Containment of Conjunctive Queries with Safe Negation

Fang Wei and Georg Lausen

University of Freiburg, Institute for Computer Science, Germany
{fwei,lausen}@informatik.uni-freiburg.de

Abstract. We consider the problem of query containment for conjunctive queries with safe negated subgoals ($CQ^{\neg}$s). We propose a new method for the containment test of $CQ^{\neg}$s. Comparing to the previous known approach, which always requires an exponential number of canonical databases to be verified to prove that $Q_1 \sqsubseteq Q_2$, the algorithm proposed in this paper exploits the containment mappings of their positive counterparts, and terminates once the specified test succeeds. We show that in the worst case, the algorithm has the same performance as the one proposed in previous work. We also extend our algorithm to unions of $CQ^{\neg}$s in a natural way. Due to the close relation between query containment and answering queries using views, we give some notes on considering answering queries using views when both queries and views have safe negated subgoals.

1 Introduction

This paper considers the problem of query containment of conjunctive queries (CQs) with safe negated subgoals $CQ^{\neg}$s. The query containment problem is to check whether the answer set of one query is always a subset of another query for all databases. Algorithms for query containment are of interest in several contexts in the database area.

Recently, there is a renewed interest in containment checking of conjunctive queries. The main motivation lies in its tight relation to the problem of *answering queries using views* [9, 1], which arises as the central problem in data integration and data warehousing (see [14] for a survey). Furthermore, query containment has also been used for checking integrity constraints [6], and for deciding query independence of updates [10].

Based on the NP-completeness result proposed by Chandra and Merlin [2], many researchers have been working on extensions of the containment question. Containment of CQs with inequalities is discussed in [8, 15]. Containment of unions of CQs is treated in [12], containment of CQs with negated subgoals in [10, 14], containment over complex objects in [11], and over semi-structured data with regular expressions in [5].

The containment problem for conjunctive queries with safe negated subgoals has drawn considerably less attention in the past. In [10] uniform containment is discussed, which is a sufficient, however not necessary condition for containment.

In [14] it is argued that the complexity of the containment test is Π_2^p-complete. An algorithm based on the approach of *canonical databases* was sketched which tests an exponential number of canonical databases. The following example explains the approach:

Example 1. Consider the following queries Q_1 and Q_2:

$$\begin{aligned}&Q_1 : \mathtt{q}(X, Z) \text{ :- } \mathtt{a}(X, Y), \mathtt{a}(Y, Z), \neg\mathtt{a}(X, Z).\\ &Q_2 : \mathtt{q}(A, C) \text{ :- } \mathtt{a}(A, B), \mathtt{a}(B, C), \mathtt{a}(B, D), \neg\mathtt{a}(A, D).\end{aligned}$$

In order to show that $Q_1 \sqsubseteq Q_2$, the approach in [14] considers all five partitions of $\{X, Y, Z\}$ in Table 1: all variables in one set of a certain partition are replaced by the same constant. From each partition, a canonical database, built out of the positive subgoals, is generated according to the predicates in the body of Q_1. At first, Q_1 has to be applied to the canonical database D from each partition, and if the answer set is not empty, then the same answer set has to be obtained from $Q_2(D)$. Next, for each canonical database D which results in a nonempty answer, we have to extend it with "other tuples that are formed from the same symbols as those in D". In fact, for each specific predicate, let k be the number of arguments of the predicate, n be the number of symbols in the canonical database, and r be the number of subgoals of Q_1 (both positive and negative), there will be $2^{(n^k - r)}$ sets of tuples which have to be checked. Taking the partition $\{X\}\{Y\}\{Z\}$, we need to consider 6 other tuples: $\{\mathtt{a}(0, 0), \mathtt{a}(1, 0), \mathtt{a}(1, 1), \mathtt{a}(2, 0), \mathtt{a}(2, 1), \mathtt{a}(2, 2)\}$. At the end, one has to check 2^6 canonical databases, and if for each database D', $Q_2(D')$ yields the same answer as Q_1, it can then be concluded that $Q_1 \sqsubseteq Q_2$, which is true in this example. □

Table 1. The five canonical databases and their answers to Q_1 and Q_2

Partition	Canonical Databases	Answers
$\{X\}\{Y\}\{Z\}$	$\{\mathtt{a}(0, 1), \mathtt{a}(1, 2)\}$	$Q_1 : \mathtt{q}(0, 2); Q_2 : \mathtt{q}(0, 2)$
$\{X, Y\}\{Z\}$	$\{\mathtt{a}(0, 0), \mathtt{a}(0, 1)\}$	Q_1 : false
$\{X\}\{Y, Z\}$	$\{\mathtt{a}(0, 1), \mathtt{a}(1, 1)\}$	Q_1 : false
$\{X, Z\}\{Y\}$	$\{\mathtt{a}(0, 1), \mathtt{a}(1, 0)\}$	$Q_1 : \mathtt{q}(0, 0); Q_2 : \mathtt{q}(0, 0)$
$\{X, Y, Z\}$	$\{\mathtt{a}(0, 0)\}$	Q_1 : false

Example 2. Consider the following queries Q_1 and Q_2:

$$\begin{aligned}&Q_1 : \mathtt{q}(X, Z) \text{ :- } \mathtt{a}(X, Y), \mathtt{a}(Y, Z), \neg\mathtt{a}(X, Z).\\ &Q_2 : \mathtt{q}(A, C) \text{ :- } \mathtt{a}(A, B), \mathtt{a}(B, C), \neg\mathtt{b}(C, C).\end{aligned}$$

This example differs from Example 1 by the negated subgoal. The application of Q_2 to the canonical databases shown in Table 1 yields the same answer as Q_1.

Similar to the above example, extra tuples have to be added into the canonical database. Taking the partition $\{X\}\{Y\}\{Z\}$, we have 15 other tuples (9 tuples with $\mathtt{b}(0,0),\ldots,\mathtt{b}(2,2)$ and 6 tuples as in Example 1), such that 2^{15} canonical databases have to be verified. Since the database $D = \{\mathtt{a}(0,1), \mathtt{a}(1,2), \mathtt{b}(2,2)\}$ is a counter-example such that $Q_2(D)$ does not generate the same answer as $Q_1(D)$, the test terminates with the result $Q_1 \not\sqsubseteq Q_2$. □

In this paper, we propose a new method to solve the general query containment problem for conjunctive queries with safe negated subgoals ($CQ^{\neg}$s). Given two $CQ^{\neg}$s Q_1 and Q_2, and their positive counterparts ${Q_1}^+$ and ${Q_2}^+$ (definitions in Section 3.1), we show that there are two factors deciding the complexity of the problem $Q_1 \sqsubseteq Q_2$:

- ${Q_1}^+ \sqsubseteq {Q_2}^+$? This is a necessary condition.
- the number of containment mappings from ${Q_2}^+$ to ${Q_1}^+$.

Comparing to the algorithm described in [14], which requires always an exponential number of canonical database to be tested to prove the result $Q_1 \sqsubseteq Q_2$, the algorithm proposed in this paper exploits the containment mappings from Q_2^+ to Q_1^+, and terminates when the specified tests succeed. We show that in the worst case, the algorithm has the same performance as the one proposed in [14]. Our algorithm also extends naturally to unions of $CQ^{\neg}$s. Due to the close relation between query containment and answering queries using views, we give some notes on considering answering queries using views when both queries and views have safe negated subgoals allowed.

The rest of the paper is organized as follows: in Section 2 we recall the definition of a $CQ^{\neg}$ and containment of both CQs and $CQ^{\neg}$s. In Section 3, we first prove two necessary conditions for the containment test of $CQ^{\neg}$s, which is then followed by the main theorem of the paper. The proof of correctness and completeness is given as well. In Section 4 the theorem and the algorithm based on it are naturally extended to the containment test of unions of $CQ^{\neg}$s. In Section 5 we discuss the issues of answering queries using views when both queries and views have negated subgoals allowed. Finally the conclusions and future work are presented.

2 Preliminaries

2.1 Query Containment

A *conjunctive query with negation* ($CQ^{\neg}$) extends a *conjunctive query CQ* by allowing negated subgoals in the body. It has the following form:

$$h(\bar{X}) \text{ :- } p_1(\bar{X_1}), \ldots, p_n(\bar{X_n}), \neg s_1(\bar{Y_1}), \ldots, \neg s_m(\bar{Y_m}).$$

where $h, p_1, \ldots, p_n, s_1, \ldots, s_m$ are predicates whose arguments are variables or constants, $h(\bar{X})$ is the *head*, $p_1(\bar{X_1}), \ldots, p_n(\bar{X_n})$ are the positive subgoals, and

$s_1(\bar{Y_1}), \ldots, s_m(\bar{Y_m})$ are the negated subgoals. We assume that, firstly, the variables occurring in the head also occur in the body; secondly, all the variables occurring in the negated subgoals also occur in positive ones, which is also called the *safeness* condition for $CQ^\neg$. The examples in this paper are *safe* $CQ^\neg$s if not mentioned otherwise.

A $CQ^\neg$ is applied to a set of finite database relations by considering all possible substitutions of values for the variables in the body. If a substitution makes all the positive subgoals true and all the negated subgoals false (i.e. they do not exist in the database), then the same substitution, applied to the head, composes one answer of the conjunctive query. The set of all answers to a query Q with respect to a certain database D is denoted by $Q(D)$.

Unlike CQs with only positive subgoals, which are always satisfiable, $CQ^\neg$s might be unsatisfiable.

Proposition 1. *A $CQ^\neg$ is unsatisfiable if and only if there exist $p_i(\bar{X_i})(1 \leq i \leq n)$ and $s_j(\bar{Y_j})(1 \leq j \leq m)$ such that $p_i = s_j$ and $\bar{X_i} = \bar{Y_j}$.* □

From now on, we only refer to satisfiable $CQ^\neg$s, if not otherwise mentioned.

The containment of $CQ^\neg$s is defined in the same manner as for positive ones: a $CQ^\neg$ Q_1 is contained in another one Q_2, denoted as $Q_1 \sqsubseteq Q_2$, if for all databases D, $Q_1(D) \subseteq Q_2(D)$. Two $CQ^\neg$s are equivalent if and only if they are contained in each other.

2.2 The Containment Checking Algorithm for *CQ*s

An algorithm for checking the containment of CQs was proposed in [2].

Lemma 1 (Containment of CQs [2]). *Consider two CQs Q_1 and Q_2:*

$$\begin{array}{l} Q_1 : h(\bar{X}) \text{ :- } p_1(\bar{X_1}), \ldots, p_n(\bar{X_n}). \\ Q_2 : h(\bar{U}) \text{ :- } q_1(\bar{U_1}), \ldots, q_l(\bar{U_l}). \end{array}$$

Then $Q_1 \sqsubseteq Q_2$ if and only if there exists a containment mapping ρ from the variables of subgoals in Q_2 to those in Q_1, such that $\{\rho(q_1(\bar{U_1})), \ldots, \rho(q_l(\bar{U_l}))\} \subseteq \{p_1(\bar{X_1}), \ldots, p_n(\bar{X_n})\}$, and $\rho(h(\bar{U})) = h(\bar{X})$. □

When the heads of both Q_1 and Q_2 do not contain variables, they are called boolean queries. It is obvious that boolean queries are the generalized forms of normal queries. The containment problem for CQs is shown to be NP-complete [2].

Example 3. Consider the queries Q_1 and Q_2: the bodies of the queries are composed of the positive subgoals from Example 1.

$$\begin{array}{l} Q_1 : \mathsf{q}(X, Z) \text{ :- } \mathsf{a}(X, Y), \mathsf{a}(Y, Z). \\ Q_2 : \mathsf{q}(A, C) \text{ :- } \mathsf{a}(A, B), \mathsf{a}(B, C), \mathsf{a}(B, D). \end{array}$$

There is one and *only one* containment mapping from Q_2 to Q_1:
$\{A \rightarrow X, B \rightarrow Y, C \rightarrow Z, D \rightarrow Z\}$. □

3 Query Containment for $CQ^{\neg}$s

In this section we discuss the containment checking for $CQ^{\neg}$s. In the next subsection we introduce some necessary conditions.

3.1 Some Necessary Conditions

Definition 1 (Super-Positive SP Q^+). *Given a $CQ^{\neg}$ Q as follows:*

$$Q : h(\bar{X}) \text{ :- } p_1(\bar{X_1}), \ldots, p_n(\bar{X_n}), \neg s_1(\bar{Y_1}), \ldots, \neg s_m(\bar{Y_m}).$$

The SP of Q, denoted as Q^+ is: $h(\bar{X})$:- $p_1(\bar{X_1}), \ldots, p_n(\bar{X_n})$. □

Lemma 2. *Given a $CQ^{\neg}$ Q with negated subgoals and its SP Q^+, $Q \sqsubseteq Q^+$ holds.* □

Proposition 2. *Let Q_1 and Q_2 be two $CQ^{\neg}$s, let Q_1^+ and Q_2^+ be their SP respectively. $Q_1 \sqsubseteq Q_2$ only if $Q_1^+ \sqsubseteq Q_2^+$.*

Proof. Assume $Q_1 \sqsubseteq Q_2$ and a tuple $t \in Q_1^+(D)$ where D is any canonical database (i.e. each variable is assigned to a unique constant) of Q_1^+. We show that $t \in Q_2^+(D)$: Let ρ be the substitution from variables of Q_1 to distinct constants in D. Let $s_i(\bar{Y_i})(1 \leq i \leq m)$ be any negated subgoal in Q_1. Since Q_1 is satisfiable, therefore we obtain that $\rho(s_i(\bar{Y_i})) \notin D$. Consequently, $t \in Q_1(D)$ and $t \in Q_2(D)$ are obtained. Following Lemma 2 it is obvious that $t \in Q_2^+(D)$.

Proposition 2 provides a necessary but not sufficient condition for query containment of $CQ^{\neg}$s. Next we give a theorem, stating a condition for $Q_1 \not\sqsubseteq Q_2$.

Theorem 1. *Let Q_1 and Q_2 be two $CQ^{\neg}$s. Assume $Q_1^+ \sqsubseteq Q_2^+$, and let $\rho_1, \ldots, \rho_r$ be all containment mappings from Q_2^+ to Q_1^+, such that $Q_1^+ \sqsubseteq Q_2^+$. Q_1 and Q_2 are given as follows:*

$$\begin{aligned} Q_1 &: h(\bar{X}) \text{ :- } p_1(\bar{X_1}), \ldots, p_n(\bar{X_n}), \neg s_1(\bar{Y_1}), \ldots, \neg s_m(\bar{Y_m}). \\ Q_2 &: h(\bar{U}) \text{ :- } q_1(\bar{U_1}), \ldots, q_l(\bar{U_l}), \neg a_1(\bar{W_1}), \ldots, \neg a_k(\bar{W_k}). \end{aligned}$$

If for each $\rho_i(1 \leq i \leq r)$, there exists at least one $j(1 \leq j \leq k)$, such that $\rho_i(a_j(\bar{W_j})) \in \{p_1(\bar{X_1}), \ldots, p_n(\bar{X_n})\}$, then $Q_1 \not\sqsubseteq Q_2$.

Proof. A canonical database D could be constructed as follows: freeze the positive subgoals of Q_1 and replace each variable in Q_1 with a distinct constant. We call this substitution σ. Let t be any tuple such that $t \in Q_1(D)$, we have to show that $t \notin Q_2(D)$: that is, for each substitution θ which makes $t \in Q_2^+(D)$ true, there is at least one negated subgoal $a_j(\bar{W_j})$, where $1 \leq j \leq k$, such that $\theta(a_j(\bar{W_j})) \in D$.

Since $\rho_1, \ldots, \rho_r$ are *all* the containment mappings from Q_2^+ to Q_1^+, it is true that $\theta \in \{\rho_1 \circ \sigma, \ldots, \rho_r \circ \sigma\}$. [1] Assume $\theta = \rho_i \circ \sigma$ $(1 \leq i \leq r)$. Since for each ρ_i, there exists a $j(1 \leq j \leq k)$, such that $\rho_i(a_j(\bar{W_j})) \in \{p_1(\bar{X_1}), \ldots, p_n(\bar{X_n})\}$, thus we have $\rho_i \circ \sigma(a_j(\bar{W_j})) \in \{\sigma(p_1(\bar{X_1})), \ldots, \sigma(p_n(\bar{X_n}))\}$ As a result, $\theta(a_j(\bar{W_j})) \in D$ is obtained.

[1] $\rho \circ \sigma$ denotes the composition of substitutions ρ and σ.

3.2 Containment of $CQ^{\neg}$s

The following theorem states a necessary and sufficient condition for the containment checking of $CQ^{\neg}$s, which is one of the main contributions of this paper.

Theorem 2. *Let Q_1 and Q_2 be two $CQ^{\neg}$s as follows:*

$$\begin{array}{l} Q_1 : h(\bar{X}) \text{ :- } p_1(\bar{X_1}), \ldots, p_n(\bar{X_n}), \neg s_1(\bar{Y_1}), \ldots, \neg s_m(\bar{Y_m}). \\ Q_2 : h(\bar{U}) \text{ :- } q_1(\bar{U_1}), \ldots, q_l(\bar{U_l}), \neg a_1(\bar{W_1}), \ldots, \neg a_k(\bar{W_k}). \end{array}$$

Then $Q_1 \sqsubseteq Q_2$ if and only if

1. *there is a containment mapping ρ from Q_2^+ to Q_1^+ such that $Q_1^+ \sqsubseteq Q_2^+$, and*
2. *for each $j(1 \le j \le k)$, $Q' \sqsubseteq Q_2$ holds, where Q' is as follows:*

$$Q' : h(\bar{X}) \text{ :- } p_1(\bar{X_1}), \ldots, p_n(\bar{X_n}), \neg s_1(\bar{Y_1}), \ldots, \neg s_m(\bar{Y_m}), \rho(a_j(\bar{W_j})).$$

Proof.

- $\Leftarrow$: Let D be any database and t the tuple such that $t \in Q_1(D)$, we have to prove that $t \in Q_2(D)$.
  Since $t \in Q_1(D)$, we have immediately $t \in Q_1^+(D)$ and $t \in Q_2^+(D)$. Let σ be the substitution from the variables in Q_1^+ to the constants in D such that $t \in Q_1^+(D)$. Let $\theta = \rho \circ \sigma$. It is apparent that $\{\theta(q_1(\bar{U_1})), \ldots, \theta(q_l(\bar{U_l}))\} \subseteq D$. If for each $j(1 \le j \le k)$, $\theta(a_j(\bar{W_j}))) \notin D$, then the result is straightforward. Otherwise, if there is any $j(1 \le j \le k)$, such that $\theta(a_j(\bar{W_j}))) \in D$, then we have $\rho \circ \sigma(a_j(\bar{W_j}))) \in D$. it can be concluded that $t \in Q'(D)$ where Q' is

  $$Q' : h(\bar{X}) \text{ :- } p_1(\bar{X_1}), \ldots, p_n(\bar{X_n}), \neg s_1(\bar{Y_1}), \ldots, \neg s_m(\bar{Y_m}), \rho(a_j(\bar{W_j})).$$

  From the assumption that $Q' \sqsubseteq Q_2$ in the above theorem, $t \in Q_2(D)$ can then be obtained.
- $\Rightarrow$: The proof is via deriving a contradiction.
  1. If $Q_1^+ \not\sqsubseteq Q_2^+$, then from Proposition 2, $Q_1 \not\sqsubseteq Q_2$ can be obtained immediately.
  2. Otherwise, if for each containment mapping ρ from ${Q_2}^+$ to ${Q_1}^+$, such that ${Q_1}^+ \sqsubseteq {Q_2}^+$, there is at least one Q' as given in the above theorem, such that $Q' \not\sqsubseteq Q_2$, then there exists at least one database D, such that $t \in Q'(D)$, but $t \notin Q_2(D)$. Since Q' has only one more positive subgoal than Q_1, it is obvious that $t \in Q_1(D)$. This leads to the result that $Q_1 \not\sqsubseteq Q_2$.

Theorem 2 involves a recursive containment test. In each round, the containment $Q' \sqsubseteq Q_2$ (the definition of Q' see the above theorem) has to be verified. This might lead to one of the two results: (1) for each Q', there is $Q' \sqsubseteq Q_2$ – either Q' is unsatisfiable, or via recursive containment test –, then the test terminates with the result $Q_1 \sqsubseteq Q_2$; (2) there exists a Q', such that $Q' \not\sqsubseteq Q_2$. This can be verified according to Theorem 1. In this case, the result of $Q_1 \not\sqsubseteq Q_2$ can be obtained. The following example illustrates the algorithm. Note that we intentionally omit the variables in the head in order to *generate* more containment mappings from Q_2^+ to Q_1^+. It is not difficult to understand that the fewer containment mappings are there from Q_2^+ to Q_1^+, the simpler the test will be.

Example 4. Given the queries Q_1 and Q_2:

$$\begin{array}{l} Q_1 : h \text{ :- } \mathtt{a}(X,Y), \mathtt{a}(Y,Z), \neg\mathtt{a}(X,Z). \\ Q_2 : h \text{ :- } \mathtt{a}(A,B), \mathtt{a}(C,D), \neg\mathtt{a}(B,C) \end{array}$$

There are four containment mappings from Q_2^+ to Q_1^+, one of which is $\rho_1 = \{A{\to}Y, B{\to}Z, C{\to}X, D{\to}Y\}$. Now a new conjunctive query is generated as follows:

$$Q' : h \text{ :- } \mathtt{a}(X,Y), \mathtt{a}(Y,Z), \mathtt{a}(Z,X), \neg\mathtt{a}(X,Z).$$

Note that the subgoal $\mathtt{a}(Z,X)$ is generated from $\rho_1(\mathtt{a}(B,C))$. One of the containment mappings from Q_2 to Q' is $\rho_2 = \{A{\to}Z, B{\to}X, C{\to}Z, D{\to}X\}$. Since the newly generated subgoal $\rho_2(\mathtt{a}(B,C))$ is $\mathtt{a}(X,Z)$, this leads to a successful unsatisfiability test of Q''.

$$Q'' : h \text{ :- } \mathtt{a}(X,Y), \mathtt{a}(Y,Z), \mathtt{a}(Z,X), \mathtt{a}(X,Z), \neg\mathtt{a}(X,Z).$$

it can then be concluded that $Q' \sqsubseteq Q_2$. In the sequel we have $Q_1 \sqsubseteq Q_2$. □

The detailed algorithm is given in the Appendix. The idea behind the algorithm can be informally stated as follows: we start with the positive subgoals of Q_1 as root. Let r be the number of all containment mappings from Q_2^+ to Q_1^+, such that $Q_1^+ \sqsubseteq Q_2^+$. r branches are generated from the root, with sets of mapped negated subgoals as nodes (cf. Figure 1(a)). Next, each node might be marked as *Contained*, if it is identical to one of the negated subgoals of Q_1, or as *Terminal*, if it is identical to one of the positive subgoals of Q_1. If there exists one branch such that each node is marked as *Contained*, then the program terminates with the result $Q_1 \sqsubseteq Q_2$. Otherwise, if at least one node of each branch is marked as *Terminal*, then the program terminates too, however with the result $Q_1 \not\sqsubseteq Q_2$. If none of these conditions is met, the program continues with expansion of the non-terminal nodes, that is, the nodes mark with *Terminal* will not be expanded any more.

It can be shown that the algorithm terminates. The reasons are: (1) the expansion does not generate new variables; (2) the number of variables in Q_1 is finite.

The next example shows how the algorithm terminates when the complement problem $Q_1 \not\sqsubseteq Q_2$ is solved.

Example 5. Given the queries Q_1 and Q_2:

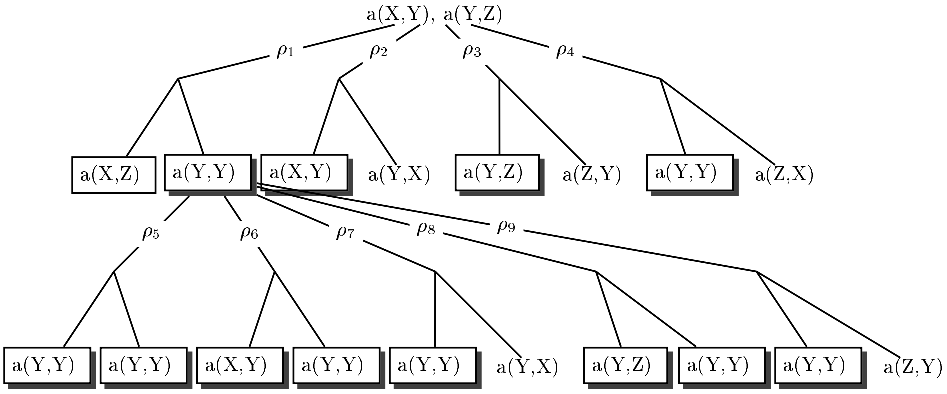
$$\begin{array}{l} Q_1 : h \text{ :- } \mathtt{a}(X,Y), \mathtt{a}(Y,Z), \neg\mathtt{a}(X,Z). \\ Q_2 : h \text{ :- } \mathtt{a}(A,B), \mathtt{a}(C,D), \neg\mathtt{a}(A,D), \neg\mathtt{a}(B,C) \end{array}$$

In Figure 1, it is shown that there are four containment mappings from Q_2^+ to Q_1^+: $\rho_1, \ldots, \rho_4$. Each mapping contains two sub-trees since there are two negated subgoals in Q_2. The branches ρ_2 and ρ_3 are marked as *Terminal* because there is at least one *Terminal* node from each of the above branches (note that we denote a node *Terminal* with a shadowed box around it, and *Contained* with a normal box). The node $\mathtt{a}(X, Z)$ from branch ρ_1 is marked as *Contained*, because it is the same as the negated subgoal in Q_1. (Note that in Figure 1 the node $\mathtt{a}(Y, Y)$ of branch ρ_1 is marked as *Terminal*, but it is the result of the next round. Up to now, it has not been marked. The same holds for the nodes of branch ρ_4.)

Next the non-terminal node $\mathtt{a}(Y, Y)$ is expanded. Five new containment mappings are generated as $\rho_5, \ldots, \rho_9$. Since all the branches are *Terminal*, and $\mathtt{a}(Y, Y)$ is also a sub-node of ρ_4, it can be concluded that the expanded query

$$Q' : h \text{ :- } \mathtt{a}(X, Y), \mathtt{a}(Y, Z), \mathtt{a}(Y, Y), \neg\mathtt{a}(X, Z).$$

is not contained in Q_2. Because all the containment mappings from Q_2^+ to Q'^+ have been verified. As a result, $Q_1 \not\sqsubseteq Q_2$ is obtained. □



(a) The generated tree

ρ_1	$\{A \to X, B \to Y, C \to Y, D \to Z\}$	ρ_2	$\{A \to X, B \to Y, C \to X, D \to Y\}$
ρ_3	$\{A \to Y, B \to Z, C \to Y, D \to Z\}$	ρ_4	$\{A \to Y, B \to Z, C \to X, D \to Y\}$
ρ_5	$\{A \to Y, B \to Y, C \to Y, D \to Y\}$	ρ_6	$\{A \to X, B \to Y, C \to Y, D \to Y\}$
ρ_7	$\{A \to Y, B \to Y, C \to X, D \to Y\}$	ρ_8	$\{A \to Y, B \to Y, C \to Y, D \to Z\}$
ρ_9	$\{A \to Y, B \to Z, C \to Y, D \to Y\}$		

(b) The containment mappings

Fig. 1. The graphic illustration of Example 5.

Comparison of the algorithms. We notice several interesting similarities and differences between our algorithm and the one in [14]: The partitioning of variables is similar to the step of checking of ${Q_1}^+ \sqsubseteq {Q_2}^+$. It can be proven that if the containment checking ${Q_1}^+ \sqsubseteq {Q_2}^+$ is successful, there exists at least one partition of variables, namely the partition with a distinct constant for each variable, such that when applied to the canonical database built from this partition, Q_2 yields the same answer set as Q_1.

The next step of the algorithm in [14] is to check an exponential number of canonical databases, as described in the Introduction. In contrast, our algorithm continues with the containment mappings of their positive counterparts and executes the specified test, which takes linear time in the size of Q_1. If the test is not successful, the query is extended with one more positive subgoal (but without new variables) and the next containment test continues. It is important to emphasize that in the worst case, the expanded tree generates all the nodes composed of variant combinations of the variables in Q_1, which coincides with the method in [14].

However, in the following examples we show that our algorithm terminates *definitely* earlier in the some cases:

(1) It turns out that the result of the containment test is $Q_1 \sqsubseteq Q_2$. As explained in the Introduction, the algorithm in [14] terminates if one canonical database as counter-example is found, but in order to obtain the result of $Q_1 \sqsubseteq Q_2$, *all* canonical databases have to be verified. In contrast, our algorithm terminates with the result $Q_1 \sqsubseteq Q_2$ if the specified test is successful.

Example 6. Given the queries Q_1 and Q_2 as in Example 1. There is a containment mapping ρ from Q_2^+ to Q_1^+ as in Example 3. There is only one Q' to be checked:

$$Q' : \mathtt{q}(X,Z) \text{ :- } \mathtt{a}(X,Y), \mathtt{a}(Y,Z), \neg\mathtt{a}(X,Z), \mathtt{a}(X,Z).$$

It is apparent that Q' is unsatisfiable, thus we have proven that $Q_1 \sqsubseteq Q_2$. □

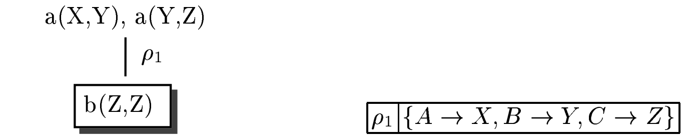
(2) To test $Q_1 \sqsubseteq Q_2$, if none of the predicates from negated subgoals in Q_2 appears in positive subgoals, the test terminates after the first round of containment checking of $Q' \sqsubseteq Q_2$. It can be explained as follows: due to the assumption, the containment mappings from Q_2 to the newly generated query Q' (see Theorem 2) are the same as the ones from Q_2 to Q_1. Since no *new* containment mapping is generated, the algorithm reaches some "fix-point", so that no new branches of the expanded tree will be generated.

Example 7. Given the queries Q_1 and Q_2 as in Example 2. There is a containment mapping ρ_1 (cf. Figure 2). A new node $\rho_1(\mathtt{b}(C,C)) = \mathtt{b}(Z,Z)$ is then generated. The new query Q' is the following:

$$Q' : \mathtt{q}(X,Z) \text{ :- } \mathtt{a}(X,Y), \mathtt{a}(Y,Z), \neg\mathtt{a}(X,Z), \mathtt{b}(Z,Z).$$

Since ρ_1 is the *only* containment mapping from Q_2 to Q', we mark the node $\mathtt{b}(Z,Z)$ as *Terminal*. Following Theorem 1, $Q' \not\sqsubseteq Q_2$ can be obtained, which is followed by the result $Q_1 \not\sqsubseteq Q_2$. □

At last, it should be mentioned that the algorithm in [14] can deal with unsafe negations, while ours cannot.



(a) The generated tree

(b) The containment mapping

Fig. 2. The graphic illustration of Example 7.

4 Containment of Unions of $CQ^{\neg}$s

In this section we consider the containment problem of unions of $CQ^{\neg}$s. First we present some basic notations.

Definition 2 (Union of CQs [13]). *Let $\mathcal{Q} = Q_1 \cup \ldots \cup Q_n$ be a union of CQs, in which $Q_1, \ldots, Q_n$ have a common head predicate.*

- *Given any database D, $\mathcal{Q}(D) = Q_1(D) \cup \ldots \cup Q_n(D)$.*
- *Let Q be a CQ and $\mathcal{Q}$ be a union of CQs. $Q \sqsubseteq \mathcal{Q}$ if $Q(D) \subseteq \mathcal{Q}(D)$ for any given database D.* □

Theorem 3 (Containment of unions of CQs [12]). *Let Q be a CQ and $\mathcal{Q} = Q_1 \cup \ldots \cup Q_n$ be a union of CQs, then $Q \sqsubseteq \mathcal{Q}$ if and only if there is a $Q_i (1 \leq i \leq n)$ such that $Q \sqsubseteq Q_i$.* □

However, when negation is allowed in CQs, the above theorem will not hold any more.

Example 8. Consider the following queries with the relations `man`, `father`, and `husband`:

$$
\begin{aligned}
Q\ &: q(X) \text{ :- } \mathtt{man}(X), \neg\mathtt{father}(X).\\
Q_1 &: q(Y) \text{ :- } \mathtt{man}(Y), \neg\mathtt{husband}(Y).\\
Q_2 &: q(Z) \text{ :- } \mathtt{man}(Z), \mathtt{husband}(Z), \neg\mathtt{father}(Z).
\end{aligned}
$$

It is apparent that neither $Q \sqsubseteq Q_1$, nor $Q \sqsubseteq Q_2$, but $Q \sqsubseteq Q_1 \cup Q_2$. □

The containment of unions of $CQ^{\neg}$s is defined in the similar way as in Definition 2. The following notations have to be additionally presented.

Definition 3. *Let $\mathcal{Q} = Q_1 \cup \ldots \cup Q_n$ be a union of $CQ^{\neg}$s. $\mathcal{Q}^+ = Q_1^+ \cup \ldots \cup Q_n^+$.* □

Lemma 3. *Let $\mathcal{Q} = Q_1 \cup \ldots \cup Q_n$ be a union of $CQ^{\neg}$s. $\mathcal{Q} \sqsubseteq \mathcal{Q}^+$ holds.* □

Lemma 4. *Given a $CQ^{\neg}$ Q and a union of $CQ^{\neg}$s $\mathcal{Q} = Q_1 \cup \ldots \cup Q_n$. If $Q \sqsubseteq \mathcal{Q}$, then there exists at least one $Q_i \in \mathcal{Q} (1 \leq i \leq n)$, such that $Q^+ \sqsubseteq Q_i^+$.*

Proof. (sketch) Using the same method as in Proposition 2, we have $Q^+ \sqsubseteq Q_1^+ \cup \ldots \cup Q_n^+$. Following [12] the result can be obtained.

Theorem 4. *Given a $CQ^\neg$ Q and a union of $CQ^\neg$s $\mathcal{Q} = Q_1 \cup Q_2 \cup \ldots \cup Q_v$. Assume $Q^+ \sqsubseteq \mathcal{Q}^+$, and let $\rho_{i,1}, \ldots, \rho_{i,r_i}$ (for $i = 1, \ldots, v$) be all the containment mappings from $\mathcal{Q}^+$ to Q^+, such that $Q^+ \sqsubseteq \mathcal{Q}^+$. Q and $Q_1, \ldots, Q_v$ are given as follows:*

$$\begin{array}{ll} Q: & h(\bar{X}) \text{ :- } p_1(\bar{X_1}), \ldots, p_n(\bar{X_n}), \neg s_1(\bar{Y_1}), \ldots, \neg s_m(\bar{Y_m}). \\ Q_i(1 \leq i \leq v): h(\bar{U}) & \text{:- } q_{i,1}(\bar{U_{i,1}}), \ldots, q_{i,l_i}(\bar{U_{i,l_i}}), \neg a_{i,1}(\bar{W_{i,1}}), \ldots, \neg a_{i,k_i}(\bar{W_{i,k_i}}). \end{array}$$

If for each $\rho_{i,j}(1 \leq i \leq v, 1 \leq j \leq r_i)$, there exists at least a $a_{i,u}(\bar{W_{i,u}})$, where $1 \leq u \leq k_i$, such that $\rho_{i,j}(a_{i,u}(\bar{W_{i,u}})) \in \{p_1(\bar{X_1}), \ldots, p_n(\bar{X_n})\}$, then $Q \not\sqsubseteq \mathcal{Q}$.

Proof. (sketch) A canonical database D can be built in the similar way as in Theorem 1. Then it can be proven that a tuple $t \in Q_1(D)$, but $t \notin Q_2(D)$.

The following theorem states a sound and complete condition.

Theorem 5. *Let Q be a $CQ^\neg$ and $\mathcal{Q}$ be the union of $CQ^\neg$s $\mathcal{Q} = Q_1 \cup Q_2 \cup \ldots \cup Q_v$ as follows:*

$$\begin{array}{ll} Q: & h(\bar{X}) \text{ :- } p_1(\bar{X_1}), \ldots, p_n(\bar{X_n}), \neg s_1(\bar{Y_1}), \ldots, \neg s_m(\bar{Y_m}). \\ Q_i(1 \leq i \leq v): h(\bar{U}) & \text{:- } q_{i,1}(\bar{U_{i,1}}), \ldots, q_{i,l_i}(\bar{U_{i,l_i}}), \neg a_{i,1}(\bar{W_{i,1}}), \ldots, \neg a_{i,k_i}(\bar{W_{i,k_i}}). \end{array}$$

Then $Q \sqsubseteq \mathcal{Q}$ if and only if

- *there is a containment mapping ρ from Q_u^+ to Q, where $1 \leq u \leq v$, such that $Q^+ \sqsubseteq \mathcal{Q}^+$, and*
- *for each $j(1 \leq j \leq k_u)$, $Q' \sqsubseteq \mathcal{Q}$ holds, where Q' is as follows:*

$$Q' : h(\bar{X}) \text{ :- } p_1(\bar{X_1}), \ldots, p_n(\bar{X_n}), \neg s_1(\bar{Y_1}), \ldots, \neg s_m(\bar{Y_m}), \rho(a_{u,j}(\bar{W_{u,j}})).$$

□

The algorithm behind the theorem can be adapted with little modification from that of containment checking for two $CQ^\neg$s. Next we explain how the containment checking of Example 8 is executed.

Example 9. Consider the queries Q, Q_1 and Q_2 in Example 8. Since Q_1 is the only query whose positive part of body contains that of Q, the containment mapping $\rho_1 = \{Y \to X\}$ is obtained. The newly generated Q' has the following form:

$$Q' : q(X) \text{ :- } \texttt{man}(X), \neg\texttt{father}(X), \texttt{husband}(X).$$

Since it is easy to check that $Q' \sqsubseteq Q_2$, it can then be concluded that $Q \sqsubseteq Q_1 \cup Q_2$.
□

5 Some Notes on Answering Queries using Views

The problem of answering queries using views has been intensively studied recently, especially in the area of data integration (see [7] for a survey). The problem can be described as follows: given a query Q over a database schema, and a set of view definitions $\mathcal{V}$ over the same schema, how to find a rewriting Q' of Q, using *only* the views. The notations of *equivalent rewriting* and *maximally-contained rewriting* can be found in [7].

Due to the close relation between the problem of query containment and answering queries using views, we are interested in giving the discussion of some new issues for answering queries using views, when safe negation is allowed in both queries and views. Since the *completeness* of any rewriting algorithm depends on the query language in which the rewritings are expressed, it has to be specified whether queries with unions and negations are allowed for expressing the rewriting. In the following example, we show that even without applying negation on the views, the *equivalent rewriting* can only be found if unions are allowed in the rewriting.

Example 10. Let Q and V_1, V_2 be the conjunctive queries as follows:

$$\begin{array}{l}Q: \ q(X,Y) \ \text{:-}\ a(X,Y).\\ V_1 : v_1(X,Y) \text{:-}\ a(X,Y), b(Y).\\ V_2 : v_2(X,Y) \text{:-}\ a(X,Y), \neg b(Y).\end{array}$$

There is an equivalent rewriting of Q using the union of V_1 and V_2. □

One important issue concerning answering queries using views is to decide whether a view is useful for the given query. It can be informally stated as the following [7]: a view can be useful for a query if the set of relations it mentions overlaps with that of the query, and it selects some of the attributes selected by the query. When a query with both positive and negative subgoals is considered, this condition remains true for the positive subgoals, but not for the negative subgoals any more. It can be shown in the following example that an *equivalent rewriting* can only be found by *negating* the views.

Example 11. Let Q and V_1, V_2 be the conjunctive queries as follows:

$$\begin{array}{l}Q: \ q(X,Y) \ \text{:-}\ p_0(X,Y), \neg p_1(X,X).\\ V_1 : v_1(X,Y) \text{:-}\ p_0(X,Y).\\ V_2 : v_2(X,Z) \text{:-}\ p_0(X,Y), p_1(X,Z).\end{array}$$

It can be shown that the rewriting Q' is an equivalent one:

$$Q' : q(X,Y) \text{:-}\ v_1(X,Y), \neg v_2(X,X).$$

□

One difficulty in choosing the views for the negated subgoals is to decide whether a view V can be useful for the query Q with its negated form. The next example differs from Example 11 only on V_2.

Example 12. Let Q and V_1, V_2 be the conjunctive queries as follows:

$$\begin{array}{l} Q:\ q(X,Y)\ \text{:-}\ p_0(X,Y), \neg p_1(X,X). \\ V_1: v_1(X,Y)\ \text{:-}\ p_0(X,Y). \\ V_2: v_2(X,Z)\ \text{:-}\ p_1(X,Z), p_2(X,Y). \end{array}$$

The rewriting Q' as follows is not even a *contained rewriting* for Q.

$$Q': q(X,Y)\ \text{:-}\ v_1(X,Y), \neg v_2(X,X).$$

When $Q' \cup V_1 \cup V_2$ is applied to the database $D = \{p_0(0,1), p_1(0,0)\}$, tuple $q(0,1)$ is obtained. However it is apparent that this is not the correct answer for Q. □

Informally speaking, in order to be useful for the query Q with its negated form, a view V should not contain *foreign* predicates, which do not appear in Q (cf. Example 12). Moreover, V has to contain at least one positive subgoal whose negated form appears in Q, and the attributes of the subgoal should not be projected out.

Related Work There is relatively less research concerning the problem of answering queries using views with safe negations appearing in the body of both queries and views. [4] provides an algorithm dealing with it. It is an extension of the *inverse rule* algorithm. However it is not clear whether the algorithm is complete w.r.t the problem of views based query answering.

6 Conclusion and Further Research

We have discussed the query containment problem for the conjunctive queries with safe negated subgoals. A new method for testing the containment of $CQ^{\neg}$s is given, comparing with the one in [14]. We have also shown that this algorithm can be naturally extended to the containment checking of unions of $CQ^{\neg}$s. At last, we have discussed the problem of answering queries using views when both queries and views have negated subgoals. Some motivating examples are given to show the cases which might not be encountered with pure conjunctive queries.

There are several interesting questions left for research. In [1] several categories of complexities for *answering queries using views* are given. However, with respect to negation, $Datalog^{\neg}$ is considered, and the setting of $CQ^{\neg}$ is not covered.

One interesting extension of our work is to consider the containment problem of CQs with *both* safe negation and arithmetic comparisons. To the best of our knowledge, no practical algorithm has been published to solve this problem. The complexity of this problem is still unknown as well.

Acknowledgments: We would like to thank Michael Benedikt for the valuable comments on this material.

References

1. S. Abiteboul and O. M. Duschka. Complexity of answering queries using materialized views. In *ACM Symp. on Principles of Database Systems (PODS)*, 1998.
2. A. K. Chandra and P. M. Merlin. Optimal implementations of conjunctive queries in relational data bases. In *ACM Symp. on Theory of Computing (STOC)*, pages 77–90. 1977.
3. O. M. Duschka and A. Y. Levy. Recursive plans for information gathering. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 778–784, 1997.
4. S. Flesca and S. Greco. Rewriting queries using views. In *TKDE*, 13(6), 2001.
5. D. Florescu, A. Levy, and D. Suciu. Query containment for conjunctive queries with regular expressions. In *ACM Symp. on Principles of Database Systems (PODS)*. 1998.
6. A. Gupta, Y. Sagiv, J. D. Ullman, and J. Widom. Constraint checking with partial information. In *ACM Symp. on Principles of Database Systems (PODS)*. 1994.
7. A. Halevy. Answering queries using views: A survey. In *VLDB Journal*, 10:4, pp. 270–294, 2001.
8. A. Klug. On conjunctive queries containing inequalities. In *Journal of the ACM* 35:1, pp. 146–160, 1988.
9. A. Y. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *ACM Symp. on Principles of Database Systems (PODS)*. 1995.
10. A. Levy and Y. Sagiv. Queries independent of updates. In *International Conference on Very Large Data Bases (VLDB)*, 1993.
11. A. Levy and D. Suciu. Deciding containment for queries with complex objects. In *ACM Symp. on Principles of Database Systems (PODS)*. ACM Press, 1997.
12. Y. Sagiv and M. Yannakakis. Equivalence among relational expressions with the union and difference operations. *Journal of the ACM, (4)*, 27(4):633–655, 1980.
13. J. Ullman. *Principles of Database and KnowledgeBase Systems, Volume II*. 1989.
14. J. D. Ullman. Information integration using logical views. In *International Conference on Database Theory (ICDT)*, 1997.
15. X. Zhang and Z. Meral Özsoyoglu. On efficient reasoning with implication constraints. In *DOOD*, pages 236–252, 1993.

APPENDIX

Algorithm Containment Checking(Q_1, Q_2)
Inputs: Q_1 and Q_2 are $CQ^{\neg}$s with the form as follows:

$$\begin{array}{l} Q_1 : h(\bar{X}) \text{ :- } p_1(\bar{X_1}), \ldots, p_n(\bar{X_n}), \neg s_1(\bar{Y_1}), \ldots, \neg s_m(\bar{Y_m}). \\ Q_2 : h(\bar{U}) \text{ :- } q_1(\bar{U_1}), \ldots, q_l(\bar{U_l}), \neg a_1(\bar{W_1}), \ldots, \neg a_k(\bar{W_k}). \end{array}$$

Begin
1. Set $\{p_1(\bar{X_1}), \ldots, p_n(\bar{X_n})\}$ as the root of a tree: c_0.
Let $\rho_1, \ldots, \rho_r$ be all the containment mappings from Q_2^+ to Q_1^+.
2. Generate r nodes $c_1, \ldots, c_r$ as children of root, and each node $c_i (1 \leq i \leq r)$ has a subtree with k children, in which each child $c_{i,j}$ with the form $\rho_i(a_j(\bar{W_j}))(1 \leq j \leq k)$.

```
3. Marking the nodes:
For i=1 to r do
  For j=1 to k do
    If $c_{i,j} \in \{p_1(\bar{X_1}, \ldots, p_n(\bar{X_n})\}$ Then mark $c_i$ as Terminal;
    If $c_{i,j} \in \{s_1(\bar{Y_1}), \ldots, s_m(ymbar)\}$ Then mark $c_{i,j}$ as Contained;
  EndFor
EndFor
4. Execute the containment checking:
If all nodes $c_1, \ldots, c_r$ are terminal nodes, Then return Not-contained;
For i=1 to r do
  If each $c_{i,j}(1 \leq j \leq k)$ is marked as Contained, Then return Contained;
EndFor
5. Continue expanding non-terminal nodes:
For i=1 to r do
  If $c_i$ is not Terminal Then
    For j = 1 to k do
      If $c_{i,j}$ is not Contained Then
        let $Q_1'$ be: $h$ :- $p_1(\bar{X_1}), \ldots, p_n(\bar{X_n}), c_{i,j}, \neg s_1(\bar{Y_1}), \ldots, \neg s_m(\bar{Y_m})$.;
        let $\rho_{i,j,1}, \ldots, \rho_{i,j,b_{i,j}}$ be the new containment mappings from $Q_2$ to $Q_1'$;
        Generate $b_{i,j}$ nodes with children in the same way as in Step 2;
        Mark the nodes in the same way as Step 3;
      EndIf
    EndFor
  EndIf
EndFor
6. Execute the containment checking:
For i=1 to r do
  If each $c_{i,j}(1 \leq j \leq k)$ either is marked as Contained
  or has a child whose children are all marked Contained
  Then return Contained;
  EndIf
  from each non-terminal node $c_i$, choose one child $c_{i,j}(1 \leq j \leq k)$;
  add all these $c_{i,j}$ to the body of $Q_1$, and let the new query be $Q_1''$;
  If each branch w.r.t. $Q_1''$ is marked as Terminal, Then return Not-contained;
EndFor
Let $r$ be all expanded nodes; Go to Step 5;
End
```