# On Containment of Conjunctive Queries with Arithmetic Comparisons (Extended Version)

**UCI ICS Technical Report, June 2003**

Foto Afrati
National Technical University of Athens
157 73 Athens, Greece
afrati@cs.ece.ntua.gr

Chen Li
Information and Computer Science
University of California, Irvine, CA 92697, USA
chenli@ics.uci.edu

Prasenjit Mitra
Department of Computer Science
Stanford University, CA 94305, USA
mitra@db.stanford.edu

## Abstract

*In this paper we study the following problem: how to test whether $Q_2$ 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 is mainly motivated by (1) reducing the problem complexity from $\Pi_2^P$-completeness to NP-completeness in these cases; and (2) utilizing 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. 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, more importantly, reducing the number of containment mappings. (2) We find large classes of queries where 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 have this homomorphism property.*

## 1 Introduction

The problem of testing query containment is stated as follows: how to test whether a query $Q_2$ is *contained* in a query $Q_1$, i.e., if for any database $D$, the set of answers to $Q_2$ is a subset of the answers to $Q_1$? This problem arises in a large variety of database applications, such as query evaluation and optimization [5], data warehousing [17], and data integration using views [19]. 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 [9].

A class of queries of great significance is select-project-join queries, a.k.a. conjunctive queries. These queries are widely used in many database applications. Often, users pose queries using conjunctive queries along with arithmetic comparisons (e.g., year $>$ 2000, price $\leq$ 5000). Thus testing containment of conjunctive queries with arithmetic comparisons becomes very important. Several algorithms have been proposed for testing containment of conjunctive queries with arithmetic comparisons (e.g., [8, 10]). These algorithms first normalize the queries by replacing constants and shared variables with new distinct variables and add the corresponding comparisons, then test the containment by checking a logical implication using *multiple* containment mappings. (See Section 2 for detail.)

In this paper, 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 [10]?

The motivation of studying these problems is three fold. The first one is the efficiency of the test procedure. Whereas the problem of containment of pure conjunctive queries is known to be NP-complete [4], the problem of containment of conjunctive queries with arithmetic comparisons is $\Pi_2^P$-

complete [10, 20]. 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 complexity of the problem. In this work, we find large classes of queries where the homomorphism property holds. Thus, we can reduce the complexity of the problem to NP. Even though the saving on the normalization step does not put the problem in a different complexity class, it can still reduce the size of the queries, as well as the number of mappings required to test containment.

The second reason is that the homomorphism property can simplify many problems such as that of answering queries using views [12], in which we want to construct a plan using views to compute the answer to a query. It is shown in [1] 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. Otherwise, we can just consider plans that are also conjunctive queries with comparisons. Hence, if we know the homomorphism property holds by analyzing the query and views, we can develop efficient algorithms for constructing a plan using views.

The third motivation is that, in studying realistic queries (e.g., those TPC benchmarks), we found that it is very hard to construct examples that need multiple mappings in the containment test. In other words, we observed that most real queries only need a single containment mapping to test the containment. If we can easily identify the cases in which a single containment mapping is sufficient to prove containment between queries, the containment test becomes substantially less expensive. Therefore, we want to derive syntactic conditions on queries, under which the homomorphism property holds and a single containment mapping suffices to prove containment. These syntactic conditions can be easily checked in polynomial time. In this paper we develop such conditions.

In the problem of testing whether query $Q_2$ is contained in query $Q_1$, we call $Q_2$ the *contained query* and $Q_1$ the *containing query*. The following are our contributions of this work. (Table 1 is a summary of results.)

1. We show cases where the normalization step in the test is not needed when their comparisons do not imply equalities. These cases are when the containing query $Q_1$ has only $\leq$ or $\geq$ comparisons, or the comparisons of $Q_1$ do not imply equalities and the homomorphism property holds. (Section 3).

2. When the containing query $Q_1$ has only arithmetic comparisons between a variable and a constant (called "semi-intervals," or "SI" for short), we present cases where the homomorphism property holds (Section 4). We show that if the homomorphism property does not hold, then some "heavy" constraints must be satisfied. An example of such a constraint is: 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 on the size of the queries. In general, all the constraints in this paper are syntactic and can be checked in polynomial time. Moreover, they are tight in that if one constraint is violated, we can give examples to show that the homomorphism property does not hold.

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 that many of them satisfy the conditions under which the homomorphism property holds (Section 6).

### 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 [15], 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 [14].

Klug [10] has shown that containment for conjunctive queries with comparison predicates is in $\Pi_2^P$, and it is proven to be $\Pi_2^P$-hard in [20]. The reduction only used $\neq$. This result is extended in [11] to use at most three occurrences of the same predicate name in the contained query (and also only $\neq$). The same reduction shows that it remains $\Pi_2^P$-complete even in the case where the containing query is acyclic, thus the results in [14] do not extend to conjunctive queries with $\neq$. The complexity is reduced to co-NP in [11] 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 [8, 10]. They show 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.

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 [3,

| Contained Query | Containing Query | Complexity | References |
|---|---|---|---|
| CQ | CQ | NP | [4] |
| CQ with closed LSI | CQ with closed LSI | NP | [8, 10] |
| CQ with open LSI | CQ with open LSI | NP | [8, 10] |
| CQ with AC | CQ with closed LSI | NP | Section 4 |
| CQ with AC Constraints | CQ with LSI (i)-lsi, (ii)-lsi, (iii)-lsi | NP | Section 4 Theorem 4.2 |
| CQ with SI Constraints | CQ with LSI, RSI (i)-lsi,rsi, (ii)-lsi,rsi, (iii)-lsi,rsi, (iv) | NP | Section 4 Theorem 4.3 |
| CQ with SI Constraints | CQ with LSI, RSI, PI as above and (v),(vi),(vii) | NP | Section 4 Theorem 4.4 |
| CQ with AC | CQ with AC | $\Pi_2^P$ | [20] |

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

7]. Containment among recursive and nonrecursive datalog queries is also studied in [6, 16].

In [1] we studied the problem of how to rewrite a query using views if both the query and views are conjunctive queries with arithmetic comparisons. The algorithm depends upon the checking for containment among queries. Besides showing the necessity of using recursive plans if the homomorphism property does not hold, we also developed an algorithm applicable when the property holds. In this work, we identify the conditions under which the homomorphism property holds. Thus the results in [1] 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 introduce the concept of minimal inequality graphs and define the homomorphism property.

**Definition 2.1** (**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 $Q_2 \sqsubseteq Q_1$. □

A conjunctive query (i.e., select-project-join query) is in the form:

$$h(\bar{X}) \text{ :- } g_1(\bar{X}_1), \ldots, g_k(\bar{X}_k)$$

In each subgoal $g_i(\bar{X}_i)$, predicate $g_i$ is a *base relation*, and every argument in the subgoal is either a variable or a constant.

Chandra and Merlin [4] 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 $Q_2$. We refer to $Q_1$ as the *containing query* and to $Q_2$ as the *contained query*.

Let $Q$ be a conjunctive query with arithmetic comparisons (CQAC). We consider the following arithmetic comparisons (inequalities): $<, \leq, >, \geq$, 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. (Otherwise the query returns the empty set as an answer on every database instance.)
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. The fix is easy. If the comparisons can imply an equality $X = Y$, we can 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 the set of all possible arithmetic comparisons that can be logically derived from $S$. For example, for the set of arithmetic comparisons $S = \{X \leq Y, Y = c\}$, the $Closure(S) = \{X \leq Y, Y = c, X \leq c\}$. 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 |
| 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 $c$ denotes 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 [8, 10], 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 original queries and the normalized queries are shown 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 mappings from $Q_{1,0}$ to $Q_{2,0}$. There are a few important observations to be noticed.

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 that $C_{new}$ 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 variables to different variables. However, after normalizing the two 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).

The following theorem is from [8, 10].

**Theorem 2.1** *$Q_2 \sqsubseteq Q_1$ if and only if the following logical implication $\phi$ 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.* □

**EXAMPLE 2.1** This example from [21] shows that the normalization step in Theorem 2.1 is critical. Consider two queries $Q_1$ and $Q_2$.

$$\begin{array}{ll} Q_1 : h(W) \text{ :- } & q(W), p(X, Y, Z, Z', U, U), \\ & X < Y, Z > Z'. \\ Q_2 : h(W) \text{ :- } & q(W), p(X, Y, 2, 1, U, U), \\ & p(1, 2, X, Y, U, U), \\ & p(1, 2, 2, 1, X, Y). \end{array}$$

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

$$\delta_1 : W \to W, X \to X, Y \to Y, Z \to 2, Z' \to 1, U \to U.$$
$$\delta_2 : W \to W, X \to 1, Y \to 2, X \to X, Z' \to Y, U \to U.$$

Notice that 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 variables $X$ and $Y$.

We can show that $Q_2 \sqsubseteq Q_1$, but the following impllication is *not* true:

$$\text{TRUE} \Rightarrow \delta_1(X < Y, Z > Z') \vee \delta_2(X < Y, Z > Z').$$
$$\text{TRUE} \Rightarrow (X < Y \wedge 2 > 1) \vee (1 < 2 \wedge X > Y).$$
$$\text{TRUE} \Rightarrow (X < Y) \vee (X > Y).$$

The reason the implication does not hold is that it is possible $X = Y$. However, in this case, we would have a new containment mapping from $Q_1$ to $Q_2$.

$$\delta_3 : W \to W, X \to 1, Y \to 2, Z \to 2, Z' \to 1, U \to X = Y$$

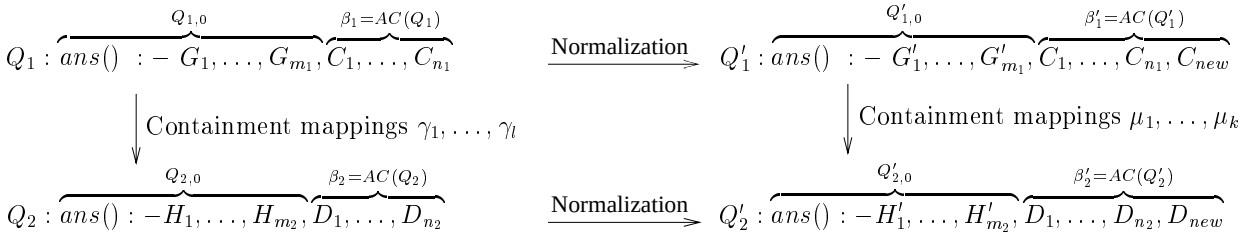


**Figure 1. Containment testing.**

In fact, 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 2.2** This example shows that the $\vee$ operation in the implication in Theorem 2.1 is critical. Consider two queries:

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

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 $Q'_{2,0}$:

$$\mu_1 : X \to A, Y \to B. \quad \mu_2 : X \to C, Y \to D.$$

We can show that:

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

That is:

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

Therefore, $Q_2 \sqsubseteq Q_1$. Notice that both containment mappings are needed to prove the implication. □

## 2.2 Challenges

There are several challenges in using Theorem 2.1 to test whether $Q_2 \sqsubseteq Q_1$. (1) The queries look less intuitive after the normalization. The computational cost of testing the implication $\phi$ 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

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

In Section 4 we discuss in what cases we only need a single containment mapping from $Q'_{1,0}$ to $Q'_{2,0}$ that can satisfy the implication. That is, we want to know in what cases the following is true. $Q_2 \sqsubseteq Q_1$ if and only if there is a containment mapping $\mu$ from $Q'_{1,0}$ to $Q'_{2,0}$, such that $\beta'_2 \Rightarrow \mu(\beta'_1)$. If this claim, we say that the containment testing has the *homomorphism property*.

Besides reducing the complexity of the problem of query containment, the homomorphism property is useful also in applications such as query rewriting using views [12]. At present we only know of efficient algorithms for constructing query rewritings using views that use heavily this property [13, 2, 1].

## 2.3 Inequality Graphs

To give the intuition of a logical implication and make it easy to reason about implications, we define the concept of *inequality graphs*.[1] Given a set $\beta$ of comparisons, the *inequality graph* of $\beta$, denoted $\mathcal{G}(\beta)$, is a graph constructed as follows:

- The nodes in the graph are the variables and constants in $\beta$.
- There is a directed arc from node $n_1$ to node $n_2$ if the inequality $n_1 < n_2$ or $n_1 \leq n_2$ is in $\beta$, or both $n_1$ and $n_2$ are constants and $n_1 < n_2$. Such an arc is labeled as "$<$" or "$\leq$" according to whether the inequality is strict ($<$) or nonstrict ($\leq$).
- If $n_1 = n_2$ is in $\beta$, there is a $\leq$-arc in each direction between $n_1$ and $n_2$.

[1] A similar concept was introduced in [10]. Here we include inequalities between constants automatically, and introduce the concept of "minimal inequality graph" to remove implied equalities such as $X = Y$.

As an example, the inequality graph of the comparisons $S = \{W \leq 4, X \geq 5, X < Y, X \leq Z, U = Z, Y \leq U\}$ is shown in Figure 2.

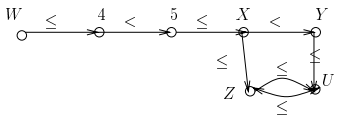


**Figure 2. Inequality Graph.**

The implication from a set $\beta$ of inequalities to an inequality can be explained using the graph as follows. $\beta$ implies an inequality $A_1 \leq A_2$, where $A_1$ and $A_2$ are either a variable or a constant, if $\mathcal{G}(\beta)$ has a directed path from $A_1$ to $A_2$. $\beta$ implies $A_1 < A_2$ if there is a directed path $A_1$ to $A_2$ having at least one $<$-arc. Hence an implication can be checked in polynomial time in the number of comparisons in the set.

To simplify the reasoning about implications, we want to merge those nodes in a graph that are implied to be equal. $\beta$ implies equality $A_1 = A_2$ if there is a directed cycle of $\leq$-arcs. The *minimal inequality graph* of a set $\beta$ of inequalities, denoted as $\mathcal{MG}(\beta)$, is obtained from the inequality graph $\mathcal{G}(\beta)$ as follows. For those nodes that are implied to be equal by $\beta$, we merge them to a single node, and correspondingly, let those arcs starting from (resp. ending at) these nodes start from (resp. end at) the merged node. For instance, Figure 3 shows the minimal inequality graph of the inequalities $S$.

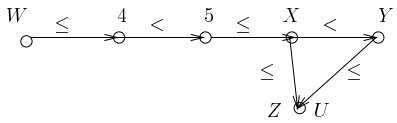


**Figure 3. Minimal inequality graph.**

Notice that in the containment implication $\phi$ we are considering, since the right-hand side is a disjunction, the left-hand side may not imply each of the disjuncts. For instance, $TRUE \Rightarrow (X < Y) \vee (X = Y) \vee (X > Y)$, but $TRUE$ cannot imply any of the three disjuncts. This disjunction in the containment implication makes the reasoning about the implication more challenging. In addition, we have the following propositions.

**Proposition 2.1** *If the set of comparisons $AC(Q)$ in a query $Q$ does not imply equalities, then the inequality graph $\mathcal{G}(AC(Q))$ is already minimal.* □

**Proposition 2.2** *After normalizing a query $Q$ to $Q'$, the minimal inequality graphs $\mathcal{MG}(AC(Q'))$ and $\mathcal{MG}(AC(Q))$ are isomorphic.* □

In the rest of the paper, we use the "graph" of an inequality set to mean the "minimal inequality graph" of the set.

## 2.4 Homomorphism Property

**Definition 2.2 (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 $\mu$ from $core(Q_1)$ to $core(Q_2)$ such that $AC(Q_2) \Rightarrow \mu(AC(Q_2))$. If $\mathcal{Q}_1 = \mathcal{Q}_2 = \mathcal{Q}$, then we say containment testing has the homomorphism property for class $\mathcal{Q}$. □

The containment test of Theorem 2.1 forj general CQACs, sonsiders normalized queries. However, in the cases where a single mapping suffices to show containment between normalized queries, it slso 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 and we refer to queries either in normalized form or not.

In the 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 $\mu$ from $core(Q_1)$ to $core(Q_2)$ and check whether $\mu$ is a containment mapping with respect to the AC subgoals too (the latter meaning that an AC subgoal $g$ maps on a AC subgoal $g'$ so that it holds $g' \Rightarrow g$ ). Note that the number of mappings is exponential on the size of the queries.

It is shown in [10] that for the class of conjunctive queries with only open-LSI (open-RSI respectively) comparisons, then 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 then the homomorphism property holds.

**Definition 2.3 (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 $\mu$ from $core(Q_1)$ to $core(Q_2)$ such that $AC(Q_2) \Rightarrow \mu(AC(Q_2))$. □

The constraints we use are given as *syntactic conditions* that relate subgoals, in both queries, which contain the same constant. 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

In order to use the result in Theorem 2.1 to test the containment of two queries $Q_1$ and $Q_2$, we need to normalize them first. This normalization step could make the resulting queries look less intuitive, as shown in Example 2.2. 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 $\beta_2 \Rightarrow \gamma_1(\beta_1) \vee \ldots \vee \gamma_l(\beta_1)$.

### 3.1 Case 1

The following theorem says that Theorem 2.1 is still true even for non-normalized queries $Q_1$, if two conditions are satisfied by the queries: (1) $\beta_1$ contains only $\leq$ and $\geq$, and (2) $\beta_1$ (correspondingly $\beta_2$) do not imply equalities. The intuitive reason why in this case we can restrict the space of mappings is due to the monotonicity property: For a CQ $Q$ whose AC's only include $\leq$ and $\geq$, if a tuple $t$ of a database $D$ is an answer to $Q$, then consider any database $D'$ obtained from $D$ by identifying some elements. That is, the constants of $D'$ is a subset of the constants of $D$ and the tuples of $D'$ are copies of the tuples of $D$ where however, we have replaced a group of different constants with the same constant. We can show that the corresponding tuple $t'$ is in the answer $Q(D')$.

**Theorem 3.1** *Consider two CQAC queries $Q_1$ and $Q_2$ shown in Figure 1 that may not be normalized. Suppose $\beta_1$ contains only $\leq$ and $\geq$, and $\beta_1$ (correspondingly $\beta_2$) 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 $Q_{2,0}$.* □

[[[ **By Chen: Check the following proof.** ]]]

**Proof:** "Only if": Let $V$ be a set of variables that is partitioned into subsets $V_1, \ldots, \mathcal{V}_j$. An *ordering* of the variables in $V$ corresponds to a total order $<$ on the subsets of the partition. For any pair of variables $X, Y \in V$, $X < Y$ iff $V_X < V_Y$ where $X$ belongs to a subset $V_X$ and $Y$ belongs to a subset $V_Y$ and $X = Y$ iff $X$ and $Y$ belong to the same subset. Given a set $V$ of variables, a *distinct ordering* of all variables in $V$ is an ordering which is obtained when each subset in the partition contains exactly one element. Each ordering can be viewed as a conjunction of inequalities/equalities which holds on the variables for this ordering. For example, let $V = \{X, Y, Z\}$. For the partition $V_1 = \{X, Y\}$, $V_2 = \{Z\}$ and the total order $V_1 < V_2$, we define the ordering $o$ $X = Y < Z$ which can be viewed as $o \equiv X = Y \wedge Y < Z$. Clearly if $o_1, \ldots, o_n$ are *all* orderings on a set of variables then $true \Rightarrow o_1 \vee \ldots \vee o_n$.

Suppose

$$Q_2 \sqsubseteq Q_1$$

but

$$\beta_2 \not\Rightarrow \gamma_1(\beta_1) \vee \ldots \vee \gamma_k(\beta_1).$$

Thus there is an instantiation $\rho$ of the set $X$ of the variables in $\beta_2, \gamma_1(\beta_1), \ldots, \gamma_k(\beta_1)$, such that $\rho(\beta_2)$ holds, but

$$\rho(\gamma_1(\beta_1)) \vee \ldots \vee \rho(\gamma_k(\beta_1))$$

does not.

Therefore, each $\rho(\gamma_i(\beta_1))$ is false. Then there is at least one built-in predicate $a_i \theta b_i$ in $\rho(\gamma_i(\beta_1))$, such that $a_i \theta b_i$ is false. Here assume $a_i = \rho(A_i)$, and $b_i = \rho(B_i)$, where $A_i$ and $B_i$ are two variables in $X$. Recall that $\beta_1$ uses $\leq$ only, thus $\theta$ is $\leq$. As a consequence, $a_i \leq b_i$ in $\rho(\gamma_i(\beta_1))$ is false, meaning $a_i > b_i$, so $a_i \neq b_i$.

Step 1: We want to change the instantiation $\rho$ to another instantiation $\rho'$, which assigns different constants to different variables in $X$. In addition, $\rho'(\beta_2)$ still holds, and

$$\rho'(\gamma_1(\beta_1)) \vee \ldots \vee \rho'(\gamma_k(\beta_1))$$

does not. Notice that the instantiation $\rho$ could assign the same constant to two different variables $V_1$ and $V_2$ in $X$. In this case, if $V_1$ and $V_2$ both appear in a built-in predicate in $\beta_1$,[2] wlog, we assume $V_1 \leq V_2$. **Since the built-in predicates in $\beta$ do not imply equalities $=$,** we know $V_2 \leq V_1$ cannot appear in $\beta_1$. Then we can always assigne two different constants $c_1 < c_2$ to $V_1$ and $V_2$, respectively, such that $V_1 \leq V_2$ in $\beta_1$ is still true under this new instantiation. In addition, based on the analysis above, under $\rho$, each pair of variable $(A_i, B_i)$ that make $\rho(\gamma_i(\beta_1))$ false must have $\rho(A_i) \neq \rho(B_i)$. For those variables in $X$ that are instantiated to the same constant under $\rho$, we can change the instantiation to $\rho'$, such that $\rho'$ assigns different constants to different variables in $X$. Furtheremore $\rho'(\beta_2)$ holds, and

$$\rho'(\gamma_1(\beta_1)) \vee \ldots \vee \rho'(\gamma_k(\beta_1))$$

Step 2: In this step we want to extend the instantiation $\rho'$ (of those variables $X$) to an instantiation $\rho''$ of the set $Y$ of *all* the variables in $Q_2$. Different variables in $Y$ are instantiated by $\rho''$ to different constants. In addition, for the database $D = \rho''(Q_{20})$, we show $Q_2(D) \not\subseteq Q_1(D)$, contradicting the fact that $Q_2 \subseteq Q_1$. The reason we need to do the extension is that some variables in $Q_2$ may not be in

[2] More accurately, $V_1$ and $V_2$ appear in a built-in predicate in the closure of $\beta_1$.

$X$. The extension is possible since **the built-in predicates in $\beta$ do not imply equalities $=$.** Thus for those variables in $Y - X$, we can just assign unique constants to them. For the database $D = \rho''(Q_{20})$, the instantiation $\rho''$ turns the head $H_2$ of $Q_2$ to a tuple $\rho''(H_2)$ in the answer to $Q_2(D)$, since $\rho''(\beta_2)$ still holds. On the other hand, $Q_1$ cannot produce this tuple in its answer $Q_1(D)$, since (1) $\rho''$ assigns different constants to different variables in $Y$; (2) for all those containment mappings $\gamma_1, \ldots, \gamma_k$ from $Q_{10}$ to $Q_{20}$, each $\gamma_i$ of them has $\rho''(\gamma_i(\beta_1))$ to be false.

"If": Let $D$ be a database and $t$ be a tuple in the answers of $Q_2$. We will prove that $t$ is also in the answers of $Q_1$. Let $o$ be the ordering of the elements of $D$ that generates $t$. There is a mapping $\nu$ from $Q_{20}$ to $D$ such that $o \Rightarrow \nu(\beta_2)$. We know that:

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

Hence,

$$\nu(\beta_2) \Rightarrow \nu\big(\gamma_1(\beta_1) \vee \ldots \vee \gamma_k(\beta_1)\big)$$

Consequently

$$o \Rightarrow \nu\big(\gamma_1(\beta_1) \vee \ldots \vee \gamma_k(\beta_1)\big)$$

Thus there is an $i$, such that: $o \Rightarrow \nu\big(\gamma_i(\beta_1)\big)$. So the mapping $\gamma_i$ followed by the mapping $\nu$ computes $t$ as and answer of $Q_1$. ∎

### 3.2 Case 2

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

**Theorem 3.2** *Assume the comparisons in $Q_1$ and $Q_2$ do not imply equalities. If there is a containment mapping $\mu$ from $Q'_{1,0}$ to $Q'_{2,0}$, such that $\beta'_2 \Rightarrow \mu(\beta'_1)$, then there must be a containment mapping $\gamma$ from $Q_{1,0}$ to $Q_{2,0}$, such that $\beta_2 \Rightarrow \gamma(\beta_1)$.* □

To prove the theorem, we *denormalize* query $Q'_1$ and $Q'_2$ back to $Q_1$ and $Q_2$ respectively. The denormalization is the opposite process of the normalization. It has two steps.

- Case (a): For each added comparison $X = X_i$ introduced for a shared variable $X$ in an ordinary subgoal, we replace all the $X_i$'s with $X$ and remove this comparison.
- Case (b): For each added comparison $Z = c$ introduced for a constant $c$ in an ordinary subgoal, we change $Z$ back to $c$ and remove this comparison.

We show that because of the implication $\beta'_2 \Rightarrow \mu(\beta'_1)$, we can modify the containment mapping $\mu$ to a new containment mapping $\gamma$ from $Q_{1,0}$ to $Q_{2,0}$, such that $\beta_2 \Rightarrow \gamma(\beta_1)$.

Recall that the containment mapping $\mu$ must map a constant to the same constant, and map a variable to either a variable or a constant. So in general it may be possible that $\mu$ will not yield a containment mapping $Q_{1,0}$ to $Q_{2,0}$ after the denormalization, as shown in Example 2.1. We consider the two cases in the denormalization.

Case (a): A variable $X_i$ was introduced to replace one occurrence of a variable $X$. After the denormalization of both queries, we want to show that the two corresponding variables of $Y_1 = \mu(X)$ and $Y_2 = \mu(X_i)$ should be replaced by the same argument (variable or constant) $A$.

$$\begin{array}{ll} Q'_1: & \ldots X \ldots X_i \ldots, X = X_i, \ldots \\ & \quad\downarrow \quad \downarrow \mu \\ Q'_2: & \ldots Y_1 \ldots Y_2 \ldots \end{array}$$

In this case the implication $\beta'_2 \Rightarrow \mu(\beta'_1)$ is $\beta'_2 \Rightarrow \mu(\ldots X = X_i \ldots)$. Thus we have:

$$\beta'_2 \Rightarrow \ldots Y_1 = Y_2 \ldots$$

In other words, in the (minimal inequality) graph $\mathcal{MG}(\beta'_2)$ of $\beta'_2$, variables $Y_1$ and $Y_2$ are at the same node. Since $\beta'_2$ was obtained from $\beta_2$ by adding equalities, and $\beta_2$ itself does not imply equalities, by Propositions 2.1 and 2.2, $Y_1$ and $Y_2$ must be replaced with the same argument (variable or constant) $A$ after the denormalization. Thus we can map the two occurrences of variable $X$ in $Q_1$ to the same argument $A$ in $Q_2$ (after denormalizing both $Q'_1$ and $Q'_2$).

Case (b): A variable $Z$ was introduced to replace one occurrence of a constant $c$, and $\mu(Z) = Y$. The argument is similar as in case (a).

For any other variable $X$ in $Q_1$, we define $\gamma(X) = \mu(X)$.

In summary, we can denormalize $Q'_1$ and $Q'_2$, and modify the containment mapping $\mu$ to another containment mapping $\gamma$ from $Q_{1,0}$ to $Q_{2,0}$, such that $\beta_2 \Rightarrow \gamma(\beta_1)$.

## 4 Classes of Queries with Homomorphism Property

In this section, we are looking 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. That is, 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 4.2, and discuss how to prove it. In subsection 4.3 we extend Theorem 4.2 by considering the general case, where the containing query may use any semi-interval subgoals and point inequality subgoals. In Section 4.4, we argue (using examples) that if the containing query has more general inequalities than SI, then it is unlikely to find interesting cases that have the homomorphism property. In the Appendix, we include many examples to show that the conditions in the main theorems are tight.

Section 4.5 gives an algorithm for checking whether these conditions are met.

### 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 2.1. We shall give some terminology (for ease of reference) and some basic technical observations. The left-hand side (lhs) is a conjunction of arithmetic comparisons (in Example 2.2, 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 $k$ arithmetic comparisons. For instance, in Example 2.2, the rhs is: $(A < 4 \wedge B = 4) \vee (C < 4 \wedge D = 4)$. There are two disjuncts: one is the $A < 4 \wedge B = 4$ and the other is the $C < 4 \wedge D = 4$. Each disjunct is the conjunction of two arithmetic comparisons. In the following, given an integer $i$, we shall call **containment implication** any implication of this form: a) the lhs is a conjunction of arithmetic comparisons, and b) 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**.[3] In Example 2.2, 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 2.2 can be equivalently written as

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

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 if it is true, then

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

is also true. (PLEASE CHECK THIS DEFINITION. CAN WE SAY THEY ARE "LOGICALLY EQUIVALENT"?) 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 4.1** *Let* $Implication1 : \beta \Rightarrow (\beta_1 \vee \beta_2 \vee \ldots \vee \beta_k)$, *where each* $\beta_i$ *is a conjunction of conjuncts* $e_{i1} \wedge e_{i2} \ldots \wedge e_{in}$. *Also assume the following property: if* $\beta \Rightarrow (d1 \vee d2 \vee \ldots \vee dk)$ *holds, where* $dj$ *is a conjunct in* $\beta_j$, *then for some* $l \in (1, \ldots, k)$, $\beta \Rightarrow dl$ *holds. Then for some* $m \in (1, \ldots, k)$, *we have* $\beta \Rightarrow \beta_m$. □

**Proof:** By contradiction. Assume $\beta$ does not imply any of the disjuncts $\beta_m$. Then, for each $\beta_m$, there must exist one $e_m = e_{mr}$, such that $\beta \Rightarrow e_{mr}$, $r \in (1, \ldots, n)$, does not hold - otherwise, if $\beta \Rightarrow e_{mr}$ holds for all $r$, $\beta \Rightarrow \beta_m$ will hold. Consider the disjunction of all such $e_{mr}$'s. Since Implication 1 holds, $Implication2 : \beta \Rightarrow (e_1 \vee e_2 \vee e_k)$ holds. By the statement of the lemma we are given that if Implication 2 holds, there exist one $e_p$ such that $\beta \Rightarrow e_p$, which contradicts our assumption. ∎

In this section we will give conditions that can guarantee this direct implications in containment test.

**Corollary 4.1** *Consider the normalized queries* $Q'_1$ *and* $Q'_2$ *in Theorem 2.1. Suppose all partial containment implications are direct. Then there is a mapping* $\mu_i$ *from* $Q'_{1,0}$ *to* $Q'_{2,0}$ *such that* $\beta'_2 \Rightarrow \mu_i(\beta'_1)$. □

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

**Lemma 4.2** *Let $Q_1 : Q_{10}, \beta_1$ and $Q_2 : Q_{20}, \beta_2$ be conjuctive queries whose cores are $Q_{10}$ and $Q_{20}$ respectively and the ACs are $\beta_1$ and $\beta_2$ respectively. Let $Q_1$ be normalized to $Q'_1 : Q'_{10}, \beta'_1$ and $Q'_2 : Q'_{20}, \beta'_2$, where $Q'_{10}(Q'_{20})$ is the core of $Q'_1(Q'_2)$ and $\beta_1(\beta_2)$ is the AC of $Q'_1(Q'_2)$ respectively. If $\beta'_2 \Rightarrow \mu(\beta'_1)$ where $\mu$ is a containment mapping from $Q'_1$ to $Q'_2$, there exists a containment mapping $\nu$ from $Q_1$ to $Q_2$ such that $\beta_2 \Rightarrow \nu(\beta_1)$.* □

**Proof:** First we construct a containment mapping $\mu'$ from $Q_1$ to $Q'_2$ using $\mu$ by setting $\mu'(X) = \mu(X)$. $\beta'_2 \Rightarrow \mu(\beta'_1)$. By construction, $\beta'_1$ contains all the ACs in $\beta_1$. That is, $\beta'_1 \Rightarrow \beta_1$. Or, $\mu(\beta'_1) \Rightarrow \mu(\beta_1)$. Therefore, $\beta'_2 \Rightarrow \mu(\beta_1)$ [1]. Consider a mapping $\mu'$ from the core subgoals of $Q'_2$ to those of $Q_2$. If we introduced a new variable $Y$ in $Q'_2$ for a shared variable $X$ in $Q_2$, $\mu'(Y) = X$. For a variable $Y$ that is in $Q'_2$ and also in $Q_2$, $\mu'(Y) = Y$. If we introduced a new variable $Y$ in $Q'_2$ for a constant $c$ in $Q_2$, set $\mu'(Y) = c$. Since a variable occurs only once in the core of $Q'_2$, $\mu'$ is a containment mapping from $Q'_2$ to $Q_2$.

We show below that $\beta_2 \Rightarrow \mu'(\beta'_2)$ holds. We consider the three types of AC's in $\beta'_2$ and show that $\beta_2$ implies each (under the mapping $\mu'$):

1. AC's in $\beta'_2$ that were also in $\beta_2$ hold by direct implication, since the variables that occur in these AC's occur in both $Q_2$ and $Q'_2$ and $\mu$ maps these variables to themselves.

2. AC's in $\beta'_2$ due to shared variables in $\beta_2$: For an occurrence of a shared variable $X$ in $Q_2$, we introduced a new variable $Y$ in $Q'_2$ and added $X = Y$ in $\beta'_2$. Now, $\mu'$ maps $Y$ to $X$. Therefore, such AC's in $\beta'_2$ are reduced to tautologies $X = X$ in $\mu(\beta'_2)$ and are trivially implied by $\beta_2$.

3. AC's in $\beta'_2$ due to constants in $Q_2$. We introduced a new variable $Y$ in $Q'_2$ and $Y = c$ in $\beta'_2$. Again, $\mu'(Y) = c$ and $c = c$ is trivially implied by $\beta_2$.

Hence $\beta_2 \Rightarrow \mu'(\beta'_2)$ holds [2].

Combining implications [1] and [2], we have $\beta_2 \Rightarrow \mu'(\mu(\beta_1))$. $\mu$ is a containment mapping from the variables in $Q_1$ to $Q'_2$. $\mu'$ is a containment mapping from $Q'_2$ to $Q_2$. Therefore $\mu'.\mu$ is a containment mapping from $Q_1$ to $Q_2$. ■

Per Lemmas 4.1 and 4.2, if we can show that the only form of implication is direct implication and rule out coupling implication, the homomorphism property will hold.

**Theorem 4.1** *$Q_1 : Q_{10}, \beta_1$ is a CQAC such that $Q_{10}$ is the core of $Q_1$, and $\beta_1$ is the arithmetic comparisons of $Q_1$. Similarly, $Q_2 : Q_{20}, \beta_2$ is a CQAC with a core $Q_{20}$ and arithmetic comparisons $\beta_2$. Let $Q'_1 : Q'_{10}, \beta'_1$ and $Q_2 : Q'_{20}, \beta'_2$ be the normalized forms of $Q_1$ and $Q_2$ respectively, with cores $Q'_{10}$ and $Q'_{20}$ and arithmetic comparisons $\beta'_1$ and $\beta'_2$ respectively. Assume, for any set of conjuncts $(e1, e2, \ldots, ek)$, where $e_i$ is a conjunct in $\mu_i(\beta'_1)$, and $\mu_1, \mu_2, \ldots, \mu_k$ are all the containment mappings from $Q'_{10}$ to $Q'_{20}$, $\beta'_2 \Rightarrow el$ holds for $l \in (1, \ldots, k)$. Then, $Q_2 \sqsubseteq inQ_1$ iff there exists a homomorphism (from $Q_{10}$ to $Q_{20}$) $\mu$, such that $\beta_2 \Rightarrow \mu(\beta_1)$.* □

**Proof:** If: follows from Theorem 2.1.

Only If: If $Q_2 \sqsubseteq Q_1$, then by Theorem 2.1 $Implication1 : \beta'_2 \Rightarrow \mu_1(\beta'_1) \vee \mu_2(\beta'_1) \vee \ldots \mu_k(\beta'_1)$, where $\mu_i, i \in 1 \ldots k$ are all the containment mappings from $Q'_1$ to $Q'_2$.

First, we argue that in the normalized queries, there exists one CM $\nu$ from $Q'_1$ to $Q'_2$ such that the implication $\beta'_2 \Rightarrow \nu(\beta'_1)$ holds. Then we show that if $\nu$ exists, there exists a CM $\mu$ from $Q_1$ to $Q_2$ such that the implication $\beta_2 \Rightarrow \mu(\beta_1)$ holds.

We rewrite the consequent of Implication 1 as a conjunction of disjunctions, by conjoining all possible combinations $Ci$, obtained by choosing one AC from each $\mu_i(\beta'_1)$ for $i \in (1, \ldots k)$, and obtaining the disjunction of the AC's in a combination $Ci$. each disjunct appears in some $\mu_i(\beta'_1)$. $\beta'_2 \Rightarrow C1 \wedge \mathcal{C}2 \wedge \ldots CN$, where $N$ is the number of inequalities in $\beta'_1$. $\beta'_2$ implies each of these conjuncts $Cj, \forall j, 1 \leq j \leq N$ separately. Consider one such implication $\beta'_2 \Rightarrow C_j$, where $C_j$ is $e_1 \vee e_2 \vee \ldots e_k$ and $e_i \in \mu_i(\beta'_1), i \in (1, \ldots k)$. We have $\beta'_2 \Rightarrow e_1 \vee e_2 \vee \ldots e_k$. By the hypothesis of the theorem, $\beta'_2 \Rightarrow e_l$, for $l \in (1, \ldots, k)$.

Using the result above, and Lemma 4.1, we have $\beta'_2 \Rightarrow \mu_j(\beta'_1)$ where $j \in (1, \ldots, k)$. Again, using the result above, and Lemma 4.2, we show that there exists a containment mapping $\mu$ from $Q_1$ to $Q_2$ such that $\beta_2 \Rightarrow \mu(\beta_1)$. ■

As shown above, if Implication 1 does not result in coupling implications, the homomorphism property holds and we can design efficient algorithms to check query containment. By exhaustively listing the different forms of coupling implications, and finding sufficient conditions to avoid them, we can identify sufficient conditions under which the homomorphism property holds. We enumerate the different forms of coupling in the appendix and provide the sufficient conditions to avoid coupling in the next section.

## 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 that describes the conditions for the homomorphism property to hold in this case.

**Theorem 4.2** *Let $Q_1$ be a conjunctive query with left semi-interval arithmetic comparison subgoals and $Q_2$ a conjunctive query with any arithmetic comparison subgoals. If $Q_1$ and $Q_2$ 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 the 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 $AC(Q_2)$.*

□

See the Appendix for an extended sketch of the proof of Theorem 4.2. If confusion does not arise we will refer to these conditions simply by (i), (ii), and (iii) instead of (i)-lsi, (ii)-lsi, and (iii)-lsi. It is straightforward to construct corollaries of Theorem 4.2 that use simpler conditions. The following corollary is an example.

**Corollary 4.2** *Let $Q_1$ be a conjunctive query with left semi-interval arithmetic comparison subgoals and $Q_2$ a conjunctive query with any arithmetic comparison subgoals. If the comparisons in $Q_1$ do not share a constant with the closue of the arithmetic comparisons of $Q_2$, then the homomorphism property holds.* □

The results in Theorem 4.2 can be symmetrically stated for RSI queries as containing queries. The symmetrical conditions of Theorem 4.2 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 4.2 to treat both LSI and RSI subgoals occurring in the same containing query. We further extend it to include point inequalities, i.e., of the form $X \neq c$ where $X$ is a variable and $c$ is a constant.

### 4.3.1 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 4.2 is extended to the following theorem, which is the second main result of this section.

**Theorem 4.3** *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 $Q_1$ and $Q_2$ satisfy all the following conditions, then the homomorphism property holds.*

- *Conditions (i)-lsi, (ii)-lsi, (iii)-lsi, (i)-rsi, (ii)-rsi, and (iii)-rsi.*

- *Condition (iv): (a) Any constant in an RSI subgoal of $Q_1$ is strictly greater than any constant in an LSI subgoal of $Q_1$, or (b) (i) $Q_1$ contains only open subgoals (open-LSI or open-RSI), and (ii) any constant in an RSI subgoal of $Q_1$ is greater than or equal to all constants in LSI-subgoals of $Q_1$, and (iii) there do not exist an open-LSI subgoal in $Q_1$, an open-RSI subgoal in $Q_1$, and $Core(Q_1)$ that share a constant.*

□

### 4.3.2 PI Queries for $Q_1$

If the containing query $Q_1$ has point inequalities, three more forms of coupling implications can occur. Thus Theorem 4.3 is further extended to Theorem 4.4, which is the third main result of this section.

**Theorem 4.4** *Let $Q_1$ be a conjunctive query with left semi-interval and right semi-interval and point inequality arithmetic comparison subgoals and $Q_2$ a conjunctive query with SI arithmetic comparison subgoals. 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, and (iii)-rsi.*

- *Condition (iv).*

- *Condition (v): Either $Q_1$ has no repeated variables, or it does not have point inequalities.*

- *Condition (vi): $core(Q_1)$ and Point-Inequalities$(Q_1)$ do not share a constant.*

- *Condition (vii)-lsi: Closed-LSI$(Q_1)$, Point-Inequality$(Q_1)$ do not share constant.*

- *Condition (vii)-rsi: Closed-RSI$(Q_1)$, Point-Inequality$(Q_1)$ do not share a constant.*

□

### 4.4 Beyond Semi-Interval Queries for $Q_1$

In this subsection, we try to see how far the kind of results we obtained so far can be pushed to hold on more general queries. We give some examples that indicate our results have already captured subtle cases where the homomorphism property holds and 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., [11]). Here is one example. There are two more in the Appendix.

Coupling 1: Couplings that occur due to the following implication:

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

indicates 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.

In summary, in this subsection, we have provided formal conditions that, for queries with SI-PI inequalities, one mapping is sufficient to test query containment *unless* the queries use shared variables *and* same constant from the same domain in "many" subgoals. We provide examples (in the Appendix) of all those exceptional cases where one mapping is not enough. The queries in those examples are contrived in order to enforce a "hard" containment test. Therefore, we believe that, in most practical cases the containment test is "easy". The formal phrasing of this informal statement are Theorems 4.2, 4.3, and 4.4.

### 4.5 A Testing Algorithm

We summarize the results in this section in an algorithm shown in Figure 4. Given two CQAC queries $Q_1$ and $Q_2$, the algorithm tests if the homomorphism property holds in checking $Q_2 \sqsubseteq Q_1$. It considers LSI, RSI, SI, and PI comparisons in the queries sequentially. Its different steps are based on the results in Subsections 4.2 and 4.3. It is possible for queries not to satisfy these conditions, while it is still possible for the homomorphism property to hold. For instance, it could happen if they do not have self-joins, or 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.

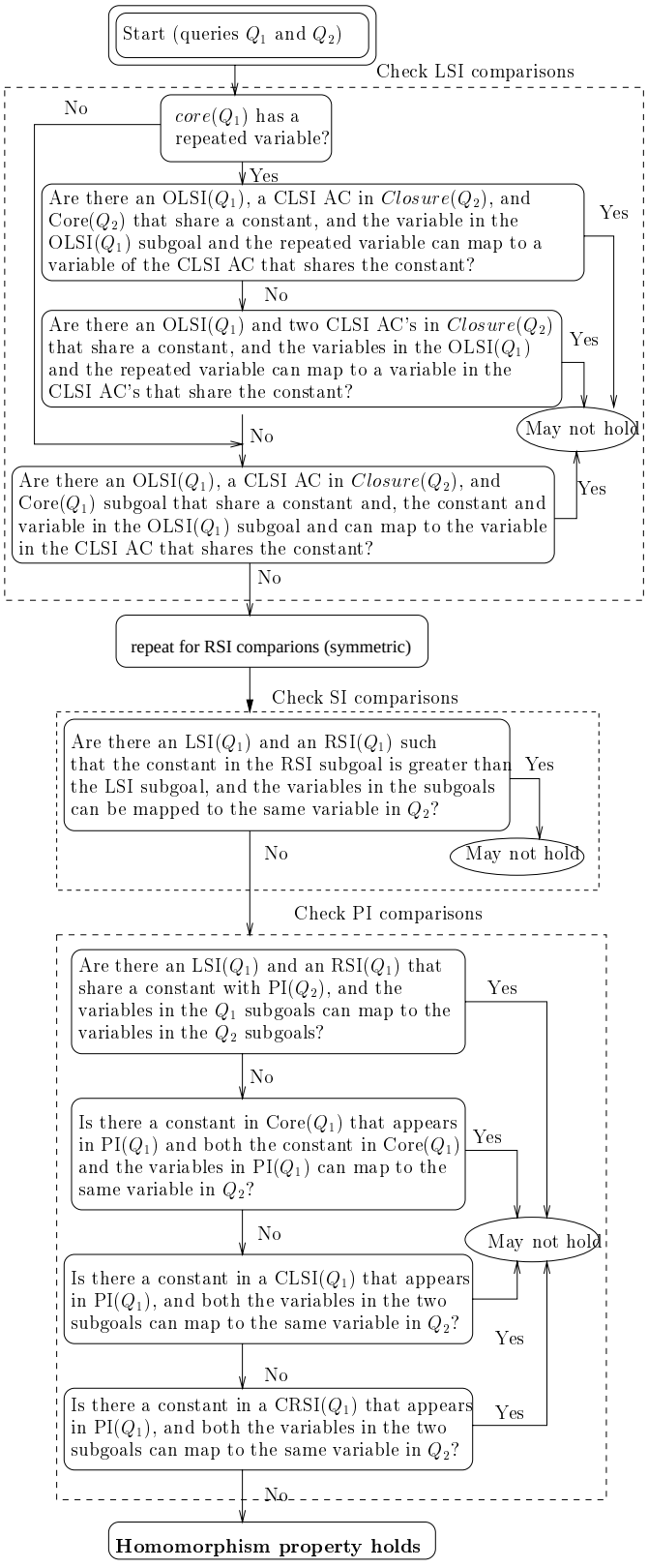


**Figure 4. An algorithm for checking homomorphism property in testing $Q_2 \sqsubseteq Q_1$.**

## 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 homomorphism. It is possible that a query does not satisfy these conditions, and the results become inapplicable. For instance, often a query may have both $<$ and $\geq$ comparisons, not satisfying the conditions in Theorem 3.1. In this section we study how to relax the conditions in the theorems 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." It is safe to 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 the attribute of $X$. As an 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 $\phi$:

$$\begin{array}{rl} & year > 2000 \wedge price \leq \$5,000 \\ \Rightarrow & year > 1998 \wedge price \leq \$6,000 \end{array}$$

we do not need to consider implication between constants or variables in different domains, such as "1998" and "$\$6,000$," and "year" and "price." As a consequence, this implication can be projected to the following implications in two domains:

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

We can show that $\phi$ is true if and only if both $\phi_y$ and $\phi_p$ are true. In this section we first formalize this domain idea, then show how to partition an implication into implications of different domains. Finally we relax the conditions of the theorems in the previous sections to conditions in different domains.

### 5.1 Domains of Relation Attributes and Query Arguments

We assume each attribute $A_i$ in a relation $R(A_1, \ldots, A_k)$ has a domain $Dom(R.A_i)$. Different attributes of the same or different relations could share the same domain. For instance, consider the following two tables.

```
house(seller, street, city, price)
crimerate(city, rate)
```

Relation `house` has housing information such as the seller, the street, the city, and the price of each house. The relation `crimerate` has information about the crime rate of each city. The following table shows the domains of different attributes in these relations. Notice that attributes `house.city` and `crimerate.city` share the same domain: $D_3$ = {city names}.

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

Given a query $Q$, each variable and constant in $Q$ also has its domain. We want to compute these domains automatically, so that the user does not need to specify them explicitly. The argument domains are calculated as follows.

- 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 a variable $X$ and a constant $c$, we set $Dom(c) = Dom(X)$. Notice that two constants could have the same value but they are from different domains, and thus they are treated as different constants. For instance, in two conditions $carYear = 2000$ and $carPrice = \$2000$, constants "2000" and "\$2000" are treated as different constants, since they are from different domains.

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. The rationale of these assumptions is that, realistic queries do not have conditions that compare two arguments from different domains.

Consider the following two queries on the relations above.

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

The computed domains of the variables and constants are shown in the following table.

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

It is easy to see that the domain information as defined in this section can be obtained in polynomial time.

**Proposition 5.1** *The domain of each variable and each constant can be found in polynomial time.* □

## 5.2 Partitioning Implication into Domain Implications

According to Theorem 2.1, 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 $\mu_i$ 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 5.1** (**Implication Projection**) Consider the following implication $\phi$ in Theorem 2.1:

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

For a domain $D$ of the arguments in $\phi$, the *projection of $\phi$ in $D$*, denoted $\phi_D$, 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 the comparisons of $\beta'_2$ in domain $D$. Similarly, $\beta'_{1,D}$ includes all the comparisons of $\beta'_1$ in domain $D$. □

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

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

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

**Theorem 5.1** *Let $D_1, \ldots, D_k$ be the domains of the arguments in the implication $\phi$. Then $\phi$ is true if and only if all the projected implications $\phi_{D_i}, \ldots, \phi_{D_k}$ are true.* □

**Proof:** (Sketch) The implication $\phi$ can be represented as an inequality graph [10]. Based on the definition of argument domains, the vertices of arguments in the same domain form a subgraph, which should be disconnected from any other subgraph of vertices of arguments in a different domain. Thus "coupling" between vertices of different domains can never happen in the implication. ■

In the example above, by Theorem 5.1, $\pi$ is true if and only if $\pi_{D_4}$ and $\pi_{D_5}$ are true. Since the latter two are true, $\pi$ is true. Thus $P_2 \sqsubseteq P_1$. In general, we can test the implication in Theorem 2.1 by testing the implications in different domains, which are much cheaper than testing the whole implication.

## 5.3 Relaxing Conditions in the Theorems

By using Theorem 5.1 we can significantly relax the conditions of the theorems in the previous sections. The conditions of all the results in the previous sections can be relaxed to those arguments in the same domain. As an example, if a theorem requires the containing query to be LSI, then we can change this requirement to requiring all AC subgoals *in the same domain* to be LSI. Notice that it is possible that one domain is LSI while the other domain is RSI.

The following theorem is an example of how to relax condition (ii)-lsi in Theorem 4.2.

**Theorem 5.2** *Let $Q_1$ be a conjunctive query with left semi-interval arithmetic comparisons and $Q_2$ a conjunctive query with any arithmetic comparisons. If $Q_1$ and $Q_2$ 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 $c$* **from the same domain***:* [4] *An open-LSI subgoal in $AC(Q_1)$, a closed-LSI subgoal in $AC(Q_2)$, and a subgoal in $core(Q_1)$.*
- Condition (ii)-lsi*: For every shared variable $X$ in $core(Q_1)$, and every constant $c$* **in the same domain** *as $X$, there do not exist subgoals as follows which all share $c$: An open-LSI subgoal in $AC(Q_1)$, a closed-LSI subgoal in $AC(Q_2)$, and a subgoal in $core(Q_2)$.*

[4] Notice, e.g., that "$4" has a different domain from that of "4 years," hence they do not count as same in this condition.

- Condition (iii)-lsi*: For every shared variable $X$ in $core(Q_1)$, and every constant $c$* **in the same domain** *as $X$, there do not exist subgoals as follows which all share $c$: An open-LSI subgoal in $AC(Q_1)$ and two closed-LSI subgoals in $AC(Q_2)$.*

□

## 6 Experiments

In this section we evaluate the results on real queries, and check whether the conditions in previous sections are satisfied by these queries. We checked queries in many introductory database courses available on the Web, data-mining queries from Microsoft Research, and the TPC-H benchmark queries [18]. We have observed that, for certain applications (e.g., the data-mining queries), it is not unusual for queries to not have self-joins, thus the homomorphism property obviously holds. In addition, among the queries that use only semi-interval (SI) and point inequality (PI) comparisons, the majority has the homomorphism property.

For a more detailed discussion, we focus on our evaluation results on the TPC-H benchmark queries [18], which represent typical queries in a wide range of decision-support applications. From what we know, our results are the first formal proof that containment is easy for those queries. In our experiments we viewed these benchmark queries as containing queries in the containment test. That is, we supposed that we wanted to check containment of a benchmark query against any CQAC query. We removed the aggregations from the queries, i.e., for each aggregate query, we considered a CQAC query with the same subgoals and the same arithmetic restrictions in the `WHERE` clause. We ran our algorithm described in Section 4.5. We also used domain information on top of the algorithm in the flowchart, applying the domain-partitioning results in Section 5. If the algorithm did not give us a definite answer as to whether the homomorphism property holds against any query, then we restricted the contained-query pool too, and gave conditions that the contained query must satisfy in order for the homomorphism property to hold.

It is also noticed that in order to be able to decide conclusively, we also used the "mapping check" of the algorithm. That is, whenever the conditions were not met, we showed that an appropriate mapping that would enable a coupling implication (and hence fail the homomorphism property) would not occur either because of domain information or because the query would be very unnatural and contrived. To justify the latter remark, we argued that often if a constant is shared by different subgoals or queries, then, we may change one of its occurrences by a small fraction without affecting the intended meaning of the query (that would be the case with a date). By doing this however, we meet the shared variable condition of our algorithm.

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 (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 this variable.

3. As a consequence of result (2), 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 the Appendix. 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]';
```

Let us view the above query $Q_3$ as a containing query to be checked against any contained query that is a conjunctive query with semi-interval comparisons. We shall apply Theorem 4.3. Because of lack of space, we do not state its relaxation, which essentially adds to the conditions of Theorem 5.2 condition (iv) that is now relaxed to hold only on inequalities in the same domain. 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, although that information is not explicitly stated in the query. Hence conditions (ii)-lsi, (ii)-rsi, (iii)-lsi, (iii)-rsi are satisfied. Also there are no constants *in the same domain* that are shared among subgoals in $core(Q_3)$ and the comparison, because the constant in the comparisons is in a different domain than the constant in $core(Q_3)$. Hence conditions (i)-lsi and (i)-rsi are satisfied.

As for condition (iv), it may not be satisfied but the simplest scenario on 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 we know that in order to test containment of any natural SI query in $Q_3$, we will need only one containment mapping. Notice that without using the domain information, we could not make in 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 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 experiment using real queries showed that many of these queries pass this test, so they do have the homomorphism property in a containment mapping, making it possible to use more efficient algorithm for the test.

# References


[1] F. Afrati, C. Li, and P. Mitra. Answering queries using views with arithmetic comparisons. In *PODS*, 2002.

[2] F. Afrati, C. Li, and J. D. Ullman. Generating efficient plans using views. In *SIGMOD*, pages 319–330, 2001.

[3] A. Chandra, H. Lewis, and J. Makowsky. Embedded implication dependencies and their inference problem. In *STOC*, pages 342–354, 1981.

[4] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. *STOC*, pages 77–90, 1977.

[5] S. Chaudhuri, R. Krishnamurthy, S. Potamianos, and K. Shim. Optimizing queries with materialized views. In *ICDE*, pages 190–200, 1995.

[6] S. Chaudhuri and M. Y. Vardi. On the equivalence of recursive and nonrecursive datalog programs. In *PODS*, pages 55–66, 1992.

[7] S. S. Cosmadakis and P. Kanellakis. Parallel evaluation of recursive queries. In *PODS*, pages 280–293, 1986.

[8] A. Gupta, Y. Sagiv, J. D. Ullman, and J. Widom. Constraint checking with partial information. In *PODS*, pages 45–55, 1994.

[9] A. Halevy. Answering queries using views: A survey. In *Very Large Database Journal*, 2001.

[10] A. Klug. On conjunctive queries containing inequalities. *Journal of the ACM*, 35(1):146–160, January 1988.

[11] P. G. Kolaitis, D. L. Martin, and M. N. Thakur. On the complexity of the containment problem for conjunctive queries with built-in predicates. In *PODS*, pages 197–204, 1998.

[12] A. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *PODS*, pages 95–104, 1995.

[13] R. Pottinger and A. Levy. A scalable algorithm for answering queries using views. In *Proc. of VLDB*, 2000.

[14] X. Qian. Query folding. In *ICDE*, pages 48–55, 1996.

[15] Y. Saraiya. Subtree elimination algorithms in deductive databases. *Ph.D. Thesis, Computer Science Dept., Stanford Univ.*, 1991.

[16] O. Shmueli. Equivalence of datalog queries is undecidable. *Journal of Logic Programming*, 15(3):231–241, 1993.

[17] D. Theodoratos and T. Sellis. Data warehouse configuration. In *Proc. of VLDB*, 1997.

[18] TPC-H. `http://www.tpc.org/tpch/`.

[19] J. D. Ullman. Information integration using logical views. In *ICDT*, pages 19–40, 1997.

[20] R. van der Meyden. The complexity of querying indefinite data about linearly ordered domains. In *PODS*, 1992.

[21] J. Wang, M. Maher, and R. Toper. Rewriting general conjunctive queries using views. In X. Zhou, editor, *Thirteenth Australasian Database Conference (ADC2002)*, Melbourne, Australia, 2002. ACS.


# A Appendix

## A.1 Proof of Theorem 4.2

The proof of Theorem 4.2 is based on two observations on the containment implication of Theorem 2.1: (a) If a coupling partial containment implication occurs, then one of the three conditions is shown to be dissatisfied. Consequently if the conditions are satisfied then we get only direct partial containment implications. (b) If we have only direct partial containment implications, then the homomorphism property holds. We settle (a) as follows: Lemma A.1 proves that only three forms of coupling partial containment implications may occur. We observe that if the conditions (i)–(iii) of Theorem 4.2 are satisfied, then the partial containment implications cannot have any of these three forms. Consequently, if the conditions of Theorem 4.2 hold, there are only direct partial containment implications.

**Lemma A.1** *Consider a partial containment implication with rhs a disjunction of comparisons of either of the following forms* $X \leq c, X < c, X = c, X = Y$. *Suppose the rhs is minimal with respect to the satisfaction of the implication, i.e., if we delete any of the disjuncts, the implication is not satisfied. Then either the rhs has exactly one AC (i.e., it is a direct implication) or the implication is one of the following three cases (up to renaming of variables and constants and up to adding any number of additional conjuncts in the lhs):*

$$(i)\ \ (X \leq c) \Rightarrow ((X < c) \vee (X = c))$$
$$(ii)\ \ ((X \leq c) \wedge (Y = c)) \Rightarrow ((X < c) \vee (X = Y))$$
$$(iii)\ \ (X \leq c) \wedge (Y \leq c) \Rightarrow (X < c) \vee (Y < c) \vee (X = Y)$$

□

Now using the above lemma, it is not hard to prove that the conditions (i), (ii), and (iii) of Theorem 4.2 do not allow coupling implications to happen. Finally, the proof of Theorem 4.2 is a direct consequence of the above results and of Corollary 4.1 and Theorem 3.2.

## A.2 Enumeration of the Different Forms of Coupling

In this subsection, we identify conditions to avoid all possible forms of coupling. We use the term *c-conjunct* to refer to a conjunct that appears in the consequent of an implication. Similarly,

we use the term *a-conjunct* to refer to a conjunct that appears in the antecedent of an implication.

Note that for coupling, we need at least 2 c-conjuncts in the consequent. Below, we list all possible combinations of two c-conjuncts e1 and e2 and for each combination, derive the type of coupling that can occur.

Forms of the c-conjuncts:

1. $X < c$
2. $X > c$
3. $X \leq c$
4. $X \geq c$
5. $X \neq c$
6. $X = c$
7. $X = Y$

A negated c-conjunct can be one of the following forms:

1. $X \geq c$
2. $X \leq c$
3. $X > c$
4. $X > c$
5. $X = c$
6. $X \neq c$
7. $X \neq Y$

Forms of a-conjuncts:

1. $X < c$
2. $X > c$
3. $X \geq c$
4. $X \leq c$
5. $X \neq c$
6. $X = c$
7. $X = Y$
8. $X \geq Y$
9. $X > Y$
10. $X \neq Y$

We now enumerate all possible cases of coupling implications and check to see if the conditions identified in Theorems 4.1, 4.2, and 4.3 avoid them. Consider the coupling implication $\beta \Rightarrow (e_1 \vee e_2)$. The implication can be rewritten as $(\neg\beta) \vee e_1 \vee e_2$. Alternatively, it is equivalent to $\neg(\beta \wedge (\neg e_1) \wedge (\neg e_2))$. If the last statement is true, the statement $(\beta \wedge (\neg e_1) \wedge (\neg e_2))$ is a contradiction. We now identify all possible forms of a-conjuncts and e-conjuncts that produce minimal contradiction. By the term *minimal contradiction* we refer to a conjunction of arithmetic comparisons that is a contradiction such that if any of the arithmetic comparisons is deleted from the conjunction it is no longer a contradiction.

In the sequel, $e1$, $e2$ are c-conjuncts. Minimal contradictions can be of the following forms:

1. $e1 = (X < c), \neg(e1) = (X >= c)$, the rest of the conjuncts in the minimal contradiction need to imply $(X < c)$.

   (a) $e2 = (X < c)$. Since $e1$ and $e2$ are the same one of them can be deleted and the contradiction still holds. Therefore, the set of conjuncts producing the contradiction is not minimal - contrary to our assumptions. Thus, this case can not occur.

   (b) $e2 = (X > c)$ is avoided due to Theorem 4.3 Condition (iv).

   (c) $e2 = (X \leq c)$ renders $e1$ redundant. Thus, this case cannot occur.

   (d) $e2 = (X \geq c)$. Theorem 4.3 Condition (iv) rules out this case.

   (e) $e2 = (X \neq c)$ renders $e1$ redundant. Thus, this case cannot occur.

   (f) $e2 = (X = c), \neg(e2) = (X \neq c), (X \neq c) \wedge D \Rightarrow (X < c), D \Rightarrow (X \leq c)$.

   $e3$ cannot take the forms that have been found unsatisfactory for $e2$ in the previous cases. We check using $e3 = (X = Y), \neg e3 = (X \neq Y), (X \neq Y) \wedge D' \Rightarrow (X \leq c)$. Thus, $D' \Rightarrow (X \leq c)$ must hold and $e3$ is redundant.

   Now we consider the cases using a-conjuncts.

   i. $X < c$ directly implies $e1$ and renders $e2$ redundant.

   ii. $X > c \wedge B \Rightarrow (X \leq c)$ - contradiction.

   iii. $X \leq c$: We have the coupling $(X \leq c) \Rightarrow (X < c) \vee (X = c)$. This form of coupling is ruled out by Theorem 4.2 Condition (i).

   iv. $X \geq c$ renders $e1$ redundant.

   v. $X \neq c$ renders $e2$ redundant.

   vi. $X = c$ - directly implies $e2$ and renders $e1$ redundant.

   vii. $a1 = (X = Y)$: The other useful AC's in the antecedent could be involving the variables i) $X$ and $c$, ii) $Y$ and $c$ and iii) $X$ and $Y$. We have considered all possible forms involving $X$ and $c$ above. Due to the AC $X = Y$, all AC's involving $Y$ and $c$ would also result in an AC involving $X$ and $c$ in the closure of the AC's in the antecedent. As indicated above, we have already eliminated all possible forms of AC's involving $X$ and $c$. Thus, we consider only the AC's involving $X$ and $Y$:

      A. $X \geq Y$ is redundant when we have $a1 = (X = Y)$.

      B. $X \leq Y$ is redundant when we have $a1 = (X = Y)$.

      C. $X < Y$ is a contradiction when we have $a1 = (X = Y)$.

      D. $X > Y$ is a contradiction when we have $a1 = (X = Y)$.

      E. $X \neq Y$ is a contradiction when we have $a1 = (X = Y)$.

   viii. $a1 = (X \leq Y)$: The different cases are:

A. $(Y < c)$ produces $X \leq c$ in the closure of the AC's of the antecedent. Thus, it has been ruled out by Theorem 4.2.

B. $(Y > c)$: In the inequality graph, if $X \leq Y$ is used to produce a path from $X$ to $c$, $Y > c$ cannot be used since in it creates a path from $c$ to $Y$ and not the opposite.

C. $Y \leq c$ produces $X \leq c$ and thus is ruled out by Theorem 4.2.

D. $Y = c$ produces $X \leq c$ and is ruled out by Theorem 4.2.

E. $Y \neq c$ cannot produce a path from $X$ to $c$ while using both $X \leq Y$ and $Y \neq c$ in the inequality graph.

F. $(X \geq Y)$ along with $a1$ creates $X = Y$ - the case has been covered above.

G. $(X < Y)$ renders $a1$ redundant.

H. $(X > Y)$ creates a contradiction.

I. $(X \neq Y)$ is equivalent to $X < Y$ renders $e2$ redundant.

ix. $(X \geq Y)$:

A. $Y > c$ creates a contradiction

B. $Y < c$ and $Y \leq c$: Both $X$ and $c$ have a path from $Y$ in the inequality graph and thus both edges cannot be used to create a path from $X$ to $c$.

C. $Y \geq c$ and $Y = c$ result in $X \geq c$ cannot imply $X < c$.

D. $Y \neq c$ cannot create a path from $X$ to $c$ while using both a-conjuncts.

E. $X < Y$ - renders $(X \geq Y)$ redundant.

F. $Y < X$ - contradiction.

G. $Y = X$ - renders $(X \geq Y)$ redundant.

H. $X \neq Y$ - this case is similar to $X < Y$.

I. $(X < Y)$:

- $Y > c$, $Y \geq c$ or $Y \neq c$ cannot create a path from $X$ to $c$ in the inequlaity graph.
- $Y < c$ or $Y = c$ renders $e2$ redundant.
- $Y \leq c$ results in $X \leq c$ in the closure of the AC's in the antecedent and is ruled out by Theorem 4.2.
- $X > Y$ creates a contradiction.
- $X \neq Y$ is redundant.

J. $(X > Y)$:

- $Y < c$, $Y \leq c$ or $Y \neq c$ cannot create a path from $X$ to $c$ in the inequlaity graph.
- $Y < c$, $Y \geq c$, or $Y = c$ contradicts with $e1$.
- $X \neq Y$ is redundant.

K. $(X \neq Y)$: No AC of the form $Y\ \theta\ c$ can form a path from $X$ to $c$ using $(X \neq Y)$ and $Y\ \theta\ c$.

x. $e2 = (X = Y), \neg(e2) = (X \neq Y), (X \neq Y) \wedge D \Rightarrow (X < c)$.

We now examine the conjunction of $(X \neq Y)$ with another c-conjunct.

A. $\neg e3 = (X < c)$ or $\neg e3 = (Y < c)$ do not jointly imply another AC.

B. $\neg e3 = (X > c)$ or $\neg e3 = (Y > c)$ do not jointly imply another AC.

C. $\neg e3 = (X \leq c)$ or $\neg e3 = (Y \leq c)$ do not jointly imply another AC.

D. $\neg e3 = (X \geq c)$ or $\neg e3 = (Y \geq c)$ do not jointly imply another AC.

E. $e3 = (X = c), \neg e3 = (X \neq c),$ do not jointly imply another AC. Similarly, $e3 = (Y = c), \neg e3 = (Y \neq c)$ does not jointly imply another AC.

F. $e3 = (X \neq c)$ or $e3 = (Y \neq c)$. Since $e2$ has repeated variables, this case is ruled out by Theorem 4.4 Condition (v).

G. $e3 = (X = Z)$. In this case, we introduce a new variable using an AC of the same form as $e2$. We now have to find a D' such that $((X \neq Y) \wedge (X \neq Z) \wedge D') \Rightarrow (X < c)$. This implication is of the same form as that for $e2$ and as seen in the cases above, none of the other cases can be used to prove the implication. In the absence of any implication that involves a variable and the constant $c$, there exists no $D'$ that obeys the constraints of Theorem 4.3 and Theorem 4.4 and still satisfies the above implication.

We now consider a-conjuncts:

A. $X < c$ directly implies $e1$.

B. $X > c$ contradicts $e1$.

C. $a1 = (X \leq c)$:

- $X \geq c$ contradicts $a1$.
- $X = c$ renders $a1$ redundant.
- $X \neq c$ renders $e2$ redundant.
- $Y < c$: If $e2 = (X = Y)$ holds, then $X < c$ holds. Thus, this case renders $e2$ redundant.
- $Y > c$: $a1 = (X \leq c)$ implies either $X < c$ or $X = c$. If $X < c$ does not hold, $e1$ does not hold, $X = c$ holds. Also, since $e1$ does not hold, $e2$ must hold. Thus, $Y = c$ that contradicts $Y > c$.
- $Y = c$ results in the coupling: $((X \leq c) \wedge (Y = c)) \Rightarrow ((X < c) \vee (X = Y))$. and is ruled out by Theorem 4.2.
- $(Y \leq c)$ from a a-conjunct and $(Y \geq c)$ from an c-conjunct results in the following coupling: $((X \leq c) \wedge (Y \leq c)) \Rightarrow ((X < c) \vee (Y < c) \vee (X = Y))$. This form of coupling is ruled out by Theorem 4.2. Though $(Y < c)$ and $(Y =$

$c$) from the a-conjuncts would also imply $(Y \leq c)$, they cause direct implication or have been covered by previous cases. $Y \neq c \wedge F \Rightarrow (Y \leq c)$ when $F \Rightarrow (Y \leq c)$ - not minimal. Using $Y = Z \wedge F \Rightarrow (Y \leq c$ when $F \Rightarrow (Z \leq c)$ - conditions are the same as $Y \leq c$.

- $(Y \geq c)$ comes from more than one conjunct one of which is an c-conjunct. The conjunction of two AC's producing $(Y \geq c)$ can not be produced with $Y < c, Y > c$, $Y \neq c$, $Y = c$ or $Y \leq c$ without breaking the minimal contradiction and $Y = Z, Z \geq c$ - conditions are the same as $Y \geq c$.
- $X \geq c$ - if $X \geq c$ holds and $X \leq c$ holds, then $X = c$ holds, there is no coupling but direct implication of $e2$.
- $X \neq c \wedge B \Rightarrow (X \leq c)$ - true but then $X \leq c, X \neq c$ directly imply $e1$.
- $X = c$ - directly implies $e2$.
- $X = Y \wedge B \Rightarrow (X \leq c)$ - coupling possible if $B \Rightarrow (Y \leq c)$ but condition 2 rules it out. If $B \Rightarrow (Y < c)$ or $(B \Rightarrow (Y = c)$ , then $e1$ or $e2$ are directly implied.
- $(X \leq Y), \wedge B \Rightarrow (X \leq c)$ - coupling possible if $B \Rightarrow (Y \leq c)$. See case (vii).
- $(X < Y), \wedge B \Rightarrow (X \leq c)$ - need $B \Rightarrow (Y \leq c)$. See case (vii).
- $(X \neq Y), \wedge B \Rightarrow (X \leq c)$ - same as (ix).
- $(X \geq Y) \wedge B \Rightarrow (X \leq c)$ - same as (ix).
- $(X < Y) \wedge B \Rightarrow (X \leq c)$ - same as (ix).

xi. $e1 = (X > c)$, need to imply $(X > c)$, symmetric to 1.

xii. $e1 = (X \geq c)$, the rest need to imply $(X \geq c)$. We ignore cases where $e2 = (X < c) or (X > c)$ since those have been covered in cases 1 and 2 above.

A. $e2 = (X \leq c), ((X > c) \wedge D \Rightarrow (X \geq c)$ This form of coupling is prohibited by Theorem 4.3 Condition (iv).

B. $e2 = (X \geq c)$ is redundant since it is the same as $e1$.

C. $e2 = (X \neq c), \neg(e2) = (X = c), (X = c) \wedge D \Rightarrow (X \leq c)$ is true for any $D$ and results in the coupling: $true \Rightarrow (X \leq c) \vee (X \neq c)$: This form of coupling is prohibited by Theorem 4.4 Condition (vii).

D. $e2 = (X = c), \neg(e2) = (X \neq c), (X \neq c) \wedge D \Rightarrow (X \leq c)$, $D$ must imply $(X \leq c)$ - no longer minimal.

E. $e2 = (X = Y), \neg(e2) = (X \neq Y), ((X \neq Y) \wedge D) \Rightarrow (X \leq c)$, $D$ must imply $(X \leq c)$ - no longer minimal.

xiii. $e1 = (X \leq c)$, need to imply $(X \leq c)$, symmetric to 3.

xiv. $e1 = X \neq c$, the rest need to imply $(X \neq c)$. Again, we examine the cases where $e2$ takes forms 5-7, since the first 4 cases have been covered above.

A. $e2 = (X = c), \neg(e2) = (X \neq c), (X \neq c) \wedge D \Rightarrow (X \neq c)$, any $D$ resulting in the coupling: $true \Rightarrow ((X = c) \vee (X \neq c))$. Theorem 4.4 Condition (vi) prohibits this form of coupling.

B. $e2 = (X = Y), \neg(e2) = (X \neq Y), (X \neq Y) \wedge D \Rightarrow (X \neq c)$. As shown above, a non-redundant $(X \neq Y)$ can not be used in conjunction with an AC to imply another AC while not violating Theorem 4.4 Condition (v).

C. $e = (X = c)$ the rest need to imply $(X = c)$. The other cases have been covered above since $e1 \wedge e2 = e2 \wedge e1$. Let $e2 = (X = Y), \neg(e2) = (X \neq Y), (X \neq Y) \wedge D \Rightarrow (X = c)$, D must imply $(X = c)$ - no longer minimal.

D. $e1 : (X = Y)$, the rest need to imply $(X = Y)$. All cases have been covered previously, since $e1 \wedge e2 = e2 \wedge e1$.

## A.3 Tightness of Conditions in the LSI Case

Here we mainly provide examples to argue that the conditions stated in Section 4.2 are tight. For each of the three conditions in Theorem 4.2, we give an example that dissatisfies only this condition, and show that more than one mapping have to be used to prove the containment. These examples also give some intuition on the proof of Theorem 4.2.

### A.3.1 On Condition (i): Single-Variable Coupling

Condition (i) avoids coupling (i) in Lemma A.1:

$$(X \leq c) \Rightarrow ((X < c) \vee (X = c))$$

**EXAMPLE A.1** The following is an example where there is no single mapping due to a coupling of the form (i).

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

Correspondingly we have the normalized queries:

$$Q_1' : ans() \text{ :- } p(X, Y), X < 4, Y = 4$$
$$Q_2' : ans() \text{ :- } p(X, Y), p(Z, U), X \leq 4, Y = 4, Z = 3, U = X$$

There are two containment mappings from $core(Q_1')$ to $core(Q_2'))$: $\mu_1(X) = X, \mu_1(Y) = Y$, and $\mu_2(X) = Z, \mu_2(Y) = U$. $Q_2'$ is contained in $Q_1'$ because of the implication

$$((X \leq 4) \wedge (Y = 4) \wedge (Z = 3) \wedge (U = X)) \Rightarrow$$
$$(((X < 4) \wedge (Y = 4)) \vee ((Z = 3) \wedge (U = X)))$$

□

Notice that Coupling (i) occurs due to the same constant appearing in the LSI arithmetic comparisons, as well as in a comparison of the form $X = c$. This comparison is introduced in LSI queries due to the normalization process. To prevent this coupling, condition (i) is sufficient. This example shows the tightness of condition (i) — it dissatisfies (i), but not (ii) and (iii), as $core(Q_1)$ has no shared variables.

### A.3.2 On Condition (ii): Multi-Variable Coupling

Condition (ii) avoids coupling (ii) in Lemma A.1:

$$(X \leq c \wedge Y = c) \Rightarrow (X < c \vee X = Y)$$

**EXAMPLE A.2** This example shows that there is no single mapping due to a coupling of the form (ii).

$$Q_1 : ans() \text{ :- } p(A, B, B), A < 4$$
$$Q_2 : ans() \text{ :- } p(X, Y, Y), p(U, X, 4), X \leq 4, U < 4$$

After normalizing the queries we have:

$$Q_1' : ans() \text{ :- } p(A, B, C), A < 4, B = C$$
$$Q_2' : ans() \text{ :- } p(X, Y, W), p(U, V, Z), X \leq 4, Z = 4, U < 4, Y = W, V = X$$

There are two mappings: $\mu_1(A) = X, \mu_1(B) = Y, \mu_1(C) = W$, and $\mu_2(A) = U, \mu_2(B) = V, \mu_2(C) = Z$. $Q_1'$ contains $Q_2'$ because the implication

$$((X \leq 4) \wedge (Z = 4) \wedge (U < 4) \wedge (Y = W) \wedge (V = X) \Rightarrow$$
$$((X < 4) \wedge (Y = W)) \vee ((U < 4) \wedge (V = Z))$$

holds. □

In this type of coupling, we note that we require shared variables in the containing query and the same constant in the comparisons of the two queries as well as the core of the contained query. Note that $Y = 4$ comes from the process of normalization, and 4 appears in the core of the contained query. To avoid this form of coupling, the condition (ii) is sufficient.

This example also shows the tightness of condition (ii). It dissatisfies condition (ii), but not (i) and (iii). Condition (i) is satisfied because $core(Q_1)$ does not contain a constant. Condition (iii) is satisfied because there are not two distinct variables in $Q_2$ in a comparison with $\leq 4$.

### A.3.3 On condition (iii): Multi-Variable Coupling

Condition (iii) avoids coupling (iii) in Lemma A.1:

$$(X \leq c) \wedge (Y \leq c) \Rightarrow (X < c) \vee (Y < c) \vee (X = Y)$$

**EXAMPLE A.3** This example shows there is no single mapping due to coupling of the form (iii).

$$Q_1 : ans() \text{ :- } p(A, B, B), A < 4$$
$$Q_2 : ans() \text{ :- } p(X, U, U), p(Y, V, V), p(Z, X, Y), X \leq 4, Z < 4, Y \leq 4$$

In this example, after normalizing the queries, we have three mappings that are used to prove containment. □

This example also shows the tightness of condition (iii). It dissatisfies condition (iii) but not (i) and (ii). Condition (i) is satisfied because $core(Q_1)$ does not contain a constant. Condition (ii) is satisfied because a constant does not appear in $core(Q_2)$. To avoid this form of coupling, condition (iii) is sufficient.

## A.4 Tightness of Conditions in the SI-PI Case

We first discuss the case where the containing query uses only LSI or RSI or both and then we discuss the case where it may use also point ibnequalities.

### SI case

Condition (iv)(a) avoids coupling (iv)(a):

$$Coupling(iv)(a): \ TRUE \Rightarrow ((X\theta_1 c1) \vee (X\theta_2 c2))$$

where $\theta_1$ is $<$ or $\leq$, $\theta_2$ is $>$ or $\geq$, and $c2 \leq c1$. For example, $TRUE \Rightarrow ((X < 5) \vee (X > 3)$.

**EXAMPLE A.4** The following is such an example.

$$Q_1 : ans() \text{ :- } p(X, Y), X < 5, Y > 3$$
$$Q_2 : ans() \text{ :- } p(X, Y), p(Y, Z), X < 5, Z > 3$$

$Q_2$ is contained in $Q_1$ and two containment mappings are necessary to prove the containment The coupling implication to prove the containment is of the same form (iv). □

A similar coupling is avoided by Condition (iv)(b):

$$Coupling(iv)(b): \ TRUE \Rightarrow ((X > c) \vee (X < c) \vee X = c)$$

### SI-PI case

Condition (v) avoids coupling (v):

$$Coupling(v): TRUE \Rightarrow (X \neq c) \vee (X = Y) \vee (Y \neq c)$$

For instance, consider the following queries.

$$Q_1 : ans() \text{ :- } p(X, Y, Y), X \neq 5$$
$$Q_2 : ans() \text{ :- } p(X, A, A), p(Y, B, B), p(C, X, Y)$$

Condition (vi) avoids coupling (vi):

$$Coupling(vi): TRUE \Rightarrow ((X = c) \vee (X \neq c))$$

For instance, consider the following queries.

$$Q_1 : ans() \text{ :- } p(X, Y), X = 5, Y \neq 5$$
$$Q_2 : ans() \text{ :- } p(X, A), p(B, X), A \neq 5, B = 5$$

Condition (vii)-lsi avoids coupling (vii)-lsi:

$$Coupling(vii): TRUE \Rightarrow (X \leq c) \vee (X \neq c)$$

For instance, consider the following queries:

$$Q_1 : ans() \text{ :- } p(X, Y), X \leq 5, Y \neq 5$$
$$Q_2 : ans() \text{ :- } p(X, A), p(A, X)$$

Note that for each example above a) there is no single mapping that proves containment, and b) *all* conditions *except one* are satisfied. This implies that none of the conditions are redundant.

## A.5 Beyond Semi-Interval Queries–continue

Coupling 2: Additional couplings can occur due to the following implication:

$$TRUE \Rightarrow ((X < Y) \vee (Y < X) \vee (X = Y))$$

indicates that if the containing queries have open comparisons with shared variables, then the homomorphism property does not hold. The following is such an example. Consider two queries.

$$Q_1 : ans() \text{ :- } p(X, Y, Z, Z), X < Y$$
$$Q_2 : ans() \text{ :- } p(X, Y, A, A), p(Y, X, B, B), p(C, D, X, Y), C < D$$

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

Coupling 3: Even without shared variables, the following implication shows a possible coupling:

$$((Y > c') \wedge (c' \geq c)) \Rightarrow ((X > c) \vee (X < Y))$$

The following is such an example.

$$Q_1 : ans() \text{ :- } p(A, B, C), A > 3, B < C$$
$$Q_2 : ans() \text{ :- } p(X, A, B), p(D, X, Y), Y > 4, A < B, D > 3$$

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

## A.6 Detailed Experimental Results on TPC-H Queries

Now we give experimental results for some of the queries. [[[ **By Foto: "Here is a list of the 22 queries where the queries (in their non-aggregate form) are concisely listed together with some comments. (Chen, this is not to be included in the paper, for the moment, it is useful to me (and I hope in our discussions)." By Chen: I keep them, since it's for the extended version anyway.** ]]]

1. Query $Q_1$:
   - Core (i.e., in the "from" clause): $lineitem$.
   - ACs: $l\_shipdate \leq date'1998 - 12 - 01' - interval'[DELTA]'day(3)$.
   - Comment: There is a single LSI comparison predicate, and the constant there may appear (we also take domain information into account) in either of the following fields in the core (i.e., the core of $lineitem$): $receiptdate$ and $commitdate$.
2. Query $Q_2$: No ACs.
3. Query $Q_3$:
   - Core: customer, order, lineitem
   - ACs: $o\_orderdate < date, l\_shipdate > date$
   - Comment: Here there is a scenario which is not desirable (i.e., the property does not hold): In the contained query, the orderdate of a 'order' subgoal contains the same variable as the shipdate of a lineitem subgoal. By "common sense" reasoning, this is not a plausible scenario.
4. Query $Q_4$:
   - Core: order, lineitem
   - ACs: $o\_orderdate >= date'[DATE]'$, $o\_orderdate < date'[DATE]' + interval'3'month$, $l\_orderkey = o\_orderkey$, $l\_commitdate < l\_receiptdate$.
   - Comment: There are non-SI ACs. Our algorithm does not apply.
5. Query $Q_5$:
   - Core: customer, orders, lineitem, supplier, nation, region
   - ACs: $o\_orderdate >= date'[DATE]'$, $o\_orderdate < date'[DATE]' + interval'1'year$
6. Query $Q_6$:
   - Core: lineitem
   - ACs: `l_shipdate >= date '[DATE]' and l_shipdate < date '[DATE]' + interval '1' year and l_discount between [DISCOUNT] - 0.01 and [DISCOUNT] + 0.01 and l_quantity < [QUANTITY];`
   - Comment: same situation as $Q_5$.
7. Query $Q_7$:
   - Core: supplier
   - lineitem, orders, customer, nation n1, nation n2
   - ACs: `s_suppkey = l_suppkey and o_orderkey = l_orderkey and c_custkey = o_custkey and s_nationkey = n1.n_nationkey and c_nationkey = n2.n_nationkey and ( (n1.n_name = '[NATION1]' and n2.n_name = '[NATION2]') or (n1.n_name = '[NATION2]' and n2.n_name = '[NATION1]') ) and l_shipdate between date '1995-01-01' and date '1996-12-31' ) as shipping`
   - Comment:same situation as $Q_5$.
8. Query $Q_8$:
   - Core.
   - ACs:
   - Comment:same as $Q_5$
9. Query $Q_9$:
   - Core:

- ACs:s_suppkey = l_suppkey and ps_suppkey = l_suppkey and ps_partkey = l_partkey and p_partkey = l_partkey and o_orderkey = l_orderkey and s_nationkey = n_nationkey and p_name like '%[COLOR]%'
- Comment: no ACs.

10. Query $Q_{10}$:
    - Core:
    - ACs:
    - Comment: same as $Q_5$.
11. Query $Q_{11}$:
    - Core:
    - ACs:
    - Comment:no ACs.
12. Query $Q_{12}$:
    - Core:
    - ACs: o_orderpriority <> '1-URGENT' and o_orderpriority <> '2-HIGH' and l_commitdate < l_receiptdate and l_shipdate < l_commitdate and l_receiptdate >= date '[DATE]' and l_receiptdate < date '[DATE]' + interval '1' year
    - Comment: domain information reduces it to "same as $Q_5$".
13. Query $Q_{13}$:
    - Core:
    - ACs:
    - Comment: no ACs.
14. Query $Q_{14}$:
    - Core:
    - ACs:
    - Comment: same as $Q_5$.
15. Query $Q_{15}$:
    - Core:
    - ACs:
    - Comment: same as $Q_5$
16. Query $Q_{16}$:
    - Core:
    - ACs: p_brand <> '[BRAND]'
    - Comment: the homomorphism propety holds.
17. Query $Q_{17}$:
    - Core:
    - ACs: l_quantity < QUANT
    - Comment: the homomorphism propety holds.
18. Query $Q_{18}$:
    - Core:
    - ACs: sum(l_quantity) > [QUANTITY]
    - Comment: the homomorphism propety holds.
19. Query $Q_19$: a disjunction of 3
    - Core: lineitem, part
    - ACs: p_partkey = l_partkey and p_brand = '[BRAND1]' and p_container in ( 'SM CASE', 'SM BOX', 'SM PACK', 'SM PKG') and l_quantity >= [QUANTITY1] and l_quantity <= [QUANTITY1] + 10 and p_size between 1 and 5 and l_shipmode in ('AIR', 'AIR REG') and l_shipinstruct = 'DELIVER IN PERSON'
      OR p_partkey = l_partkey and p_brand = '[BRAND2]' and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK') and l_quantity >= [QUANTITY2] and l_quantity <= [QUANTITY2] + 10 and p_size between 1 and 10 and l_shipmode in ('AIR', 'AIR REG') and l_shipinstruct = 'DELIVER IN PERSON'
      OR p_partkey = l_partkey and p_brand = '[BRAND3]' and p_container in ( 'LG CASE', 'LG BOX', 'LG PACK', 'LG PKG') and l_quantity >= [QUANTITY3] and l_quantity <= [QUANTITY3] + 10 and p_size between 1 and 15 and l_shipmode in ('AIR', 'AIR REG') and l_shipinstruct = 'DELIVER IN PERSON'
    - Comment: same as $Q_5$.
20. Query $Q_{20}$:
    - Core: supplier, nation
    - ACs: ps_availqty > QUANT l_shipdate >= date('[DATE]') and l_shipdate < date('[DATE]') + interval '1' year
    - Comment: same as $Q_5$
21. Query $Q_{21}$:
    - Core: supplier, lineitem l1, orders, nation
    - ACs: s_suppkey = l1.l_suppkey and o_orderkey = l1.l_orderkey and o_orderstatus = 'F' and l1.l_receiptdate > l1.l_commitdate and exists ( select * from lineitem l2 where l2.l_orderkey = l1.l_orderkey and l2.l_suppkey <> l1.l_suppkey ) and not exists ( select * from lineitem l3 where l3.l_orderkey = l1.l_orderkey and

```
l3.l_suppkey <> l1.l_suppkey and
l3.l_receiptdate > l3.l_commitdate
) and s_nationkey = n_nationkey and
n_name = '[NATION]'
```

- Comment:

22. Query $Q_{22}$:
    - Core:
    - ACs:
    - Comment: