because it is based on the existence of a *single* containment mapping, i.e., the homomorphism property holds. In the latter case, the test needs multiple containment mappings, which significantly increases the problem complexity. We find large classes of queries where the homomorphism property holds; thus we can reduce the problem complexity to NP. Although the saving on the normalization step does not put the problem in a different complexity class (i.e., it is still in NP), it can still reduce the sizes of the queries and the number of containment mappings in the containment test.

The second reason is that the homomorphism property can simplify many problems such as that of answering queries using views [9], in which we want to construct a plan using views to compute the answer to a query. It is shown in [10] that if both the query and the views are conjunctive queries with arithmetic comparisons, and the homomorphism property does not hold, then a plan can be recursive. Hence, if we know the homomorphism property holds by analyzing the query and the views, we can develop efficient algorithms for constructing a plan using the views.

The third motivation is that, in studying realistic queries (e.g., in TPC benchmarks), we found it hard to construct examples that need multiple mappings in the containment test. We observed that most real query pairs only need a single containment mapping to test the containment. To easily detect such cases, we want to derive syntactic conditions on queries, under which the homomorphism property holds. These syntactic conditions should be easily checked in polynomial time. In this paper, we develop such conditions.

The following are our contributions of this work. (Table 1 is a summary of results.)

Contained Query	Containing Query	Complexity	References
CQ	CQ	NP	[7]
CQ with closed LSI	CQ with closed LSI	NP	[5, 6]
CQ with open LSI	CQ with open LSI	NP	[5, 6]
CQ with AC	CQ with closed LSI	NP	Section 4
CQ with AC	CQ with LSI	NP	Section 4
Constraints	(i)-lsi, (ii)-lsi, (iii)-lsi		Theorem 4
CQ with SI	CQ with LSI, RSI	NP	Section 4
Constraints	(i)-lsi,rsi, (ii)-lsi,rsi, (iii)-lsi,rsi, (iv)		Theorem 5
CQ with SI	CQ with LSI, RSI, PI	NP	Section 4
Constraints	as above and (v),(vi),(vii)		Theorem 6
CQ with AC	CQ with AC	Π_2^P	[8]

Table 1. Results on containment test. The classes in NP have the homomorphism property. (See Table 2 for symbol definitions.)

1. We show cases where the normalization step is not needed (Section 3).
2. When the containing query Q_1 has only arithmetic comparisons between a variable and a constant (called "semi-interval," or "SI" for short), we present cases where the homomorphism property holds (Section 4). If the homomorphism property does not hold, then some "heavy" constraints must be satisfied. Such a constraint could be: An ordinary subgoal of Q_1, an ordinary subgoal of Q_2, an open-left-semi-interval subgoal of Q_2, and a closed-left-semi-interval subgoal of Q_2 all use the same constant. (See Table 1 for the definitions of these terms.) Notice that these conditions are just syntactic constraints, and can be checked in time polynomial in the size of the queries.
3. We further relax the conditions of the homomorphism property using practical domain knowledge that is easily obtainable (Section 5).
4. We conducted experiments on real queries, and show many of them satisfy the conditions under which the homomorphism property holds (Section 6).

Due to space limitation, we leave theorem proofs and more experimental results in the complete version [11].

1.1 Related Work

For conjunctive queries, restricted classes of queries are known for which the containment problem is polynomial. For instance, if every database predicate occurs in the contained query at most twice, then the problem can be solved in linear time [12], whereas it remains NP-complete if every database predicate occurs at least three times in the body of the contained query. If the containing query is acyclic, i.e., the predicate hypergraph has a certain property, then the containment problem is polynomial [13].

Klug [6] has shown that containment for conjunctive queries with comparison predicates is in Π_2^P, and it is proven to be Π_2^P-hard in [8]. The reduction only used $\neq$. This result is extended in [14] to use only $\neq$ and at most three occurrences of the same predicate name in the contained query. The same reduction

shows that it remains Π_2^P-complete even in the case where the containing query is acyclic, thus the results in [13] do not extend to conjunctive queries with $\neq$. The complexity is reduced to co-NP in [14] if every database predicate occurs at most twice in the body of the contained query and only $\neq$ is allowed.

The most relevant to our setting is the work in [5, 6]. It is shown that if only left or right semi-interval comparisons are used, the containment problem is in NP. It is stated as an open problem to search for other classes of conjunctive queries with arithmetic comparisons for which containment is in NP. Furthermore, query containment has been studied also for recursive queries. For instance, containment of a conjunctive query in a datalog query is shown to be EXPTIME-complete [15, 16]. Containment among recursive and nonrecursive datalog queries is also studied in [17, 18].

In [10] we studied the problem of how to answer a query using views if both the query and views are conjunctive queries with arithmetic comparisons. Besides showing the necessity of using recursive plans if the homomorphism property does not hold, we also developed an algorithm in the case where the property holds. Thus the results in [10] are an application of the contributions of this paper. Clearly testing query containment efficiently is a critical problem in many data applications as well.

2 Preliminaries

In this section, we review the definitions of query containment, containment mappings, and related results in the literature. We also define the homomorphism property.

Definition 1. *(Query containment) A query Q_2 is contained in a query Q_1, denoted $Q_2 \sqsubseteq Q_1$, if for any database D, the set of answers to Q_2 is a subset of the answers to Q_1. The two queries are equivalent, denoted $Q_1 \equiv Q_2$, if $Q_1 \sqsubseteq Q_2$ and $Q_2 \sqsubseteq Q_1$.*

We consider conjunctive queries that are in the following form: $h(\bar{X})$:- $g_1(\bar{X}_1), \ldots, g_k(\bar{X}_k)$. In each subgoal $g_i(\bar{X}_i)$, predicate g_i is a *base relation*, and every predicate argument $\bar{X}_i$ is either a variable or a constant. Chandra and Merlin [7] showed that for two conjunctive queries Q_1 and Q_2, $Q_2 \sqsubseteq Q_1$ if and only if there is a *containment mapping* from Q_1 to Q_2, such that the mapping maps a constant to the same constant, and maps a variable to either a variable or a constant. Under this mapping, the head of Q_1 becomes the head of Q_2, and each subgoal of Q_1 becomes *some* subgoal in Q_2.

Let Q be a conjunctive query with arithmetic comparisons (CQAC). We consider the following arithmetic comparisons: $<$, $\leq$, $>$, $\geq$, and $\neq$. We assume that database instances are over densely totally ordered domains. In addition, without loss of generality, throughout the paper we make the following assumptions about the comparisons. (1) The comparisons are not contradictory, i.e., there exists an instantiation of the variables such that all the comparisons are true. (2) All the comparisons are safe, i.e., each variable in the comparisons appears

in some ordinary subgoal. (3) The comparisons do not imply equalities. If they imply an equality $X = Y$, we rewrite the query by substituting X for Y.

We denote $core(Q)$ as the set of ordinary (uninterpreted) subgoals of Q that do not have comparisons, and denote $AC(Q)$ as the set of subgoals that are arithmetic comparisons in Q. We use the term *closure* of a set of arithmetic comparisons S, to represent the set of all possible arithmetic comparisons that can be logically derived from S. For example, for the set of arithmetic comparisons $S = \{X \leq Y, Y = c\}$, we have $Closure(S) = \{X \leq Y, Y = c, X \leq c\}$. In addition, for convenience, we will denote Q_0 as the corresponding conjunctive query whose head is the head of Q, and whose body is $core(Q)$. See Table 2 for a complete list of definitions and notations on special cases of arithmetic comparisons such as semi-interval, point inequalities, and others.

Symbol	Meaning
CQ	Conjunctive Query
AC	Arithmetic Comparison ($X\ \theta\ Y$)
CQAC	Conjunctive Query with ACs
$core(Q)$	Set of ordinary subgoals of query Q
$AC(Q)$	Set of arithmetic-comparison subgoals of query Q
SI	Semi-interval: $X\ \theta\ c$, $\theta \in \{<, \leq, >, \geq\}$
LSI	Left-semi-interval: $X\ \theta\ c$, $\theta \in \{<, \leq\}$
closed-LSI	$X \leq c$
open-LSI	$X < c$
PI	Point Inequalities ($X \neq c$)
SI-PI	Some subgoals are SI, and some are PI

Table 2. Symbols used in the paper. X denotes a variable and c is a constant. The RSI cases are symmetrical to those of LSI.

2.1 Testing Containment

Let Q_1 and Q_2 be two conjunctive queries with arithmetic comparisons (CQACs). Throughout the paper, we study how to test whether $Q_2 \sqsubseteq Q_1$. To do the testing, according to the results in [5, 6], we first *normalize* both queries Q_1 and Q_2 to Q_1' and Q_2' respectively as follows.

- For all occurrences of a shared variable X in the normal subgoals except the first occurrence, replace the occurrence of X by a new distinct variable X_i, and add $X = X_i$ to the AC's of the query; and
- For each constant c in the query, replace the constant by a new distinct variable Z, and add $Z = c$ to the AC's of the query.

The testing is illustrated in Figure 1. For simplicity, we denote $\beta_1 = AC(Q_1)$, $\beta_2 = AC(Q_2)$, $\beta_1' = AC(Q_1')$, and $\beta_2' = AC(Q_2')$. Let $\mu_1, \ldots, \mu_k$ be all the containment mappings from $Q'_{1,0}$ to $Q'_{2,0}$. Let $\gamma_1, \ldots, \gamma_l$ be all the containment

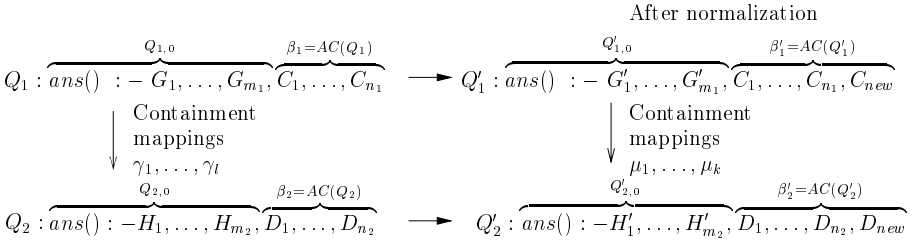

Fig. 1. Containment testing ([5, 6]).

mappings from $Q_{1,0}$ to $Q_{2,0}$. There are a few important observations: (1) The number of ordinary subgoals in Q_1 (resp. Q_2) does not change after the normalization. Each subgoal G_i (resp. H_i) has changed to a new subgoal G'_i (resp. H'_i). (2) While the comparisons $C_1, \ldots, C_{n_1}$ (resp. $D_1, \ldots, D_{n_2}$) are kept after the normalization, we may have introduced new comparions C_{new} (resp. D_{new}) after the normalization. Note C_{new} and D_{new} contain only equalities. (3) There can be more containment mappings for the normalized queries than the original queries, i.e., $k \geq l$. The reason is that a containment mapping cannot map a constant to a variable, nor map different instances of the same variable to different variables. However, after normalizing the queries, their ordinary subgoals only have distinct variables, making any variable in Q'_1 mappable to any variable in Q'_2 (for the same position of the same predicate).

Theorem 1. *$Q_2 \sqsubseteq Q_1$ if and only if the following logical implication ϕ is true:*

$$\phi : \beta'_2 \Rightarrow \mu_1(\beta'_1) \vee \ldots \vee \mu_k(\beta'_1)$$

That is, the comparisons in the normalized query Q'_2 logically implies (denoted "$\Rightarrow$") the disjunction of the images of the comparisons of the normalized query Q'_1 under these mappings [5, 6].

Example 1. These two queries show that the normalization step in Theorem 1 is critical [19].

$Q_1 : h(W)$:- $q(W), p(X, Y, Z, Z', U, U), X < Y, Z > Z'$.
$Q_2 : h(W)$:- $q(W), p(X, Y, 2, 1, U, U), p(1, 2, X, Y, U, U), p(1, 2, 2, 1, X, Y)$.

There are two containment mappings from $Q_{1,0}$ to $Q_{2,0}$.

$$\mu_1 : W \to W, X \to X, Y \to Y, Z \to 2, Z' \to 1, U \to U.$$
$$\mu_2 : W \to W, X \to 1, Y \to 2, Z \to X, Z' \to Y, U \to U.$$

Notice we do not have a containment mapping from the p subgoal in Q_1 to the last p subgoal in Q_2, since we cannot map the two instances of variable U to both X and Y.

We can show $Q_2 \sqsubseteq Q_1$, but the following implication

$$TRUE \Rightarrow \mu_1(X < Y, Z > Z') \vee \mu_2(X < Y, Z > Z')$$

is not true, since it is possible $X = Y$. However, when $X = Y$, we would have a new "containment mapping" from Q_1 to Q_2:

$$\mu_3 : W \to W, X \to 1, Y \to 2, Z \to 2, Z' \to 1, U \to X = Y$$

After normalizing the two queries, we will have three (instead of two) containment mappings from the normalized query of Q_1 to that of Q_2.

Example 2. These two queries show that the $\vee$ operation in the implication in Theorem 1 is critical.

$$Q_1 : ans() \text{ :- } p(X, 4), X < 4.$$
$$Q_2 : ans() \text{ :- } p(A, 4), p(3, A), A \leq 4.$$

Their normalized queries are:

$$Q'_1 : ans() \text{ :- } p(X, Y), X < 4, Y = 4.$$
$$Q'_2 : ans() \text{ :- } p(A, B), p(C, D), A \leq 4, B = 4, C = 3, A = D.$$

There are two containment mappings from $Q'_{1,0}$ to $Q'_{2,0}$: $\mu_1 : X \to A, Y \to B$, and $\mu_2 : X \to C, Y \to D$. We can show that:

$$A \leq 4, B = 4, C = 3, A = D \Rightarrow \mu_1(X < 4, Y = 4) \vee \mu_2(X < 4, Y = 4)$$

Thus, $Q_2 \sqsubseteq Q_1$. Note both mappings are needed to prove the implication.

There are several challenges in using Theorem 1 to test whether $Q_2 \sqsubseteq Q_1$. (1) The queries look less intuitive after the normalization. The computational cost of testing the implication ϕ increases since we need to add more comparisons. (2) The implication needs the disjunction of the images of multiple containment mappings. In many cases it is desirable to have a single containment mapping to satisfy the implication. (3) There can be more containment mappings between the normalized queries than those between the original queries. In the rest of the paper we study how to deal with these challenges. In Section 3 we study in what cases we do not need to normalize the queries. That is, even if Q_1 and Q_2 are not normalized, we still have $Q_2 \sqsubseteq Q_1$ if and only if $\beta_2 \Rightarrow \gamma_1(\beta_1) \vee \ldots \vee \gamma_l(\beta_1)$.

2.2 Homomorphism Property

Definition 2. *(Homomorphism property) Let $\mathcal{Q}_1$, $\mathcal{Q}_2$ be two classes of queries. We say that containment testing* on the pair $(\mathcal{Q}_1, \mathcal{Q}_2)$ has the homomorphism property *if for any pair of queries (Q_1, Q_2) with $Q_1 \in \mathcal{Q}_1$ and $Q_2 \in \mathcal{Q}_2$, the following holds: $Q_2 \sqsubseteq Q_1$ iff there is a homomorphism μ from $core(Q_1)$ to $core(Q_2)$ such that $AC(Q_2) \Rightarrow \mu(AC(Q_1))$. If $\mathcal{Q}_1 = \mathcal{Q}_2 = \mathcal{Q}$, then we say containment testing has the homomorphism property for class $\mathcal{Q}$.*

Although the property is defined for two classes of queries, in the rest of the paper we refer to the homomorphism property holding for two queries when the two classes contain only one query each. The containment test of Theorem

1 for general CQACs considers normalized queries. However, in Theorem 3, we show that in the cases where a single mapping suffices to show containment between normalized queries, it also suffices to show containment between these queries when they are not in normalized form and vice versa. Hence, whenever the homomorphism property holds, we need not distinguish between normalized queries and non-normalized ones.

In cases where the homomorphism property holds, we have the following non-deterministically polynomial algorithm that checks if $Q_2 \sqsubseteq Q_1$. Guess a mapping μ from $core(Q_1)$ to $core(Q_2)$ and check whether μ is a containment mapping with respect to the AC subgoals too (the latter meaning that an AC subgoal g maps on an AC subgoal g' so that $g' \Rightarrow g$ holds). Note that the number of mappings is exponential on the size of the queries.

Klug [6] has shown that for the class of conjunctive queries with only open-LSI (open-RSI respectively) comparisons, the homomorphism property holds. In this paper, we find more cases where the homomorphism property holds. Actually, we consider pairs of classes of queries such as (LSI-CQ, CQAC) and we look for constraints which, if satisfied, the homomorphism property holds.

Definition 3. *(Homomorphism property under constraints) Let $\mathcal{Q}_1$, $\mathcal{Q}_2$ be two classes of queries and $\mathcal{C}$ be a set of constraints. We say that containment testing* on the pair $(\mathcal{Q}_1, \mathcal{Q}_2)$ w.r.t. the constraints in $\mathcal{C}$ has the homomorphism property *if for any pair of queries (Q_1, Q_2) with $Q_1 \in \mathcal{Q}_1$ and $Q_2 \in \mathcal{Q}_2$ and for which the constraints in $\mathcal{C}$ are satisfied, the following holds: $Q_2 \sqsubseteq Q_1$ iff there is a homomorphism μ from $core(Q_1)$ to $core(Q_2)$ such that $AC(Q_2) \Rightarrow \mu(AC(Q_1))$.*

The constraints we use are given as *syntactic conditions* that relate subgoals, in both queries. The satisfaction of the constraints can be checked in polynomial time in the size of the queries. When the homomorphism property holds, then the query containment problem is in NP.

3 Containment of Non-normalized Queries

To test the containment of two queries Q_1 and Q_2, using the result in Theorem 1, we need to normalize them first. Introducing more comparisons to the queries in the normalization can make the implication test computationally more expensive. Thus, we want to have a containment result that does not require the queries to be normalized. In this section, we present two cases, in which even if Q_1 and Q_2 are not normalized, we still have $Q_2 \sqsubseteq Q_1$ if and only if $\beta_2 \Rightarrow \gamma_1(\beta_1) \vee \ldots \vee \gamma_l(\beta_1)$.

Case 1: The following theorem says that Theorem 1 is still true even for non-normalized queries Q_1, if two conditions are satisfied by the queries: (1) β_1 contains only $\leq$ and $\geq$, and (2) β_1 (correspondingly β_2) do not imply equalities. In this case we can restrict the space of mappings because of the monotonicity property: For a query Q whose AC's only include $\leq, \geq$, if a tuple t of a database D is an answer to Q, then on any database D' obtained from D, by identifying some elements, the corresponding tuple t' is in the answer to $Q(D')$. Due to space limitations, we give the proofs of all theorems in [11].

Theorem 2. *Consider two CQAC queries Q_1 and Q_2 shown in Figure 1 that may not be normalized. Suppose β_1 contains only $\leq$ and $\geq$, and β_1 (correspondingly β_2) do not imply "=" restrictions. Then $Q_2 \sqsubseteq Q_1$ if and only if:*

$$\beta_2 \Rightarrow \gamma_1(\beta_1) \vee \ldots \vee \gamma_l(\beta_1)$$

where $\gamma_1, \ldots, \gamma_l$ are all the containment mappings from $Q_{1,0}$ to $Q_{2,0}$.

Case 2: The following theorem shows that we do not need to normalize the queries if they have the homomorphism property.

Lemma 1. *Assume the comparisons in Q_1 and Q_2 do not imply equalities. If there is a containment mapping μ from $Q'_{1,0}$ to $Q'_{2,0}$, such that $\beta'_2 \Rightarrow \mu(\beta'_1)$, then there must be a containment mapping γ from $Q_{1,0}$ to $Q_{2,0}$, such that $\beta_2 \Rightarrow \gamma(\beta_1)$.*

Using the lemma above, we can prove:

Theorem 3. *Suppose the comparisons in Q_1 and Q_2 do not imply equalities. The homomorphism property holds between Q_1 and Q_2 iff it holds between Q'_1 and Q'_2.*

4 Conditions for Homomorphism Property

Now we look for constraints in the form of syntactic conditions on queries Q_1 and Q_2, under which the homomorphism property holds. The conditions are sufficiently tight in that, if at least one of them is violated, then there exist queries Q_1 and Q_2 for which the homomorphism property does not hold. The conditions are syntactic and can be checked in polynomial time. We consider the case where the containing query (denoted by Q_1 all through the section) is a conjunctive query with only arithmetic comparisons between a variable and a constant; i.e., all its comparisons are *semi-interval* (SI), which are in the forms of $X > c$, $X < c$, $X \geq c$, $X \leq c$, or $X \neq c$. We call $X \neq c$ a *point inequality* (PI).

This section is structured as follows. Section 4.1 discusses technicalities on the containment implication, and in particular in what cases we do not need a disjunction. In Section 4.2 we consider the case where the containing query has only left-semi-interval (LSI) subgoals. We give a main result in Theorem 4. In Section 4.3, we extend Theorem 4 by considering the general case, where the containing query may use any semi-interval subgoals and point inequality subgoals. In Section 4.4, we discuss the case for more general inequalities than SI. Section 4.5 gives an algorithm for checking whether these conditions are met. In [11], we include many examples to show that the conditions in the main theorems are tight.

4.1 Containment Implication

In this subsection, we will focus on the implication

$$\phi : \beta'_2 \Rightarrow \mu_1(\beta'_1) \vee \ldots \vee \mu_k(\beta'_1)$$

in Theorem 1. We shall give some terminology and some basic technical observations. The left-hand side (lhs) is a conjunction of arithmetic comparisons (in Example 2, the lhs is: $A \leq 4 \wedge B = 4 \wedge C = 3 \wedge A = D$). The right-hand side (rhs) is a disjunction and each disjunct is a conjunction of arithmetic comparisons. For instance, in Example 2, the rhs is: $(A < 4 \wedge B = 4) \vee (C < 4 \wedge D = 4)$, which has two disjuncts, and each is the conjunction of two comparisons. Given an integer i, we shall call **containment implication** any implication of this form: i) the lhs is a conjunction of arithmetic comparisons, and ii) the rhs is a disjunction and each disjunct is a conjunction of i arithmetic comparisons.

Observe that the rhs can be equivalently written as a conjunction of disjunctions (using the distributive law). Hence this implication is equivalent to a conjunction of implications, each implication keeping the same lhs as the original one, and the rhs is one of the conjuncts in the implication that results after applying the distributive law. We call each of these implications a **partial containment implication**.[4] In Example 2, we write equivalently the rhs as: $(A < 4 \vee C < 4) \wedge (A < 4 \vee D = 4) \wedge (B = 4 \vee C < 4) \wedge (B = 4 \vee D = 4)$. Thus, the containment implication in Example 2 can be equivalently written as

$$
\begin{array}{l}
(A \leq 4, B = 4, C = 3, A = D \Rightarrow A < 4 \vee C < 4) \wedge \\
(A \leq 4, B = 4, C = 3, A = D \Rightarrow A < 4 \vee D = 4) \wedge \\
(A \leq 4, B = 4, C = 3, A = D \Rightarrow B = 4 \vee C < 4) \wedge \\
(A \leq 4, B = 4, C = 3, A = D \Rightarrow B = 4 \vee D = 4).
\end{array}
$$

Here we get four partial containment implications.

A partial containment implication $\alpha \Rightarrow (\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_k)$ is called a *direct implication* if there exists an i, such that if this implication is true, then $\alpha \Rightarrow \alpha_i$ is also true. Otherwise, it is called a *coupling implication*. For instance,

$$(A \leq 4, B = 4, C = 3, A = D \Rightarrow B = 4 \vee D = 4)$$

is a direct implication, since it is logically equivalent to $(A \leq 4, B = 4, C = 3, A = D \Rightarrow B = 4)$. On the contrary, $(A \leq 4, B = 4, C = 3, A = D \Rightarrow A < 4 \vee D = 4)$ is a coupling implication. The following lemma is used as a basis for many of our results.

Lemma 2. *Consider a containment implication $\alpha \Rightarrow (\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_k)$ that is true, where each of the α and α_i's is a conjunction of arithmetic comparisons. If all its partial containment implications are direct implications, then there exists a single disjunct α_i in the rhs of the containment implication such that $\alpha \Rightarrow \alpha_i$.*

We give conditions to guarantee direct implications in containment test.

Corollary 1. *Consider the normalized queries Q'_1 and Q'_2 in Theorem 1. Suppose all partial containment implications are direct. Then there is a mapping μ_i from $Q'_{1,0}$ to $Q'_{2,0}$ such that $\beta'_2 \Rightarrow \mu_i(\beta'_1)$.*

[4] Notice that containment implications and their partial containment implications are not necessarily related to mappings and query containment, only the names are borrowed.

4.2 Left Semi-interval Comparisons (LSI) for Q_1

We first consider the case where Q_1 is a conjunctive query with left semi-interval arithmetic comparison subgoals only (i.e., one of the form $X < c$ or $X \leq c$ or both may appear in the same query). The following theorem is a main result describing the conditions for the homomorphism property to hold in this case.

Theorem 4. *Let Q_1 be a conjunctive query with left semi-interval arithmetic comparisons and Q_2 a conjunctive query with any arithmetic comparisons. If they satisfy all the following conditions, then the homomorphism property holds:*

- Condition (i)-lsi*: There do not exist subgoals as follows which all share the same constant: An open-LSI subgoal in $AC(Q_1)$, a closed-LSI subgoal in closure of $AC(Q_2)$, and a subgoal in $core(Q_1)$.*
- Condition (ii)-lsi*: Either $core(Q_1)$ has no shared variables or there do not exist subgoals as follows which all share the same constant: An open-LSI subgoal in $AC(Q_1)$, a closed-LSI subgoal in the closure of $AC(Q_2)$ and, a subgoal in $core(Q_2)$.*
- Condition (iii)-lsi*: Either $core(Q_1)$ has no shared variables or there do not exist subgoals as follows which all share the same constant: An open-LSI subgoal in $AC(Q_1)$ and two closed-LSI subgoals in the closure of $AC(Q_2)$.*

It is straightforward to construct corollaries of Theorem 4 with simpler conditions. The following is an example.

Corollary 2. *Let Q_1 be a conjunctive query with left semi-interval arithmetic comparisons and Q_2 a conjunctive query with any arithmetic comparisons. If the arithmetic comparisons in Q_1 do not share a constant with the closure of the arithmetic comparisons in Q_2, then the homomorphism property holds.*

The results in Theorem 4 can be symmetrically stated for RSI queries as containing queries. The symmetrical conditions of Theorem 4 for the RSI case will be referred to as conditions (i)-rsi, (ii)-rsi, and (iii)-rsi, respectively.

4.3 Semi-Interval (SI) and Point-Inequalities (PI) Queries for Q_1

Now we extend the result of Theorem 4 to treat both LSI and RSI subgoals occurring in the same containing query. We further extend it to include point inequalities (of the form $X \neq c$). The result is the following.

SI Queries for Q_1: We consider the case where Q_1 has both LSI and RSI inequalities called "SI inequalities," i.e., any of the $<$, $>$, $\leq$, and $\geq$. In this case we need one more condition, namely Condition (iv), in order to avoid coupling implications. Thus Theorem 4 is extended to the following theorem, which is the second main result of this section.

Theorem 5. *Let Q_1 be a conjunctive query with left semi-interval and right semi-interval arithmetic comparisons and Q_2 a conjunctive query with SI arithmetic comparisons. If they satisfy all the following conditions, then the homomorphism property holds:*

- *Conditions (i)-lsi, (ii)-lsi, (iii)-lsi, (i)-rsi, (ii)-rsi, and (iii)-rsi.*
- *Condition (iv)-si: Any constant in an RSI subgoal of Q_1 is strictly greater than any constant in an LSI subgoal of Q_1.*

We refer to the last conditin as (iv)-si.

PI Queries for Q_1: If the containing query Q_1 has point inequalities, three more forms of coupling implications can occur. Thus Theorem 5 is further extended to Theorem 6, which is the third main result of this section.

Theorem 6. *Let Q_1 be a conjunctive query with left semi-interval and right semi-interval and point inequality arithmetic comparisons and Q_2 a conjunctive query with SI arithmetic comparisons. If Q_1 and Q_2 satisfy all the following conditions, then the homomorphism property holds:*

- *Conditions (i)-lsi, (ii)-lsi, (iii)-lsi, (i)-rsi, (ii)-rsi, (iii)-rsi and (iv)-si.*
- *Condition (v)-pi: Either Q_1 has no repeated variables, or it does not have point inequalities.*
- *Condition (vi)-pi: Point-Inequality(Q_1) does not have a constant that occurs in core(Q_1), or Closed-LSI(Q_1), or Closed-RSI(Q_1).*

4.4 Beyond Semi-Interval Queries for Q_1

Our results have already captured subtle cases where the homomorphism property holds. There is not much hope beyond those cases, unless we restrict the number of subgoals of the contained query, which is known in the literature (e.g., [14]). Couplings due to the implication:

$$TRUE \Rightarrow \big((X \leq Y) \vee (Y \leq X)\big)$$

indicate that if the containing query has closed comparisons, then the homomorphism does not hold. The following is such an example:

$$Q_1 : ans() \text{ :- } p(X,Y), X \leq Y.$$
$$Q_2 : ans() \text{ :- } p(X,Y), p(Y,X).$$

Clearly Q_2 is contained in Q_1, but the homomorphism property does not hold.

4.5 A Testing Algorithm

We summarize the results in this section in an algorithm shown in Figure 2. Given two CQAC queries Q_1 and Q_2, the algorithm tests if the homomorphism property holds in checking $Q_2 \sqsubseteq Q_1$. Queries may not satisfy these conditions but still the homomorphism property may hold. For instance, it could happen if they do not have self-joins, or if domain information yields that certain mappings are not possible (see Section 5). Hence, in the diagram, we can also put this additional check: Whenever one of the conditions is not met, we also check whether there are mappings that would enable a coupling implication. We did not include the formal results for this last test for brevity, as they are a direct consequence of the discussion in the present section.

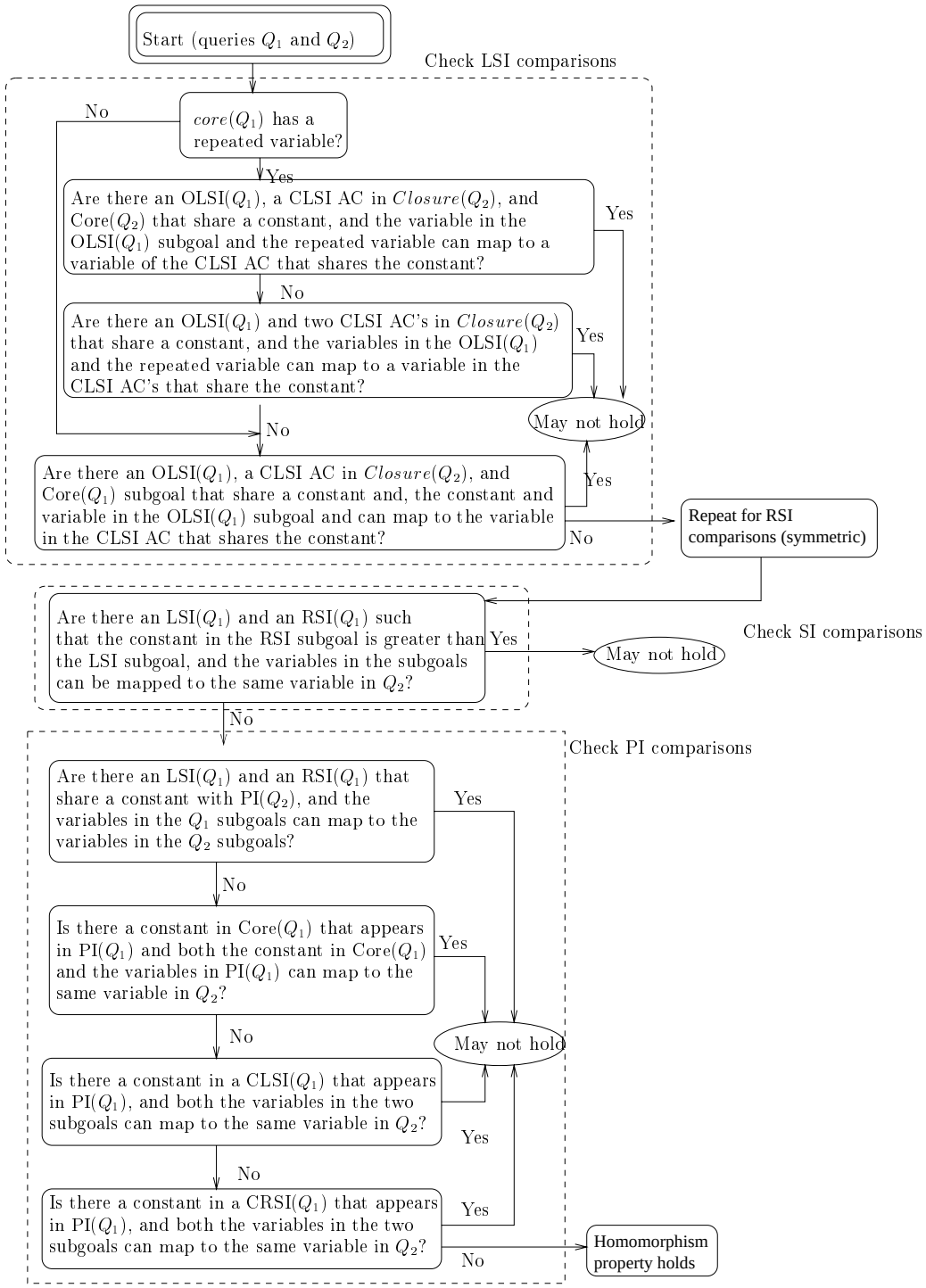

Fig. 2. An algorithm for checking homomorphism property in testing $Q_2 \sqsubseteq Q_1$.

5 Improvements Using Domain Information

So far we have discussed in what cases we do not need to normalize queries in the containment test, and in what cases we can reduce the containment test to checking the existence of a single homomorphism. If a query does not satisfy these conditions, the above results become inapplicable. For instance, often a query may have both $<$ and $\geq$ comparisons, not satisfying the conditions in Theorem 2. In this section, we study how to relax these conditions by using domain knowledge of the relations and queries.

The intuition of our approach is the following. We partition relation attributes into different domains, such as "car models," "years," and "prices." We can safely assume that for realistic queries, their conditions respect these domains. In particular, for a comparison $X\ \theta\ A$, where X is a variable, A is a variable or a constant, the domain of A should be the same as that of X. For example, it may be meaningless to have conditions such as "carYear = \$6,000." Therefore, in the implication of testing query containment, it is possible to partition the implication into different domains. The domain information about the attributes is collected only once before queries are posed. For instance, given the following implication ϕ: $year > 2000 \wedge price \leq \$5,000 \Rightarrow year > 1998 \wedge price \leq \$6,000$. We do not need to consider implication between constants or variables in different domains, such as between "1998" and "\$6,000," and between "year" and "price." As a consequence, this implication can be projected to the following implications in two domains:

Year domain ϕ_y: $year > 2000 \Rightarrow year > 1998$.
Price domain ϕ_p: $price \leq \$5,000 \Rightarrow price \leq \$6,000$.

We can show that ϕ is true iff both ϕ_y and ϕ_p are true. In this section, we first formalize this domain idea, and then show how to partition an implication into implications of different domains.

5.1 Domains of Relation Attributes and Query Arguments

Assume each attribute A_i in a relation $R(A_1, \ldots, A_k)$ has a domain $Dom(R.A_i)$. Consider two tables: `house(seller, street, city, price)` and `crimerate(city, rate)`. Relation `house` has housing information, and relation `crimerate` has information about crime rates of cities. The following table shows the domains of different attributes in these relations. Notice that attributes `house.city` and `crimerate.city` share the same domain: $D_3 = \{\text{city names}\}$.

Attribute	Domain
`house.seller`	$D_1 = \{\text{person names}\}$
`house.street`	$D_2 = \{\text{street names}\}$
`house.city`	$D_3 = \{\text{city names}\}$
`house.price`	$D_4 = \{\text{float numbers in dollars}\}$
`crimerate.city`	$D_3 = \{\text{city names}\}$
`crimerate.rate`	$D_5 = \{\text{crime-rate float numbers}\}$

We equate domains of variables and constants using the following rules:

- For each argument X_i (either a variable or a constant) in a subgoal $R(X_1, \ldots, X_k)$ in query Q, the domain of X_i, $Dom(X_i)$, is the corresponding domain of the j-th attribute in relation R.
- For each comparison $X\ \theta\ c$ between variable X and constant c, we set $Dom(c) = Dom(X)$. Constants from different domains are always treated as different constants. For instance, in two conditions $carYear = 2000$ and $carPrice = \$2000$, constants "2000" and "\$2000" are different constants.

We perform this process on all subgoals and comparisons in the query. In this calculation we make the following realistic assumptions: (1) If X is a shared variable in two subgoals, then the corresponding attributes of the two arguments of X have the same domain. (2) If we have a comparison $X\ \theta\ Y$, where X and Y are variables, then $Dom(X)$ and $Dom(Y)$ are always the same.

Consider the following queries on the relations above.

P_1: $ans(t_1, c_1)$:- $house(s_1, t_1, c_1, p_1), crimerate(c_1, r_1), p_1 \leq \$300,000, r_1 \geq 3.0\%$.
P_2: $ans(t_2, c_2)$:- $house(s_2, t_2, c_2, p_2), crimerate(c_2, r_2), p_2 \leq \$250,000, r_2 \geq 3.5\%$.

The computed domains of the variables and constants are shown in the table below. It is easy to see that the domain information as defined in this section can be obtained in polynomial time.

P_1: Variable/constant	P_1: Domain	P_2: Variable/constant	P_2: Domain
s_1	D_1	s_2	D_1
t_1	D_2	t_2	D_2
c_1	D_3	c_2	D_3
p_1	D_4	p_2	D_4
r_1	D_5	r_2	D_5
$\$300,000$	D_4	$\$250,000$	D_4
3.0%	D_5	3.5%	D_5

5.2 Partitioning Implication into Domains

According to Theorem 1, to test the containment $Q_1 \sqsubseteq Q_2$ for two given queries Q_1 and Q_2, we need to test the containment implication in the theorem. We want to partition this implication to implications in different domains, since testing the implication in each domain is easier. Now we show that this partitioning idea is feasible. We say a comparison $X\ \theta\ A$ is in domain D if X and A are in domain D. The following are two important observations.

- If a mapping μ_i maps an argument X in query Q_1 to an argument Y in query Q_2, based on the calculation of argument domains, clearly X and Y are from the same domain.
- In query normalization, each new introduced variable has the same domain as the replaced argument (variable or constant).

Definition 4. *Consider the following implication ϕ in Theorem 1:*

$$\beta'_2 \Rightarrow \mu_1(\beta'_1) \vee \ldots \vee \mu_k(\beta'_1).$$

For a domain D of the arguments in ϕ, the projection of ϕ in D*, denoted ϕ_D, is the following implication:*

$$\beta'_{2,D} \Rightarrow \mu_1(\beta'_{1,D}) \vee \ldots \vee \mu_k(\beta'_{1,D}).$$

$\beta'_{2,D}$ includes all comparisons of β'_2 in domain D. Similarly, $\beta'_{1,D}$ includes all comparisons of β'_1 in domain D.

Suppose we want to test $P_2 \sqsubseteq P_1$ for the two queries above. There is only one containment mapping from P_1 to P_2, and we need to test the implication:

$$\pi : p_2 \leq \$250,000, r_2 \geq 3.5\% \Rightarrow p_2 \leq \$300,000, r_2 \geq 3.0\%.$$

The projection of π on domain D_4 (float numbers in dollars) π_{D_4} is $p_2 \leq \$250,000 \Rightarrow p_2 \leq \$300,000$. Similarly, π_{D_5} is $r_2 \geq 3.5\% \Rightarrow r_2 \geq 3.0\%$.

Theorem 7. *Let $D_1, \ldots, D_k$ be the domains of the arguments in the implication ϕ. Then ϕ is true iff all the projected implications $\phi_{D_i}, \ldots, \phi_{D_k}$ are true.*

In the example above, by Theorem 7, π is true iff π_{D_4} and π_{D_5} are true. Since the latter two are true, π is true. Thus $P_2 \sqsubseteq P_1$. In general, we can test the implication in Theorem 1 by testing the implications in different domains, which are much cheaper than the whole implication. [11] gives formal results that relax the conditions in the theorems of the previous section to apply only on elements of the same domain.

6 Experiments

In this section we report on experiments to determine whether the homomorphism property holds for real queries. We checked queries in many introductory database courses available on the Web, some data-mining queries provided by Microsoft Research, and the TPC-H benchmark queries [20]. We have observed that, for certain applications (e.g., the data-mining queries), usually queries do not have self-joins; thus the homomorphism property holds. In addition, among the queries that use only semi-interval (SI) and point inequality (PI) comparisons, the majority have the homomorphism property.

For a more detailed discussion, we focus on our evaluation results on the TPC-H benchmark queries [20], which represent typical queries in a wide range of decision-support applications. To the best of our knowledge, our results are the first evidence that containment is easy for those queries. The following is a summary of our experiments on the TPC-H benchmark queries.

1. All, except two (Q_4 and Q_{21}) of the 22 queries use semi-interval comparisons (SI's) and point inequalities (PI's) only.

2. When the homomorphism property may not hold, it is always because of the following situation: a variable X (usually of "date" type) is bounded in an interval between two constants. In such a case, the property is guaranteed to hold if the contained query does not contain self-joins of the subgoal that uses a variable that X can map to.
3. As a consequence, if the contained query is also one of the 22 queries, since they do not have self-joins of relations that share a variable with SI predicates, the homomorphism property holds.

The detailed experimental results are in [11]. Here we use the following query adapted from TPC-H query Q_3 as an example. (For simplicity we call this query Q_3.) We show how to apply the results in the earlier sections to test the following: in testing if Q_3 is containing another CQAC query, does the homomorphism property hold in the test?

```
SELECT l_orderkey, l_extendedprice, l_discount, o_orderdate, o_shippriority
FROM   customer, orders, lineitem
WHERE  c_mktsegment = '[SEGMENT]'
       AND c_custkey = o_custkey AND l_orderkey = o_orderkey
       AND o_orderdate < date '[DATE]' AND l_shipdate > date '[DATE]';
```

Consider the case where we check for the containment of any conjunctive query with semi-interval arithmetic comparisons in the above query Q_3. We shall apply Theorem 5. Notice that the above query has shared variables (expressed by the equality $c\_custkey = o\_custkey$ in the `WHERE` clause), as well as it contains both LSI and RSI arithmetic comparisons. However the variables $o\_orderdate$ (used in a comparison) and $c\_custkey$ (a shared variable) are obviously of different domains. Hence conditions (ii)-lsi, (ii)-rsi, (iii)-lsi, (iii)-rsi are satisfied. Also using domain information, we see that (i)-lsi and (i)-rsi are satisfied.

In general, the condition (iv) in Theorem 5 may not be satisfied, but the scenario in which it is not satisfied either uses a query with a self-join on relation `lineitem` or a self-join on relation `orders`. Such a query (a) is not included in the benchmark, and (b) would ask for information that is not natural or is of a very specific and narrow interest (e.g., would ask of pairs of orders sharing a property). Consequently, to test containment of any natural SI query in Q_3, we need only one containment mapping. Notice that without using the domain information, we could not derive this conclusion.

7 Conclusion

In this paper we considered the problem of testing containment between two conjunctive queries with arithmetic comparisons. We showed in what cases the normalization step in the algorithm [5, 6] is not needed. We found various syntactic conditions on queries, under which we can reduce considerably the number of mappings needed to test containment to a single mapping (homomorphism property). These syntactic conditions can be easily checked in polynomial time.

Our experiments using real queries showed that many of these queries pass this test, so they do have the homomorphism property, making it possible to use more efficient algorithms for the test.

Acknowledgments: We thank Microsoft Research for providing us their data-mining queries to do our experiments.

References

1. Chaudhuri, S., Krishnamurthy, R., Potamianos, S., Shim, K.: Optimizing queries with materialized views. In: ICDE. (1995) 190–200
2. Theodoratos, D., Sellis, T.: Data warehouse configuration. In: Proc. of VLDB. (1997)
3. Ullman, J.D.: Information integration using logical views. In: ICDT. (1997) 19–40
4. Halevy, A.: Answering queries using views: A survey. In: Very Large Database Journal. (2001)
5. Gupta, A., Sagiv, Y., Ullman, J.D., Widom, J.: Constraint checking with partial information. In: PODS. (1994) 45–55
6. Klug, A.: On conjunctive queries containing inequalities. Journal of the ACM **35** (1988) 146–160
7. Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational data bases. STOC (1977) 77–90
8. van der Meyden, R.: The complexity of querying indefinite data about linearly ordered domains. In: PODS. (1992)
9. Levy, A., Mendelzon, A.O., Sagiv, Y., Srivastava, D.: Answering queries using views. In: PODS. (1995) 95–104
10. Afrati, F., Li, C., Mitra, P.: Answering queries using views with arithmetic comparisons. In: PODS. (2002)
11. Afrati, F., Li, C., Mitra, P.: On containment of conjunctive queries with arithmetic comparisons (extended version). Technical report, UC Irvine (2003)
12. Saraiya, Y.: Subtree elimination algorithms in deductive databases. Ph.D. Thesis, Computer Science Dept., Stanford Univ. (1991)
13. Qian, X.: Query folding. In: ICDE. (1996) 48–55
14. Kolaitis, P.G., Martin, D.L., Thakur, M.N.: On the complexity of the containment problem for conjunctive queries with built-in predicates. In: PODS. (1998) 197–204
15. Chandra, A., Lewis, H., Makowsky, J.: Embedded implication dependencies and their inference problem. In: STOC. (1981) 342–354
16. Cosmadakis, S.S., Kanellakis, P.: Parallel evaluation of recursive queries. In: PODS. (1986) 280–293
17. Chaudhuri, S., Vardi, M.Y.: On the equivalence of recursive and nonrecursive datalog programs. In: PODS. (1992) 55–66
18. Shmueli, O.: Equivalence of datalog queries is undecidable. Journal of Logic Programming **15** (1993) 231–241
19. Wang, J., Maher, M., Toper, R.: Rewriting general conjunctive queries using views. In: 13th Australasian Database Conf. (ADC), Melbourne, Australia, ACS (2002)
20. TPC-H: `http://www.tpc.org/tpch/` (2003)