Answering Queries Using Views with Arithmetic Comparisons

Paper 87

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 92697-3425
chenli@ics.uci.edu

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

Abstract

We consider the problem of answering queries using views, where queries and views are conjunctive queries with arithmetic comparisons (CQACs) over dense orders. Previous work only considered limited variants of this problem, without giving a complete solution. We have developed an efficient algorithm to obtain maximally-contained rewritings (MCRs) for queries having left (or right) semi-interval-comparison predicates. For semi-interval queries we show that at least recursive datalog is necessary to find a maximally-contained solution, and identify cases where datalog is sufficient. Finally, we show that obtaining equivalent rewritings for CQAC’s is decidable.

1 Introduction

In many data-management applications, such as information integration [4, 11, 18, 19, 24, 32], data warehousing [29], web-site designs [15], and query optimization [9], the problem of answering queries using views [23] has taken on special significance. The problem can be stated as follows: given a query on a database schema and a set of views over the same schema, can we answer the query using only the answers to the views? Most of the recent work has addressed the problem when both queries and views are conjunctive. See [22] for a good survey.

In most commercial scenarios, users require the flexibility to pose queries using conjunctive queries along with arithmetic comparisons (e.g., $<$, $\leq$) between variables and constants that can take any value from a dense domain (e.g., real numbers). Similarly, views are also described using conjunctive queries with arithmetic comparisons. Although prior research has addressed the issue of containment of conjunctive queries with inequalities [20, 17], not many results are known on the problem of answering queries with inequality predicates using views.

When answering queries using views, we often need to find equivalent rewritings for a query [3, 23], or a maximally-contained rewriting [1, 26]. For instance, consider the following query $Q_1$, and views $v_1$ and $v_2$:

\begin{align*}
Q_1(A) &::= r(A), A \leq 4 \\
v_1(Y, Z) &::= r(X), s(Y, Z), Y \leq X, X \leq Z \\
v_2(Y, Z) &::= r(X), s(Y, Z), Y \leq X, X < Z
\end{align*}

The following query $P$ is a contained rewriting
(CR) of the query $Q_1$ using $v_1$:

$$P(A) :\triangleright v_1(A,A), A < 4$$

To see why, suppose we expand this query by replacing the view subgoal $v_1(A,A)$ by its definition. We get the expansion of $P$:

$$P(A) :\triangleright r(X), s(A,A), A \leq X, X \leq A, A < 4$$

Thus we can equate $X$ and $A$, and the expansion is contained in $Q_1$. Notice that the presence of the comparison predicates affects the existence of the rewriting. Although $v_1$ and $v_2$ differ only on their second inequalities, $v_2$ cannot be used to answer $Q_1$, since variable $X$ of $r(X)$ in $v_2$ is not exported in its head, hence constraint $A < 4$ cannot be enforced. On the other hand, $P$ is an equivalent rewriting (ER) of the following query:

$$Q_1'(A) :\triangleright r(A), s(A,A), A < 4$$

The following query and views show that we need at least the power of datalog to find a maximally-contained rewriting (MCR):

$$Q_2() :\triangleright e(X,Z), e(Z,Y), X > 5, Y < 7$$

$$v_1(X,Y) :\triangleright e(X,Z), e(Z,Y), Z > 5$$

$$v_2(X,Y) :\triangleright e(X,Z), e(Z,Y), Z < 7$$

$$v_3(X,Y) :\triangleright e(X,I_1), e(I_1,I_2), e(I_2,I_3), e(I_3,Y)$$

We can show that for any positive integer $k > 0$, the following is a CR:

$$P_k :\triangleright v_1(X,Z_1), v_3(Z_1,Z_2), v_3(Z_2,Z_3), \ldots, v_3(Z_{k-2}, Z_{k-1}), v_3(Z_{k-1}, Z_k), v_2(Z_k,Y)$$

In fact, the following recursive datalog program is an MCR of the query:

$$Q_2() :\triangleright v_1(X,W), T(W,Z), v_2(Z,Y)$$

$$T(W,Z) :\triangleright T(W,U), v_3(U,Z)$$

In this paper we study the problem of finding rewritings of a query using views when the query and views are conjunctive with comparisons (e.g., $<, \leq, >, \geq$), called CQAC queries. We consider both equivalent queries (ERs) and contained rewritings (CRs). We first review preliminary results in the literature on this problem (Section 2). We give a result on the existence of a single containment mapping between two CQAC queries, which is a stronger than that of [17, 20].

In Section 3, we study the decidability of the problem. We first prove the decidability for the ER case by extending a result in [28]. In particular, we can use finite unions of CQAC queries to represent ERs. We then address the problem of finding MCRs. We prove that if all view variables are distinguished, the problem is decidable, and finite unions of CQACs are expressive enough to represent MCRs.

In Section 4 we consider the problem of generating MCRs when queries are left-semi-interval (LSI) or right-semi-interval (RSI), and views have general comparisons. We present an algorithm to generate MCRs efficiently. We show the subtleties of finding MCRs in the presence of comparisons, which make our algorithm different from other algorithms in the literature.

In Section 5 we study the problem of finding MCRs when view variables can be nondistinguished. We first prove that at least recursive datalog is necessary for presenting MCRs. Then we consider a subcase, where datalog with semi-interval predicates is sufficient to express an MCR. We show that query containment in this case can be reduced to containment of a CQ in a datalog query. Based on this result, we develop an algorithm for finding MCRs. For this special case, we also obtain a result of independent interest, that the containment problem is in NP.

In the rest of the paper, we take the Open-World Assumption (OWA) [13]. That is, the views do not guarantee that they export all tuples in the world that satisfy their definition. Instead views export only a subset of such tuples. Due to space limitations, we do not provide all the proofs of the lemmas and theorems. Some results are explained in the complete version of this paper [2].
1.1 Related Work

Our problem is closely related to testing query containment. In [8] the problems of containment, minimization, and equivalence of CQs are shown NP-complete. In [20], it is shown that containment for CQs with inequality comparison predicates is in $\Pi_2^p$, whereas when only left or right semi-interval comparisons are used, the containment problem is in NP. In [33], containment for CQs with inequality comparison predicates is proven to be $\Pi_2^p$-complete. In [20], searching for other classes of CQs with inequality comparison predicates for which containment is in NP is stated as an open problem. [21] studies the computational complexity of the query containment problem of queries with disjunction ($\lor$). Containment of a CQ in a datalog query is shown to be EXPTIME-complete [12, 7]. Containment among recursive and nonrecursive datalog queries is also studied in [10].

The problem of answering queries using views has been studied in many papers. Table 1 summarizes the results presented in this paper, and other known results in the literature. [5, 6] deal with the problem of answering CQs over description logics using views expressed in description logics. On one hand, description logics are more expressive than CQACs. Therefore, the result that datalog may be necessary to rewrite queries using views that have nondistinguished variables cannot be used in our context. On the other hand, description logics allows only unary or binary predicates, whereas CQACs put no such restriction. Therefore, the result that it is possible to find a rewriting of a query using views expressed in description logics where all the view variables are distinguished does not imply the same result for CQACs.

2 Preliminaries

This section gives the notation used in the paper. Conjunctive queries with arithmetic comparisons: We focus on conjunctive queries and views with arithmetic comparisons of the following form:

$$h(\bar{X}) := g_1(\bar{X}_1), \ldots, g_n(\bar{X}_n), C_1, \ldots, C_m$$

The head $h(\bar{X})$ represents the results of the query. Each $g_i(\bar{X}_i)$ in the body is an ordinary subgoal. The variables $\bar{X}$ are called distinguished variables. Each $C_i$ is an arithmetic comparison in the form of “$A_1 \theta A_2$,” where $A_1$ and $A_2$ are variables or constants. If they are variables, they appear in the ordinary subgoals. Operator “$\theta$” is $<, \leq, =, >,$ or $\geq$. For sake of simplicity, we use “CQ” to represent a conjunctive query, “AC” for a arithmetic comparison, and “CQAC” for a conjunctive query with arithmetic comparisons. If a CQAC is written as “$Q = Q_0 + \beta$” it means that “$\beta$” is the ACs of $Q$, and “$Q_0$” is the query obtained by deleting the ACs from $Q$.

Query containment and equivalence: A query $Q_1$ is contained in a query $Q_2$, denoted $Q_1 \sqsubseteq Q_2$, if for any database $D$, the set of answers to $Q_1$ is a subset of the answers to $Q_2$. The two queries are equivalent, denoted $Q_1 \equiv Q_2$, if $Q_1 \sqsubseteq Q_2$ and $Q_2 \sqsubseteq Q_1$. Chandra and Merlin [8] showed that for two CQs $Q_1$ and $Q_2$, $Q_1 \sqsubseteq Q_2$ if and only if there is a containment mapping from $Q_2$ to $Q_1$. For containment between CQACs, [17, 20] gave the following theorem.\textsuperscript{1}

**Theorem 2.1** Let $Q_1 = Q_{10} + \beta_1$ and $Q_2 = Q_{20} + \beta_2$ be two CQACs. The ACs do not imply “$=$” restrictions. Let $\mu_1, \ldots, \mu_k$ be all the containment mappings from $Q_{10}$ to $Q_{20}$. Then $Q_2 \sqsubseteq Q_1$ if and only if:

$$\beta_2 \Rightarrow \mu_1(\beta_1) \lor \ldots \lor \mu_k(\beta_1)$$

i.e., $\beta_2$ logically implies (denoted “$\Rightarrow$”) the union of the images of $\beta_1$ under these mappings. \hfill $\square$

\textsuperscript{1}In [17], they assume that no variable appears twice among their ordinary subgoals, and no constant appears in their ordinary subgoals. Theorem 2.1 is a variation of their result.
Notice that the OR operation in the implication is critical, since there might not be a single mapping \( \mu_i \) from \( Q_{10} \) to \( Q_{20} \), such that \( \beta_2 \Rightarrow \mu_i(\beta_1) \).

**Answering queries using views:** The problem of answering queries using views [23] is as follows: given a query on a database schema and views over the same schema, can we answer the query using only the answers to the views? The following notations define the problem formally.

**Definition 2.1 (expansion)** The expansion of a query \( P \) on a set of views \( V \), denoted by \( P^{exp} \), is obtained from \( P \) by replacing all the views in \( P \) with their corresponding base relations. Nondistinguished variables in a view are replaced by fresh variables in \( P^{exp} \).

**Definition 2.2 (ERs and CRs)** Given a query \( Q \) and a view set \( V \), a query \( P \) is a contained rewriting ("CR" for short) of query \( Q \) using \( V \) if \( P \) uses only the views in \( V \), and \( P^{exp} \subseteq Q \). That is, \( P \) computes a partial answer to the query.\(^2\) We call \( P \) an equivalent rewriting ("ER" for short) of \( Q \) using \( V \) if \( P^{exp} \equiv Q \). \( P \) is a maximally-contained rewriting ("MCR" for short) of \( Q \) if (1) \( P \) is a CR of \( Q \), and (2) there is no CR \( P_1 \) of \( Q \) such that \( P_1^{exp} \) properly contains \( P^{exp} \).

\(^2\)In the rest of the paper, we use "rewritings" to mean "contained rewritings."

Several algorithms have been developed for answering queries using views, such as the bucket algorithm [24, 16], the inverse-rule algorithm [27, 14], and the algorithms in [3, 25, 26]. [23, 1] study the complexity of answering queries using views. In particular, it has been shown that the problem of finding a rewriting of a query using views is \( \mathcal{NP} \)-hard, even if the query and views are conjunctive. In this paper we study how to construct ERs and MCRs of a CQAC using CQAC views. Notice that a CR can be in a language different from that of the query and views. For instance, as shown by the second example in Section 1, we need at least datalog [31] to represent MCRs of a CQAC.

**Existence of a single containment mapping:** Theorem 2.1 is a fundamental result to solve our problem. However, the union operation in the logical implication makes the problem much harder than the case where the query and views are purely CQs. In particular, the following example shows that it is not clear how to construct the ACs in an MCR, since its ACs could be quite different from those in the query and views. For instance, consider the two CQACs shown in Figure 1.

\[
Q_1() \quad r(X_1, X_2), r(X_2, X_3), r(X_3, X_4), r(X_4, X_5), r(X_5, X_1), X_1 < X_2
\]

\[
Q_2() \quad r(X_1, X_2), r(X_2, X_3), r(X_3, X_4), r(X_4, X_5), r(X_5, X_1), X_1 < X_3
\]

These two queries are equivalent even though they have same ordinary subgoals but different ACs. Now consider the following two views that
are “decomposed” from $Q_2$:

$$v_1(X_1, X_3) \vdash r(X_1, X_2), r(X_2, X_3)$$
$$v_2(X_1, X_3) \vdash r(X_3, X_4), r(X_4, X_5), r(X_5, X_1)$$

The following is an ER of $Q_1$ using the views:

$$Q_1() \vdash v_1(X_1, X_3), v_2(X_1, X_3), X_1 < X_3$$

Notice that the AC $X_1 < X_3$ in this rewriting is different from the AC $X_1 < X_2$ in $Q_1$.

The problem becomes easier to solve in the case where a single containment mapping can prove containment between two CQACs. Such a case is where CQACs have ACs that are left or right semi-interval. A CQAC is called left semi-interval (“LSI” for short), if all its ACs are of the form $X < c$ or $X \leq c$, where $X$ is a variable, and $c$ is a constant. A right semi-interval CQAC (“RSI query” for short) is defined similarly. The following theorem is from [17, 20].

**Theorem 2.2** Let $Q_1 = Q_{10} + \beta_1$ and $Q_2 = Q_{20} + \beta_2$ be two LSI queries. Then $Q_2 \sqsubseteq Q_1$ if and only if there is a single containment mapping $\mu$ from $Q_{10}$ to $Q_{20}$, such that $\beta_2 \Rightarrow \mu(\beta_1)$.

Our first contribution this paper is to allow $\beta_2$ in the theorem to be general ACs.

**Theorem 2.3** Theorem 2.2 is true even if $\beta_2$ is general ACs.

---

The results on LSI queries in this paper are also true for RSI queries. We focus on LSI queries for sake of simplicity.

The theorem is a corollary of the following lemma.

**Lemma 2.1** Let $D_1, D_2, \ldots, D_n$ be conjunctions of LSI ACs, and $E$ be a conjunction of general ACs. Then $E \Rightarrow (D_1 \lor D_2 \lor \ldots \lor D_n)$ if and only if $E \Rightarrow D_i$ for some $i$, $1 \leq i \leq n$.

---

### 3 Decidability: ER and MCR

In this section we study decidability of finding equivalent rewritings (ERs) and maximally-contained rewritings (MCRs) for a CQAC query using CQAC views. For ERs, we give a decidability proof. For MCRs, we consider the case where all view variables are distinguished, and present the decidability result. The following decidability result is from [30].

**Theorem 3.1** It is decidable if there is a single-CQAC ER of a CQAC using CQAC views.

**Corollary 3.1** It is decidable if there is a union-of-CQAC ER of a CQAC using CQAC views.

Now we study the decidability of finding MCRs of queries, assuming all view variables are distinguished.

**Theorem 3.2** It is decidable if there is a union-of-CQAC MCR for a CQAC query using CQAC views, where all the view variables are distinguished.

**Proof:** Sketch. If there is a CR $P$ that is a CQAC, we can construct a set of CRs whose union contains $P$, and for each of them there is a single containment mapping that proves containment in the query. The latter is feasible because all view variables are distinguished, and AC’s can be enforced on them. A consequence of a single containment mapping is that the number of ordinary subgoals for each of those CRs is
bounded by the number of ordinary subgoals of the query.

In the rest of this paper we investigate the case of finding MCRs when some view variables may be nondistinguished.

4 An efficient algorithm for finding MCRs for LSI queries

In this section, we present an algorithm to generate MCRs for left-semi-interval (LSI) and right-semi-interval (RSI) queries using views with comparisons. The algorithm is based on Theorem 2.3, and proceeds finding useful mappings from query subgoals to view subgoals, as do most of the known algorithms in the literature [24, 26, 25]. These algorithms are designed mainly for the case where both the query and views are CQs. We restrict our attention in this section in pointing out the subtleties in our new algorithm because of the presence of arithmetic comparisons. The algorithm works in two steps. In step 1, we create buckets for query subgoals, and add useful mappings in the buckets. In step 2, we consider combinations of useful mappings and add comparisons to produce rewritings.

4.1 Constructing buckets

In step 1, for each query subgoal $g$ we construct a bucket that includes views that can be used in a rewriting to answer $g$. For each view $v$, for each of its subgoals, we try to construct a partial mapping $\mu$ from $g$ to this view subgoal. Let $\mu$ map a query argument $X$ in $g$ to a view argument $Y$. The view is put into the bucket of $g$ if $v$ can be used to map $g$ in a rewriting. We proceed with a few technical definitions and modules of the algorithm, which we illustrate with examples.

4.1.1 Exportable nondistinguished view variables

In the mapping, if $Y$ is a nondistinguished view variable, while $X$ appears in another query subgoal or a comparison predicate that cannot be mapped to view $v$, we might not be able to put the corresponding restriction on $Y$ in a candidate rewriting using this mapping. In Section 1, we showed an example where a nondistinguished view variable can be exported due to the comparison predicates in the views.

Definition 4.1 (exportable view variables)
A nondistinguished variable $X$ in a view $v$ is exportable if there is a head homomorphism $h$ on $v$, such that the inequalities in $h(v)$ imply that $X$ is equal to a distinguished variable in $v$. □

To discuss whether a nondistinguished view variable can be exported, we need to consider head homomorphisms of view head variables [26]. A head homomorphism of the head variables in a view is a partitioning of these variables, such that all the variables in each partition are equated. For instance, in the first example in Section 1, $\{\{Y, Z\}\}$ and $\{\{Y\}, \{Z\}\}$ are two head homomorphisms of the head variables of $v_2$.

Due to the comparison predicates in the views, a head homomorphism might not be sound. For example, we can have a valid head homomorphism that equates $Y$ and $Z$, only if the comparison predicates in the view do not entail $Y < Z$ or $Y > Z$.

Definition 4.2 (valid head homomorphisms) A head homomorphism is valid if all equations of variables in it are consistent with the comparison predicates in the view. □

4.1.2 Finding exportable variables

To find exportable nondistinguished variables in a view $v$, we use the comparison predicates in $v$ to construct its inequality graph [20], denoted
$G(v)$. That is, for each comparison predicate $A \theta B$, where $\theta$ is $<$ or $\leq$, we introduce two nodes labeled $A$ and $B$, and an edge labeled $\theta$ from $A$ to $B$. Clearly if there is a path between two nodes $A$ and $C$, we have $A < C$. If there is no $<\!$-labeled edge on any path between $A$ and $C$, then $A \leq C$.

**Definition 4.3** (leq-set) Given a nondistinguished variable $X$ in a view $v$, the leq-set (less-than-or-equal-to set) of $X$, denoted $S_\leq(v, X)$, includes all distinguished variables $Y$ of $v$ that satisfy the following conditions. There exists a path from $Y$ to $X$ in the inequality graph $G(v)$, and all edges on all paths from $Y$ to $X$ are labeled $\leq$. In addition, in all paths from $Y$ to $X$, there is no other distinguished variable except $Y$. $\Box$

Correspondingly, we define the geq-set (greater-than-or-equal-to set) of a variable $Y$, denoted $S_\geq(v, Y)$. We want to know which view variables are exportable. Not surprisingly, we have:

**Lemma 4.1** A nondistinguished variable $X$ in view $v$ is exportable iff both $S_\leq(v, X)$ and $S_\geq(v, Y)$ are nonempty. $\Box$

To export a nondistinguished variable $X$ in a view $v$, we can equate any pair of variables $(Y_1, Y_2)$, where $Y_1 \in S_\geq(v, X)$ and $Y_2 \in S_\leq(v, X)$. $X$ becomes exported since it is equal to $Y_1$ and $Y_2$, as are all variables in the path from $Y_1$ to $Y_2$.

**4.1.3 Equating a nondistinguished variable to another variable**

While constructing a partial mapping from a query subgoal $g$ to a subgoal in view $v$, a query variable $A$ might be mapped to two different view variables $X_1$ and $X_2$.

**Example 4.1** Consider query $Q(A) :- r(X_4, X_5)$, $s(X_1, X_2, X_3, X_6, X_7, X_8)$, $X_3 \leq X_5$, $X_5 \leq X_7$, $X_1 \leq X_4$, $X_8 \leq X_2$, $X_2 \leq X_4$, $X_4 \leq X_6$. Figure 2 shows the graph $G(v)$. In order to construct a mapping from query subgoal $r(A, A)$ to view subgoal $r(X_4, X_5)$, we need to equate $X_4$ and $X_5$, since both are the images of $A$. That is, we need $X_4 \leq X_5$ and $X_5 \leq X_4$. For the former, it can be satisfied if there is a path from $X_4$ to $X_5$ in graph $G(v)$. If such a path does not exist, we can have this inequality by equating a variable in $S_\geq(v, X_4)$ with a variable in $S_\leq(v, X_5)$. A similar argument holds for $X_5 \leq X_4$. Since neither inequality exists in the graph, we need to satisfy them by equating distinguished variables.

![Figure 2: The graph $G(v)$ in Example 4.1.](image)

Clearly we have $S_\geq(v, X_5) = \{X_3\}$, $S_\geq(v, X_5) = \{X_7\}$, $S_\geq(v, X_4) = \{X_1, X_2\}$, and $S_\geq(v, X_4) = \{X_6\}$. Note that $X_8$ is not in $S_\leq(v, X_4)$, because $X_8$ is “closer” to $X_4$ in the path from $X_8$ to $X_1$. The following are two most relaxing ways to equate variables to imply $X_4 = X_5$: (1) $X_6 = X_3, X_1 = X_7$, and (2) $X_6 = X_3, X_2 = X_7$. $\Box$

We construct a set $P$ of pairs of view variables that the algorithm needs to equate, so as to construct a valid partial mapping. Note that as in head homomorphisms, we have to consider only valid equating of variables. Namely, while equating variables to generate head homomorphisms for a view, some head homomorphisms make the comparison predicates in the view not satisfiable, and the view should be removed from the buckets. For instance, consider the following query and view.

**Example 4.1** Consider query $Q(A) :- r(A, A)$, and a view $v(X_1, X_2, X_3, X_6, X_7, X_8) :- Q(X, Y) :- p(X, Y), X < 3, Y > 5$

$v(A) :- p(A, A)$
We construct a mapping $\mu$ to map both $X$ and $Y$ to $A$. However, $\mu$ will map the query comparison predicates to $A < 3$ and $A > 5$, which is not satisfiable. Thus we cannot use this view to cover the query subgoal.

### 4.2 Satisfying the arithmetic comparisons in the query

In the second step of the algorithm, we consider combinations of views from the buckets to answer all query subgoals. Each combination represents a candidate rewriting, and we add comparison predicates to satisfy the comparison predicates in the query. Consider a query arithmetic comparison “$X \theta c$,” where $X$ is mapped to a view variable $Y$ in a partial mapping, and $\theta$ is $< \text{or} \leq$. The expansion of a rewriting should imply the image of this restriction, i.e., $Y \theta c$. If $Y$ is distinguished, we can just add “$Y \theta c$” to the rewriting. If $Y$ is nondistinguished, we cannot add any arithmetic comparison using $Y$, since $Y$ does not appear in the rewriting at all. However, there are two cases to satisfy this restriction. Case I: the arithmetic comparisons of the view $v$ imply “$Y \theta c$” by themselves. Case II: there is a path in $G(v)$ from $Y$ to a distinguished variable $Z$, so we can just add a arithmetic comparison “$Z \theta c$” to the rewriting to satisfy “$Y \theta c$.” For example, consider the following query and views.

$$
Q(A) :\ p(A), A < 3 \\
v_1(X_1) :\ p(X_1), X_1 < 3 \\
v_2(X_2, X_3) : p(X_1), r(X_2, X_3), X_2 \leq X_1, X_1 \leq X_3 \\
v_3(X_2, X_3) : p(X_1), r(X_2, X_3, X), X_2 \leq X_1, X_3 \leq X_1, X_1 \leq X_4
$$

While mapping the query subgoal $p(A)$ to the view subgoal $p(X_1)$ in view $v_1$, we have a partial mapping $\mu$ that maps variable $A$ to $X_1$. For a rewriting that uses this view, its expansion should entail $\mu(A < 3)$, i.e., $X_1 < 3$. The comparison predicate in $v_1$ belongs to case I, since its comparison predicate $X_1 < 3$ can satisfy this inequality. The comparison predicates in $v_2$ belong to case II. In particular, since $v_2$ has a comparison predicate $X_1 \leq X_3$, and $X_3$ is distinguished, thus we can add $X_3 \leq 3$ to satisfy the inequality $X_1 < 3$. The comparison predicates in $v_3$ do not belong to either case, thus $v_3$ cannot be used to cover the query subgoal. For lack of space, we describe the algorithm and prove correctness in the full version [2].

### 5 Recursive MCRs

In Section 3 we showed that if all view variables are distinguished, finite unions of CQACs are expressive enough for MCRs. In this section, we study the problem of finding MCRs in the case where some view variables are not distinguished. In Section 5.1 we prove that at least datalog is necessary for representing MCRs. In Section 5.2, we consider a subcase, where datalog with semi-interval predicates is sufficient to express an MCR. We show that query containment in this case can be reduced to containment of a CQ in a datalog query. Based on this result we develop an algorithm for finding MCRs in this case. Wlog, we restrict attention in this section to boolean queries.

#### 5.1 Datalog is necessary

We first prove that in the case where view variables can be nondistinguished, even when the view definitions contain semi-interval predicates, finite unions of CQACs are not sufficient to express an MCR, we need at least the power of datalog with semi-interval comparison predicates.

**Proposition 5.1** For the query $Q_2$ in the example of Section 1, there is no finite union of CQACs that contains all $P_k$’s and is contained in $Q_2$. \[\square\]

This example shows that at least datalog is necessary for representing MCRs. We use this example for simplicity. We have similar examples that cover the CWA case, and even the case when the views have no comparison predicates.

In the rest of this section, we consider a subcase where datalog with semi-interval pred-
icates is sufficient to express MCRs. The sub-case is when the query contains a single left-semi-interval inequality and several right-semi-interval inequalities, or it contains a single right-semi-interval inequality and several left-semi-interval inequalities. This kind of queries are called CQAC-SI queries. We assume that the views contain semi-interval inequalities, called CQAC-SI views. We show in this case, containment of a CQAC-SI query in a CQAC-SI H query can be reduced to containment of a CQ in a datalog query. Based on this result, we develop an algorithm for finding MCRs in this case. We first present preliminary results that are useful for the rest of this section.

**Lemma 5.1** Suppose each $e_i$ of $e_1, \ldots, e_n$ is a single SI inequality, and $b_1, \ldots, b_k$ are general inequalities. Then it holds:

1. $b_1 \land \ldots \land b_k \Rightarrow e_1 \lor \ldots \lor e_n$ iff either there is an $e_i$ such that $b_1 \land \ldots \land b_k \Rightarrow e_i$, or there are $e_i$ and $e_j$ such that $b_1 \land \ldots \land b_k \Rightarrow e_i \lor e_j$.

2. If moreover each $b_i$ is also a single SI inequality, then, $b_1 \land \ldots \land b_k \Rightarrow e_1 \lor \ldots \lor e_n$ iff either (a) there are $b_k$ and $e_i$ such that $b_k \Rightarrow e_i$, or (b) there are $e_i$ and $e_j$ such that $true \Rightarrow e_i \lor e_j$. \hfill $\Box$

The following lemma assumes the presence of SI comparison predicates only.

**Lemma 5.2** Let $Q_1 = Q_{10} + \beta_1$ and $Q_2 = Q_{20} + \beta_2$ be two CQACs, and $\mu_1, \ldots, \mu_n$ be containment mappings from $Q_{10}$ to $Q_{20}$.

1. Then these mappings minimally satisfy the entailment (“minimally” in the sense that dropping one mapping makes the entailment not true):

   $$\beta_2 \Rightarrow \mu_1(\beta_1) \lor \ldots \lor \mu_n(\beta_1)$$

   iff for every $i = 1, \ldots, n$ and for every inequality $e$ in $\beta_1$, either $\beta_2 \Rightarrow \mu_i(e)$, or there are $\mu_j$ and inequality $e'$ in $\beta_1$ such that $true \Rightarrow \mu_i(e) \lor \mu_j(e')$.

2. If moreover, in $\beta_1$, there is only a single RSI inequality, then there is a $\mu_1(\beta_1)$ with each inequality except one entailed by an inequality of $\beta_2$. \hfill $\Box$

### 5.2 Reducing CQAC-SI containment to datalog program containment

This subsection has two parts. In the first part, we give a procedure that takes as input any CQAC-SI query $Q_2$, and outputs a CQ (without arithmetic comparisons), denoted $Q_2^{CQ}$. Also, we give a procedure that takes as input any CQAC-SI query $Q_1$, and outputs a datalog program $Q_1^{datalog}$. In the second part, we prove that $Q_2$ is contained in $Q_1$ if and only if $Q_2^{CQ}$ is contained in $Q_1^{datalog}$.

Lemmas 5.2 and 5.1 are critical for this subsection. We will describe the construction assuming we have only strict inequality SIs. It extends easily to the case there are SIs with both $<, >, \leq$, and $\geq$. We describe the construction of $Q_1^{datalog}$ and $Q_2^{CQ}$ using the following example:

$$Q_1() \quad : \quad e(X,Y), e(Y,Z), X > 5, Z < 8$$

$$Q_2() \quad : \quad e(A,B), e(B,C), e(C,D), e(D,E), A > 6, E < 7$$

We can show $Q_1$ contains $Q_2$. We first construct $Q_2^{CQ}$ for $Q_2$ as follows. We introduce new unary EDB predicates $[31]$, two for each constant $c$ in $Q_2$, namely $U_{>c}$ and $U_{<c}$. We also introduce IDB predicates $I_{D}$ and $I_{<c}$ similarly. For each AC of the form $X \theta c$ (where $X$ is a variable), we refer to $I_{\theta c} (U_{\theta c}$ respectively as the associated $I$-predicate (U-predicate respectively). We introduce one recursive rule for $Q_2^{CQ}$. We copy the regular subgoals of $Q_2$. For each AC $X \theta c_i$ in $\beta_2$, we add a unary predicate subgoal $I_{\theta c_i}(X_1)$. We add a set of initialization rules, one for each pair $(U_{\theta c}, I_{\theta c})$. Note that $Q_2^{CQ}$ is equivalent to a CQ. The following is the $Q_2^{CQ}$ for $Q_2$ in our running example.
\[ Q_2^{CQ} \quad \vdash \quad e(A, B), e(B, C), e(C, D), e(D, E), \\
I_{\geq 6}(A), I_{<7}(E) \]
\[ I_{\geq 6}(A) \quad \vdash \quad \mu_{\geq 6}(A) \]
\[ I_{<7}(E) \quad \vdash \quad \mu_{<7}(E) \]

We construct a datalog program \( Q_1^{datalog} \) for \( Q_1 \):

\[ Q_1^{datalog} \quad \vdash \quad e(X, Y), e(Y, Z), I_{\geq 6}(X), I_{\leq 8}(Z) \]
\[ J_{\geq 8}(Z) \quad \vdash \quad e(X, Y), e(Y, Z), I_{\geq 5}(X) \] - mapping rule
\[ J_{\geq 5}(X) \quad \vdash \quad e(X, Y), e(Y, Z), I_{\leq 8}(Z) \] - mapping rule
\[ I_{\leq 8}(X) \quad \vdash \quad J_{\geq 5}(X) \] - coupling rule
\[ I_{\geq 5}(X) \quad \vdash \quad I_{\leq 8}(X) \] - coupling rule
\[ I_{\geq 6}(X) \quad \vdash \quad I_{<7}(X) \] - linking rule
\[ I_{\leq 8}(X) \quad \vdash \quad I_{<7}(X) \] - linking rule
\[ I_{\geq 6}(A) \quad \vdash \quad \mu_{\geq 6}(A) \] - initialization linking rule
\[ I_{<7}(E) \quad \vdash \quad \mu_{<7}(E) \] - initialization linking rule

In our running example, we want to show that \( Q_2^{CQ} \) is contained in \( Q_1^{datalog} \). We unfold the rules in \( Q_1^{datalog} \) and transform the program to the query rule:

\[ Q_1^{datalog} \quad \vdash \quad e(X, Y), e(Y, Z), e(X_1, Y_1), e(Y_1, X), \\
U_{\geq 6}(X_1), U_{<7}(Z) \]

This CQ maps on \( Q_2^{CQ} \), thus showing the containment.

**Theorem 5.1** Let \( Q_1 \) be a CQAC-SI query and \( Q_2 \) be a CQAC-SI query. Then \( Q_2 \subset Q_1 \) iff \( Q_2^{CQ} \subset Q_1^{datalog} \).

The following result is a consequence of this reduction.

**Theorem 5.2** The complexity of checking containment of a CQSI query in a CQSI1 query is in NP.

Our algorithm uses the algorithm in [14] to compute an MCR of a datalog program using CQ views, and transforms in a straightforward way the datalog MCR to an datalog \((<, \leq)\) MCR for the CQSI1 query.

**Future Work:** The decidability of finding an MCR of a query with comparison predicates using views with comparison predicates, especially, when all the view variables are not distinguished, needs to be investigated. We also need to design an efficient algorithm for obtaining ERIs of such queries using views especially, if we want the solution to scale.

**References**


J. Ullman. Personal communication.

