Answering Queries Using Views with Arithmetic Comparisons

Foto Afrati
Electrical and Computing Engineering
National Technical University
157 73 Athens, Greece
afrati@cs.ece.ntua.gr

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

Prasenjit Mitra
Department of Electrical Engineering
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 a novel algorithm to obtain maximally-contained rewritings (MCRs) for queries having left (or right) semi-interval-comparison predicates. For semi-interval queries, we show that the language of finite unions of CQAC rewritings is not sufficient to find a maximally-contained solution, and identify cases where datalog is sufficient. Finally, we show that it is decidable to obtain equivalent rewritings for CQAC queries.

1. INTRODUCTION

In many data-management applications, such as information integration [4, 11, 18, 20, 25, 26, 27, 33], data warehousing [31], web-site designs [15], and query optimization [9], the problem of answering queries using views [24] 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 [23] 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 [17, 21], 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, 24], or a *maximally-contained rewriting* (MCR) [1, 29].

PODS 2002, Madison, Wisconsin, USA

EXAMPLE 1.1. *Consider the following query Q_1, and views v_1 and v_2.*

$$
\begin{array}{ll}
Q_1(A) & \text{:- } r(A), A < 4 \\
v_1(Y,Z) & \text{:- } r(X), s(Y,Z), Y \leq X, X \leq Z \\
v_2(Y,Z) & \text{:- } r(X), s(Y,Z), Y \leq X, X < Z
\end{array}
$$

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

$$P(A) \text{ :- } v_1(A,A), A < 4$$

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:*

$$P(A) \text{ :- } r(X), s(A,A), A \leq X, X \leq A, A < 4$$

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 cannot be "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:*

$$Q_1'(A) \text{ :- } r(A), s(A,A), A < 4$$

□

EXAMPLE 1.2. *The following query and views show that there are cases where there is no maximally-contained rewriting that is a finite union of conjunctive queries with arithmetic comparisons.*

$$
\begin{array}{ll}
Q_2() & \text{:- } e(X,Z), e(Z,Y), X > 5, Y < 7 \\
v_1(X,Y) & \text{:- } e(X,Z), e(Z,Y), Z > 5 \\
v_2(X,Y) & \text{:- } e(X,Z), e(Z,Y), Z < 7 \\
v_3(X,Y) & \text{:- } e(X,Z), e(Z,Y)
\end{array}
$$

We can show that for any positive integer $k > 1$, the following is a CR:

$$
\begin{array}{ll}
P_k() \text{ :- } & v_1(X,W_1), v_3(W_1,W_2), v_3(W_2,W_3), \ldots, \\
& v_3(W_{k-1},W_k), v_2(W_k,Y)
\end{array}
$$

We can show that there is no finite union of conjunctive queries with arithmetic comparisons that contains all these P_k's. It is easy to observe, however, that the following recursive *datalog program is a CR of the query that contains all the P_k's:*

$$\begin{array}{ll} Q_2() & \text{:- } v_1(X,W), T(W,Z), v_2(Z,Y) \\ T(W,W) & \text{:- } \\ T(W,Z) & \text{:- } T(W,U), v_3(U,Z) \end{array}$$

□

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 (Theorem 2.3), which is stronger than the result in [17, 21], and it leads to the algorithm in Section 4.

In Section 3, we study the decidability of finding equivalent rewritings (ERs) and maximally-contained rewritings (MCRs) in the space of finite unions of CQACs. We first extend the decidability result in [34] to show that it is decidable to find ERs. For MCRs, in the case where all view variables are distinguished, we show that it is decidable whether there exists an MCR in the language of finite unions of CQACs.

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

In Section 5, we first give the following observation. When the query is conjunctive with semi-interval arithmetic comparisons (some of which are left and some right semi-interval comparisons), then there is no MCR in the language of finite unions of CQACs, even if the views have no comparison predicates. Then we consider a subcase where there exists an MCR in datalog with semi-interval predicates. We show that query containment in this case can be reduced to the containment of a CQ in a datalog query. Based on this result, we develop an algorithm for finding MCRs. We also obtain a result of independent interest, that is, the containment problem is in NP for this special case.

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 definitions. 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 [21], it is shown that containment of CQs with inequality comparison predicates is in Π_2^P, whereas when only left or right semi-interval comparisons are used, the containment problem is in NP. In [35], containment for CQs with inequality comparison predicates is proven to be Π_2^P-complete. In [21], searching for other classes of CQs with inequality comparison predicates for which containment is in NP is stated as an open problem. [22] studies the computational complexity of the query-containment problem of queries with ($\neq$). Containment of a CQ in a datalog query is shown to be EXPTIME-complete [12, 7]. Containment among recursive and nonrecursive datalog queries is studied in [10].

The problem of answering queries using views has been studied extensively. Table 1 summarizes the results presented in this paper, and other known results in the literature. (Please see Table 2 for the notations.) [36] studied conjunctive queries with arithmetic comparisons in the framework of finding whether a conjunctive query always produces an empty relation on database instances satisfying a given set of constraints. [5, 6] deal with the problem of answering CQs over description logics using views expressed in description logics. However, the difference in expressiveness of description logics and the fact that it allows only unary or binary predicates make their problem different from the one of conjunctive queries with arithmetic comparisons.

2. PRELIMINARIES

Conjunctive Queries with Comparisons

We focus on conjunctive queries and views with arithmetic comparisons of the following form:

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

The head $h(\bar{X})$ represents the results of the query. The variables $\bar{X}$ are called *distinguished* variables. Each $g_i(\bar{X}_1)$ in the body is an *ordinary subgoal*. 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 "θ" is $<$, $\leq$, $>$, or $\geq$. For sake of simplicity, we use "CQ" to represent a conjunctive query, "AC" for an arithmetic comparison, and "CQAC" for a conjunctive query with arithmetic comparisons. If a CQAC is written as

$$Q = Q_0 + \beta$$

It means that "β" 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, 21] gave the following theorem.[1]

THEOREM 2.1. *Let $Q_1 = Q_{10} + \beta_1$ and $Q_2 = Q_{20} + \beta_2$ be two CQACs, where each β_i ($i = 1, 2$) does 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:*

$$\beta_2 \Rightarrow \mu_1(\beta_1) \vee \ldots \vee \mu_k(\beta_1) \quad (1)$$

[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.

Query	Views	MCR	References
CQ	CQ	union of CQs	[16, 25, 28, 29]
Datalog	CQ	Datalog	[14]
CQ with LSI, RSI	CQ with LSI, RSI	union of CQs with LSI, RSI	[29]
CQ($\neq$)	CQ	co-NP-hard (data complexity)	[1]
CQ with comparisons	CQ with comparisons (all variables distinguished)	union of CQs with comparisons	Section 3
CQ with LSI, RSI	CQ with comparisons	union of CQs with LSI, RSI	Section 4
CQ with LSI1, RSI1	CQ with SI	Datalog (SI)	Section 5

Table 1: Results on MCRs

i.e., β_2 logically implies (denoted "$\Rightarrow$") the disjunction of the images of β_1 under these mappings. □

In the theorem we assume that the ACs do not imply "=" restrictions. That is, they do not imply any AC of the form $X = A$, where X is a variable and A is a variable or a constant. This assumption is not more restrictive than the result in [17], because we can do a preprocessing on any CQAC query, and identify any set of variables that are implied equal. Then we can replace these equal variables with one of them. For example, for query

$$q(X,Z) \text{ :- } e(X,Y), e(Y,Z), X \leq Y, Y \leq X$$

Its ACs imply $X = Y$. So the query can be equivalently written as:

$$q(X,Z) \text{ :- } e(X,X), e(X,Z)$$

Also notice that the OR operation ("$\vee$") in Equation 1 is critical, since there might not be a single mapping μ_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 [24] 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}.* □

Let Q be a query and V be a set of views.

1. A query P is a *contained rewriting* ("CR" for short) of query Q using V if P uses only the views in V, and $P^{exp} \sqsubseteq Q$. That is, P computes a partial answer to the query.[2]

2. A contained rewriting P of Q is an *equivalent rewriting* ("ER" for short) of Q using V if $P^{exp} \equiv Q$.

3. Given a language L, a query P is a *maximally-contained rewriting* ("MCR" for short) of Q using the views w.r.t. the language L if:

   (a) $P \in L$ is a contained rewriting of Q; and

   (b) For every contained rewriting $P_1 \in L$ of Q, $P_1 \sqsubseteq P$ as queries.

[2]In the rest of the paper, we use "rewritings" to mean "contained rewritings."

In this paper, we focus on the languages of finite unions of CQACs, and Datalog with arithmetic comparisons.

Several algorithms have been developed in the literature for answering queries using views, such as the bucket algorithm [25, 16], the inverse-rule algorithm [30, 14], and the algorithms in [3, 28, 29]. The complexity of answering queries using views is studied in [24, 1]. 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 query using CQAC views. Notice that a CR can be in a language different from that of the query and views.

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. Consider the following two CQACs, illustrated by Figure 1.

$$\begin{aligned} Q_1() &\text{ :- } r(X_1,X_2), r(X_2,X_3), r(X_3,X_4), \\ &\quad\; r(X_4,X_5), r(X_5,X_1), X_1 < X_2 \\ Q_2() &\text{ :- } r(X_1,X_2), r(X_2,X_3), r(X_3,X_4), \\ &\quad\; r(X_4,X_5), r(X_5,X_1), X_1 < X_3 \end{aligned}$$

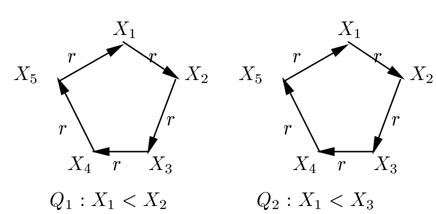

Figure 1: Two equivalent CQACs.

These two queries have same ordinary subgoals but different ACs. We can show that these two queries are in fact equivalent. Now consider the following two views that are "decomposed" from Q_2.

$$\begin{aligned} v_1(X_1,X_3) &\text{ :- } r(X_1,X_2), r(X_2,X_3) \\ v_2(X_1,X_3) &\text{ :- } r(X_3,X_4), r(X_4,X_5), r(X_5,X_1) \end{aligned}$$

The following is an ER of Q_1 using the views:

$$Q_1() \text{ :- } v_1(X_1,X_3), v_2(X_1,X_3), X_1 < X_3$$

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

Our problem becomes easier to solve if a single containment mapping can prove containment between two CQACs. One such 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 [21].[3]

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 μ from Q_{10} to Q_{20}, such that $\beta_2 \Rightarrow \mu(\beta_1)$.* □

Our first contribution in this paper is to extend this theorem by allowing β_2 to include any ACs.

THEOREM 2.3. *Let $Q_1 = Q_{10} + \beta_1$ be an LSI query, and $Q_2 = Q_{20} + \beta_2$ be a CQAC query. Then $Q_2 \sqsubseteq Q_1$ if and only if there is a* single *containment mapping μ from Q_{10} to Q_{20}, such that $\beta_2 \Rightarrow \mu(\beta_1)$.* □

The difference between these two theorems is that Theorem 2.3 allows Q_2 to have general ACs. This theorem is a corollary of the following lemma.

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

The proof is similar to that of Theorem 2.2 in [17]. The "If" part is straightforward. The "Only If" part uses the following result. For any constants a_1 and a_2 for a variable X_i, and constants b_1 and b_2 for a variable X_j, we have:

$$(a_1\ \theta\ b_1) \wedge (a_2\ \theta\ b_2) \Rightarrow \big(max(a_1, a_2)\ \theta\ max(b_1, b_2)\big)$$

Symbol	Meaning
CQ	Conjunctive Query
AC	Arithmetic Comparison
CQAC	Conjunctive Query with ACs
ER	Equivalent Rewriting
CR	Contained Rewriting
MCR	Maximally-Contained Rewriting
SI	Semi-interval ($X\theta c$, where $\theta \in \{<, \leq, >, \geq\}$)
LSI	Left-semi-interval ($X\theta c$, where $\theta \in \{<, \leq\}$)
RSI	Right-semi-interval ($X\theta c$, where $\theta \in \{>, \geq\}$)
SI1	one LSI AC + several RSI ACs; or one RSI AC + several LSI ACs

Table 2: Symbols.

Table 2 summarizes the symbols used in this paper.

3. DECIDABILITY RESULTS

In this section we study the decidability of finding ERs and MCRs of a CQAC query using CQAC views. For ERs, The following decidability result is from [34].

THEOREM 3.1. *It is decidable whether there is a single-CQAC ER of a CQAC query using CQAC views.* □

PROOF. The key of the proof is to compare a CQAC query Q with the expansion E of an ER P, which is a single CQAC. Suppose Q has s variables. We consider all 2^s orderings of the variables of Q that satisfy the comparisons in Q. For each total ordering, there must be a containment mapping from E to Q that preserves order. Associate with each variable of E a list of the 2^s variables that each of these mappings sends the variable of E to. We define two variables of E are "equivalent" if their lists are the same. Since lists are of length 2^s and each entry on the list has one of s values, there are at most s^{2^s} equivalence classes.

Design a new solution P' that equates all equivalent variables. P' is surely contained in P after expansion, since all we did was equate variables, thus restricting P and E. However, E', the expansion of P', has containment mappings to Q for all orderings, since all we did was equate variables that always went to the same variable of Q anyway. Thus Q is contained in P'. Since Q contains E, which contains E', it is also true that E' is contained in E. Thus, P' is another ER of Q. Thus, there is a doubly exponential bound on the number of subgoals in P', since there are only s^{2^s} variables, and a finite number of predicates each with finite arity. The conclusion is that we need to look only at some doubly-exponentially sized solutions. □

COROLLARY 3.1. *The problem of deciding whether there is an ER, which is a finite union of CQACs, of a CQAC query using CQAC views is decidable.* □

PROOF. (Sketch) We can extend the proof of Theorem 3.1 to the case where an ER is a union of CQACs. The main idea is to consider all total orderings of the variables in the query, and each corresponding query is contained in one of the CQACs in the ER. □

For MCRs, we first consider the case where all view variables are distinguished.

THEOREM 3.2. *Given a CQAC query Q, and a set V of CQAC views, such that all view variables are distinguished, consider the rewriting language L of finite unions of CQACs. It is decidable if there is an empty MCR (in L) of Q using V. In addition, such an MCR can be found in exponential time.* □

PROOF. (Sketch) If there is a CQAC CR P of Q using V, we can construct a set of CRs whose union contains P. In addition, 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 ACs 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. □

As illustrated by the query Q_2 in Section 1, if some view variables are not distinguished, we might need recursive datalog programs to represent MCRs. In Section 5 we will consider this case.

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

4. AN ALGORITHM FOR FINDING MCRS FOR LSI QUERIES

In this section, we present a novel algorithm for generating maximally-contained rewritings for left-semi-interval (LSI) and right-semi-interval (RSI) queries using views with general arithmetic comparisons. We assume that the ACs in each view and query cannot imply equalities. This assumption can be enforced by rewriting the query using a preprocessing step (See Section 2). In addition, the comparisons are consistent, i.e., there exists an assignment of constants to the variables, so that all the comparisons are true.

To the best of our knowledge, there exists no published algorithm that rewrites LSI queries using views that have arithmetic comparisons. Theorem 2.3 allows us to design such an algorithm. Our algorithm shares the same steps as the known algorithms in the literature [25, 28, 29]. That is, it first adds views to buckets representing query subgoals, and then constructs a rewriting by choosing views from the buckets. In this section, we first briefly review existing algorithms, and then outline our algorithm.

4.1 The MS Algorithms

We first review the MiniCon algorithm [29] and the Shared-Variable-Bucket algorithm [28], which share similar steps. Thus we will refer them as the MS algorithms heretofore. The MS algorithms have two steps. In the first step, they map each query subgoal to a view subgoal, and determine if there is a partial mapping from the query subgoal to the view subgoal. One important observation is the following. If a variable, say, X, appears multiple times in the query, and it maps to a nondistinguished view variable in a query rewriting, then all the query subgoals in which X occurs must map to subgoals in the body of the view. X is called a *shared variable.* [29] refers to the set of such query subgoals that have to be mapped to subgoals from one view (and the mapping information) as a *MiniCon Description* (MCD). We illustrate the MS algorithms using the following example.

Suppose we have three predicates, car, loc, and $color$. A tuple $car(c, d)$ means that a car c is sold by dealer d. A tuple $loc(d, p)$ means that a dealer d is located in place p. A tuple $color(a, b)$ means that a car a has color b. Consider the following query and views. (We use upper-case names for predicates and constants, and lower-case names for variables.)

$$
\begin{array}{lll}
Q: & q(C,L) & \text{:- } car(C,A), loc(A,L), color(C,red) \\
V_1: & v_1(X,Y) & \text{:- } car(X,D), loc(D,Y) \\
V_2: & v_2(W,Z) & \text{:- } color(W,Z)
\end{array}
$$

Consider the first query subgoal $g_1 = car(C, A)$ and view V_1. The variable C maps to the view variable X. The shared variable A maps to a nondistinguished view variable D. Therefore, the MS algorithms try to map all query subgoals in which A occurs to subgoals in view V_1. Variable A occurs in the first two query subgoals, which map to the two subgoals in V_1 using the mapping

$$\{C \to X, A \to D, L \to Y\}$$

Thus the algorithms create an MCD corresponding to query subgoals g_1 and g_2, where g_2 is the second query subgoal. Similarly, we have an MCD for the third query subgoal and view V_2. Table 3 shows the two MCDs created by the algorithms.

MCD	Query subgoals	View subgoals
1	$car(C,A), loc(A,L)$	$V_1 : car(X,D), loc(D,Y)$
2	$color(C,red)$	$V_2 : color(W,Z)$

Table 3: MCDs.

We then combine the MCDs to form the rewriting:

$$q(C,L) \text{ :- } v_1(C,L), v_2(C,red)$$

4.2 Algorithm: RewirteLSIQuery

We develop an algorithm, called "RewirteLSIQuery," that finds an MCR of an LSI query Q using views V, where the views in V can have general ACs. It shares the main steps of the MS algorithms. However, the presence of ACs introduces additional complexity to the problem. Our algorithm is novel in the following aspects.

1. Some nondistinguished variables can be "exported" and then treated as distinguished variables (Section 4.3).

2. There are several ways to satisfy the ACs in the query (Section 4.4).

Figure 2 shows the algorithm. In the next two subsections we will focus on the two novelties of our algorithm.

Algorithm RewriteLSIQuery
Input: • Q: an LSI query.
• V: a set of CQAC views.
Output: An MCR of Q using V.

Method:
Step 1: MCD Construction
for each subgoal g in Q
for each view $v \in V$, each subgoal g' in v:
1. Construct a mapping μ_i from g to g' considering exportable variables in g' as distinguished variables. (See Section 4.3.)

2. Construct a least restrictive set of head homomorphisms H, such that $\forall h_i \in H$, μ_i is a mapping from g to the expansion of g' in $h_i(v)$.

3. For each valid h_i, create an MCD for g.

Step 2: Rewriting Construction
Select a set of MCDs such that their query subgoals are disjoint, and they cover all the subgoals in Q.

For each chosen set of MCDs, form a CR P by joining their views, and equating the different distinguished view variables that one query variable maps to.

// Satisfy query's ACs. (See Section 4.4.)
For each AC "$X\ \theta\ c$" in Q:
1) If ACs in P already imply $\mu(X)\ \theta\ c$, then continue.
2) else, if possible, add ACs to P to satisfy $\mu(X)\ \theta\ c$.
3) else, mark this P as an invalid rewriting.

Return the union of all valid CRs.

Figure 2: Algorithm: RewriteLSIQuery.

4.3 Exporting Nondistinguished Variables

In the first step of the algorithm, we check if there is a mapping from a query subgoal to a view subgoal. In Example 1.1, we exported a nondistinguished view variable using the ACs in the view by equating it to another distinguished variable. Let an exported view variable X be equated to a distinguished view variable Y. The view supplies values for the distinguished view variable Y, and these values can be used to satisfy restrictions on the distinguished query variable that maps to X. Therefore, while constructing the mapping in the first step, we allow exportable nondistinguished view variables to be treated as distinguished view variables. In this subsection we discuss how we can export a nondistinguished view variable.

We first review the concept of *head homomorphisms* [29]. 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 Example 1.1, $\{\{Y, Z\}\}$ and $\{\{Y\}, \{Z\}\}$ are two head homomorphisms of the head variables of v_1. Notice that those variables not in the same partition still have the freedom to be equated.

DEFINITION 4.1. (**exportable view variables**) *We say 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 find exportable nondistinguished variables in a view v, we use the ACs in v to construct its *inequality graph* [21], denoted by $G(v)$. That is, for each comparison predicate "$A\ \theta\ B$," where θ is $<$ or $\leq$, we introduce two nodes labeled A and B, and an edge labeled θ from A to B. Clearly if there is a path between the two nodes A and C, we have $A \leq C$. If there is a $<$-labeled edge on any path between A and C, then $A < C$.

We need the following concepts to show how to export a nondistinguished view variable.

DEFINITION 4.2. (**leq-set**) *Let X be a nondistinguished variable in a view v. The* leq-set *(less-than-or-equal-to set) of X, denoted by $S_{\leq}(v, X)$, 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.* □

Correspondingly, we define the *geq-set* (greater-than-or-equal-to set) of a variable X, denoted by $S_{\geq}(v, X)$. The following lemma can help us decide if a view variable is exportable.

LEMMA 4.1. *A nondistinguished variable X in view v is exportable iff both $S_{\leq}(v, X)$ and $S_{\geq}(v, X)$ are nonempty.* □

EXAMPLE 4.1. *Consider the following view v.*

$$
\begin{aligned}
v(X_1, X_3, X_4, X_5, X_7, X_8) \text{ :- }\ & r(X_2, X_6),\\
& s(X_1, X_3, X_4, X_5, X_7, X_8),\\
& X_1 \leq X_2, X_2 \leq X_3, X_4 \leq X_5,\\
& X_5 \leq X_6, X_6 \leq X_7, X_8 \leq X_6
\end{aligned}
$$

Figure 3 shows the inequality graph $G(v)$. For the two nondistinguished variables, X_2 and X_6, we have:

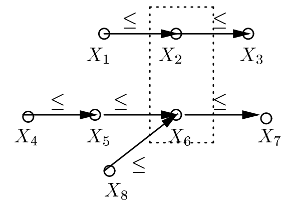

Figure 3: An inequality graph $G(v)$.

$$
S_{\leq}(v, X_2) = \{X_1\}, \qquad S_{\geq}(v, X_2) = \{X_3\}.
$$
$$
S_{\leq}(v, X_6) = \{X_5, X_8\}, \quad S_{\geq}(v, X_6) = \{X_7\}.
$$

Note that X_4 is not *in $S_{\leq}(v, X_6)$, because X_5 lies in the path from X_4 to X_6.* □

To export a nondistinguished variable X, we can just equate any pair of variables (Y_1, Y_2), where $Y_1 \in S_{\leq}(v, X)$ and $Y_2 \in S_{\geq}(v, X)$. Then X becomes equal to Y_1 and Y_2, as are all variables in the path from Y_1 to Y_2. In the example above, X_2 is exported by equating X_1 and X_3, corresponding to the following head homomorphism:

$$
h_1 = \{\{X_1, X_3\}, \{X_4\}, \{X_5\}, \{X_7\}, \{X_8\}\}
$$

Similarly, the following head homomorphisms can export variable X_6. They correspond to equating X_5 and X_7, and equating X_8 and X_7, respectively.

$$
h_2 = \{\{X_5, X_7\}, \{X_1\}, \{X_3\}, \{X_4\}, \{X_8\}\}
$$
$$
h_3 = \{\{X_8, X_7\}, \{X_1\}, \{X_3\}, \{X_4\}, \{X_5\}\}
$$

Notice that we could equate variables X_4 and X_7 to export X_6. However, the corresponding head homomorphism h is more restrictive than h_3, since h requires $X_4 = X_5$, while h_2 allows $X_4 \leq X_5$.

To export two nondistinguished variables, we need to combine their corresponding head homomorphisms. For instance, we can combine h_1 and h_2 above to form the following more restrictive head homomorphism:

$$
h_4 = \{\{X_1, X_3\}, \{X_5, X_7\}, \{X_4\}, \{X_8\}\}
$$

which exports and exports both variables X_2 and X_6.

If a query variable maps to two distinguished variables, a head homomorphism is constructed to equate the two distinguished variables. Similarly, if a distinguished variable is equated to an exportable variable, or two exportable variables are equated, the corresponding head homomorphisms are combined. For example, we can equate X_1 and X_5 in h_4 to have the following head homomorphism:

$$
h_5 = \{\{X_1, X_3, X_5, X_7\}, \{X_4\}, \{X_8\}\}
$$

which equates the two nondistinguished variables X_2 and X_6.

Here is another example to show how to combine head homomorphisms. Consider the following view:

$$
\begin{aligned}
v(X_1, X_3, X_4) \text{ :- }\ & r(X_2, X_5), s(X_1, X_3, X_4),\\
& X_1 \leq X_2, X_2 \leq X_3, X_1 \leq X_5,\\
& X_5 \leq X_3, X_4 \leq X_5
\end{aligned}
$$

We can export X_2 using the head homomorphism:

$$
h_6 = \{\{X_1, X_3\}, \{X_4\}\}
$$

and export X_5 using either h_6 or

$$h_7 = \{\{X_4, X_3\}, \{X_1\}\}$$

Now let us consider the case where both X_2 and X_5 need to be exported in an MCD. There are two ways to combine the head homomorphisms to export both variables: (1) h_6 and h_7; (2) h_6 and h_6. Clearly, the first combination gives a homomorphism that is more restrictive than the homomorphism generated in the second case. Therefore, we consider only $h_6(v)$ for insertion into the MCD. In general, to export several required nondistinguished variables, we consider all possible combinations of the homomorphisms that export the variables, and choose the least restrictive combinations.

4.4 Satisfying The Query's Comparisons

In the second step of the algorithm, we combine different MCDs to form a candidate CR, and then add comparisons to satisfy the ACs in the query. In this subsection, we discuss how to satisfy the query's arithmetic comparisons.

For each candidate rewriting P, consider an AC $A\ \theta\ c$ in the query. Suppose μ is the mapping from the query to the expansion of P, and μ maps A to a view variable X. Without loss of generality, assume θ is $<$ or $\leq$. The expansion of the corresponding CR using P should imply the image of this restriction, i.e., $X\ \theta\ c$. There are three possible cases to satisfy $X\ \theta\ c$.

1. The ACs in the expansion of P already imply $X\ \theta\ c$.

2. If X is a distinguished view variable, then we can just add an AC "$X\ \theta\ c$" to P.

3. X is a nondistinguished variable. There is a view v of X in the MCDs of P, whose inequality graph $G(v)$ has the following property. There is a path from X to a distinguished variable Y, such that the labels of all the edges in the path are either in $<$ or $\leq$. If the path contains a label $<$, we add $Y \leq c$. Otherwise, we add $Y\ \theta\ c$.

For example, consider the following query and views.

$$\begin{array}{lll}
Q(A) & \text{:-} & p(A), A < 3 \\
v_1(X_2) & \text{:-} & p(X_1), s(X_2), X_1 < 2 \\
v_2(X_1) & \text{:-} & p(X_1) \\
v_3(X_2, X_3) & \text{:-} & p(X_1), r(X_2, X_3, X_4), X_1 \leq X_3 \\
v_4(X_2, X_3) & \text{:-} & p(X_1), r(X_2, X_3, X_4), X_2 \leq X_1, \\
 & & X_3 \leq X_1, X_1 \leq X_4
\end{array}$$

While mapping the query subgoal $p(A)$ to the view subgoal $p(X_1)$ in view v_1, we have a partial mapping μ that maps variable A to X_1. For a rewriting that uses this view, its expansion should entail $\mu(A < 3)$, i.e., $X_1 < 3$. We satisfy this inequality using case (1), since the comparison predicate $X_1 < 2$ in v_1 implies $X_1 < 3$. The comparison predicate in v_2 belongs to case (2). Since variable X_1 is distinguished, we can add a comparison $X_1 < 3$.

The comparison predicate in v_3 belongs to case (3). In particular, since v_2 has a comparison predicate $X_1 \leq X_3$, and X_3 is distinguished, we can just add $X_3 \leq 3$ to satisfy the inequality $X_1 \leq 3$. The comparison predicates in v_4 do not belong to either case, thus v_4 cannot be used to cover the query subgoal.

Due to the comparison predicates in the views, the final comparisons in a rewriting might not be sound. For example, after combining MCDs for different query subgoals, we may have a rewriting with comparisons $Y < Z$ and $Y > Z$. In this case we can just discard this rewriting. Finally, we take the union of these rewritings from different combinations of MCDs, and form an MCR of the query using the views.

Now we illustrate how the complete algorithm works using an example. Consider the following query and views.

$$\begin{array}{lll}
Q(A) & \text{:-} & p(A, B), r(C), A > 5, B > 3 \\
v_1(X_1, X_2, X_3) & \text{:-} & p(X, Y), s(X_1, X_2, X_3), X_3 \leq X, \\
 & & X \leq X_1, X \leq X_2, X_3 \leq Y \\
v_2(U) & \text{:-} & r(U)
\end{array}$$

We construct the mapping from the first query subgoal to the first subgoal in v_1. Variable X needs to be exported, since it is used in $A > 5$. The following head homomorphisms can export X.

$$h_1 = \{\{X_1, X_3\}, \{X_2\}\},$$
$$h_2 = \{\{X_2, X_3\}, \{X_1\}\}.$$

So we put $v_1(A, X_2, A)$ and $v_1(X_1, A, A)$ (corresponding to h_1 and h_2, respectively) into the two MCDs for the first query subgoal. We also create an MCD for the second query subgoal and view v_2.

Then we consider the combinations of these MCDs to form CRs, and add necessary comparisons. Since B maps to view variable Y, its comparison $B > 3$ needs to be satisfied by Y. The ACs in view v_1 do not automatically satisfy $Y > 3$. However, since $Y \geq X_3$, we can use $X_3 > 3$ to satisfy it. Therefore, we generate an MCR that is a union of the following CRs:

$$P_1 : Q(A) \text{ :- } v_1(A, X_2, A), v_2(C), A > 5, A > 3$$
$$P_2 : Q(A) \text{ :- } v_1(X_1, A, A), v_2(C), A > 5, A > 3$$

Optionally, we might remove the AC $A > 3$ from the rewritings, since it is implied by $A > 5$. Note that the algorithm can be optimized by checking whether the query's comparison predicates are satisfied in Step 1 [2].

4.5 Correctness of RewriteLSIQuery

In this subsection, we prove the soundness and completeness of the RewriteLSIQ algorithm. The following theorem proves the soundness.

THEOREM 4.1. *Given an LSI query Q, and a set of CQAC views V, the RewriteLSIQuery algorithm generates a union of CQACs that is contained in Q.* □

PROOF. (Sketch) By Theorem 2.3, it suffices to show that for each generated contained rewriting P_i in the MCR, there exists a containment mapping μ from Q to P_i^{exp}, such that $\beta(P_i^{exp}) \Rightarrow \mu(\beta(Q))$. Here $\beta(Q)$ and $\beta(P_i^{exp})$ are the ACs of Q and P_i^{exp}, respectively.

If the algorithm maps a query variable to a nondistinguished view variable, then all occurrences of X map to the same variable in the view in the corresponding MCD. If X occurs in two MCDs, the algorithm only maps it to distinguished or exportable variables. In Step 2, these variables are equated. Thus, there exists a mapping μ from the query to the expansion of the rewriting. Note that P_i is constructed by choosing MCDs, whose sets of query subgoals are disjoint. That is, each query subgoal is covered by exactly one MCD. Thus combining the partial mappings from the query to the expansion of each MCD gives us the mapping μ.

It remains to show that $\beta(P_i^{exp}) \Rightarrow \mu(\beta(Q))$. By construction in step 2, for each comparison in the query, a contained rewriting generated by the algorithm can guarantee to satisfy this comparison. Therefore, $P_i^{exp} \sqsubseteq Q$. □

The following theorem proves the completeness.

THEOREM 4.2. *Let R be a single-CQAC rewriting of an LSI query Q using views V. There exists a containment rewriting R' in the union of CRs generated by the RewriteLSIQuery algorithm, such that $R \sqsubseteq R'$.* □

PROOF. (Sketch) Since R is a rewriting of Q, $R^{exp} \sqsubseteq Q$. Thus, there exists a mapping μ from Q to R^{exp}. Let μ map a query subgoal g to g' in R^{exp}. Assume g' is derived from $h(v)$ for some head homomorphism h on a view v. The algorithm generates the (partial) mapping from g to g' in the the MCD-construction step. The partial mapping is at least as relaxed as μ. The combination of such partial mappings constructs a mapping from the query to the expansion of a rewriting, such that the combined mapping is still at least as relaxed as μ. Furthermore, we can prove that the algorithm for exporting variables and satisfying the ACs in the query are sound and complete. They can guarantee that no valid partial mapping is missed in the MCD-construction step. The existence of the more relaxed mapping allows us to prove that RewriteLSIQuery will generate a CR R' that contains R. □

4.6 Efficiency of RewriteLSIQuery

The RewriteLSIQuery algorithm shares the same basic steps as the MS algorithms. We analyze the cost of the extra phases in the algorithm in the presence of ACs. To export a variable X of a view v, we need to compute the $S_{\leq}(v, X)$ and $S_{\geq}(v, X)$, which takes linear time in terms of the number of view variables. The number of head homomorphisms to export such a variable can be quadratic with respect to the number of view variables. Finding the set of the least restrictive head homomorphisms can be exponential with respect to the number of distinguished view variables. However, the number of head homomorphisms tend to be small in practice. Satisfying the ACs of the query variables can be done in time polynomial with respect to the number of view variables, since it involves finding paths to distinguished variables in the inequality graph.

In summary, the phases we added over those in the MS algorithms have either polynomial or exponential complexity. Since the worst-case complexity of the MS algorithms is exponential, our algorithm has the same complexity.

5. RECURSIVE MCRS

Example 1.2 in Section 1 showed if some view variables are not distinguished, we can have an MCR that is a recursive datalog program. We can show that if we only consider the language of finite unions of CQACs, the query Q_2 does not have an MCR. This observation is not surprising given the results in [1], even though it does not follow directly from the results in that paper.

PROPOSITION 5.1. *For the query Q_2 in Example 1.2, there is no finite union of CQAC rewritings that contains all the P_k's.* □

In this section, we consider the following case of answering a query Q using views V:

1. The query Q contains (possibly) a single left-semi-interval inequality and several right-semi-interval inequalities; or symmetrically, it contains a single right-semi-interval inequality and several left-semi-interval inequalities. (This kind of queries are called *CQAC-SI1* queries.)
2. The views are *CQAC-SI* views, i.e., they only contain semi-interval inequalities.

We believe that this case appears in many applications, and it is more general than those considered in previous work. We show that in this case, containment of a CQAC-SI query in a CQAC-SI1 query can be reduced to the containment of a CQ in a datalog query. Based on this result, we develop an algorithm for finding MCRs. Without loss of generality, in this section we restrict our attention to boolean queries. When we refer to "SI inequality" or simply to "inequality" in this section, we mean an inequality of the form $X\theta c$, where X is a variable, c is a constant, and θ is in $\{<, >, \leq, \geq\}$.

5.1 A Motivating Example

The following is a motivating example.

EXAMPLE 5.1. *Consider the following two queries.*

$$\begin{array}{lll} Q_1() & \text{:-} & e(X,Y), e(Y,Z), X > 5, Z < 8 \\ Q_2() & \text{:-} & e(A,B), e(B,C), e(C,D), e(D,E), \\ & & A > 6, E < 7 \end{array}$$

There are three containment mappings from the ordinary subgoals of Q_1 to the ordinary subgoals of Q_2:

$$\begin{array}{ll} \mu_1: & X \to A, Y \to B, Z \to C \\ \mu_2: & X \to B, Y \to C, Z \to D \\ \mu_3: & X \to C, Y \to D, Z \to E \end{array}$$

The following entailment holds.

$$\begin{array}{ll} A > 6 \wedge E < 7 \Rightarrow & \mu_1(X > 5 \wedge Z < 8) \\ & \vee \mu_3(X > 5 \wedge Z < 8) \end{array}$$

By Theorem 2.1, Q_2 is contained in Q_1. □

To look in details of this entailment, let us rewrite it as:

$$A > 6 \wedge E < 7 \Rightarrow (A > 5 \wedge C < 8) \vee (C > 5 \wedge E < 8)$$

It is equivalent to:

$$\begin{array}{ll} A > 6 \wedge E < 7 \Rightarrow & (A > 5 \vee C > 5) \wedge (A > 5 \vee E < 8) \\ & \wedge (C < 8 \vee C > 5) \wedge (C < 8 \vee E < 8) \end{array}$$

The latter holds because

1. $A > 6 \Rightarrow A > 5$, and $E < 7 \Rightarrow E < 8$;
2. $true \Rightarrow C < 8 \vee C > 5$.

In other words, the entailment of each conjunct in the right-hand side follows either

1. because a single inequality in the left-hand side implies a single inequality in the right-hand side (called a *direct implication*); or
2. because the disjunction of two inequalities in the right-hand side is true (called *coupling implication*).

It turns out that we only need to consider these two kinds of implications for SI inequalities.

LEMMA 5.1. *Let $b_1, \ldots, b_k$ and $e_1, \ldots, e_n$ be SI inequalities. Then*

$$b_1 \wedge \ldots \wedge b_k \Rightarrow e_1 \vee \ldots \vee e_n$$

*iff either (a) there are b_k and e_i such that $b_k \Rightarrow e_i$ (*direct implication*), or (b) there are e_i and e_j such that true $\Rightarrow e_i \vee e_j$ (*coupling implication*).* □

PROOF. Note that

$$b_1 \wedge \ldots \wedge b_k \Rightarrow e_1 \vee \ldots \vee e_n$$

is equivalent to the following boolean expression (denoted by ϕ):

$$\neg(b_1 \wedge \ldots \wedge b_k) \wedge (\neg e_1) \ldots \wedge (\neg e_n) \quad (2)$$

Since the boolean expression ϕ is true, $\neg\phi$ must be a contradiction. Note that $\neg\phi$ is a conjunction of SI inequalities, so its implication graph has a cycle. For example,

$$\begin{aligned} X < 5 \Rightarrow X < 6 \quad &\equiv \neg(X < 5) \vee (X < 6) \\ &\equiv \neg((X < 5) \wedge \neg(X < 6)) \\ &\equiv \neg((X < 5) \wedge (X \geq 6)) \end{aligned}$$

The implication graph of $(X < 5) \wedge (X \geq 6)$ has a $<$-edge from X to 5, a $\leq$-edge from 6 to X, and an obvious $<$-edge from 5 to 6. The presence of the "contradiction" cycle proves that $X < 5 \Rightarrow X < 6$ is true, and vice versa.

In general, we claim that if there is a cycle in the implication graph of $\neg\phi$, then there is an implication that uses at most two SI ACs, thus prove the lemma. To prove the claim, note that SI ACs introduce constants in the cycle. Moreover, if there are more than two SIs in the implication cycle, then there are at least two constants, with both paths between them in the cycle containing at least one variable. In this case, we can construct a cycle with at most two SI ACs, by "shortcutting" a pair of constants c_i and c_j. We choose them to eliminate any other constant on the cycle. For instance, suppose we have a contradiction cycle: $X < c_1 < Y < c_3 < W < c_2 < X$. Then we can shortcut c_1 and c_2, and get another contradiction cycle $X < c_1 < c_2 < X$.

After the shortcutting, we get a cycle with two SI ACs. Since we assumed that the left-hand side does not contain a contradiction, the two ACs cannot be both from the left-hand side of the original entailment. Therefore, there are two possible cases about the two SI ACs in the cycle.

1. One AC is from the left-hand side and one is from the right-hand side of the original entailment.
2. Both are from the right-hand side of the original entailment.

These cases correspond to case (a) and case (b) in the lemma, respectively. The presence of the cycle proves the corresponding implication. □

5.2 Reducing CQAC-SI Containment to Datalog Containment

Let P be a CQAC-SI query. Suppose we want to check whether P is contained in the query Q_1 in Example 5.1. We define a new query, denoted P^{CQ}, based on the inequalities of Q_1 as follows.

1. It retains the ordinary subgoals of P.
2. Consider each inequality of the form $X\theta c_1$ in P. For each constant c that appears in Q_1, if $(X \ \theta \ c_1) \Rightarrow (X \ \theta \ c)$, then add unary predicate $U_{\theta c}(X)$.
3. It drops other inequalities in P.

For example, for the Q_2 in Example 5.1, we have:

$$Q_2^{CQ}() \text{ :- } e(A,B), e(B,C), e(C,D), e(D,E), U_{>5}(A), U_{<8}(E)$$

Strictly speaking, the above is Q_2^{CQ} *with respect to* Q_1. However, in most cases, once the corresponding Q_1 is clearly stated, we will refer to it as simply Q_2^{CQ} without mentioning Q_1 every time. Notice that for any variable X in P and any constant c in Q_1, $U_{\theta c}(X)$ is in P^{CQ} iff $X\theta c$ is implied by the AC predicates in P.

The key observation now is that we can reduce checking for an implication as in Example 5.1 to checking for the existence of a containment mapping between conjunctive queries. To illustrate this observation, we prove the following proposition. (It is purely a motivating proposition, but its proof presents the main technical points involved in the proof of Theorem 5.1, the main result of this section, and hence some intuition.)

PROPOSITION 5.2. *Let P be a CQAC-SI query. Suppose we know that either P is not contained in the query Q_1 in Example 5.1, or exactly two mappings are sufficient and necessary to prove containment of P in Q_1. Then P is contained in Q_1 iff P^{CQ} is contained in the following query:*

$$\begin{aligned} Q_1'() \text{ :- } \ & e(X_1,Y_1), e(Y_1,X_2), e(X_2,Y_2), e(Y_2,Z_1), \\ & U_{>5}(X_1), U_{<8}(Z_1) \end{aligned}$$

□

PROOF. Let $P = Q_0 + \beta$, and $Q_1 = Q_{10} + \beta_1$.

Consider the "if" direction. If P^{CQ} is contained in Q_1', then there is a containment mapping μ from the subgoals of Q_1' into the subgoals of P^{CQ}.

Consider the following two mappings from the ordinary subgoals of Q_1 into the subgoals of Q_1':

$$\begin{aligned} \mu_1': \quad & X \rightarrow X_1, Y \rightarrow Y_1, Z \rightarrow X_2 \\ \mu_2': \quad & X \rightarrow X_2, Y \rightarrow Y_2, Z \rightarrow Z_1. \end{aligned}$$

Construct the mappings μ_1 and μ_2 from the ordinary subgoals of Q_1 into the subgoals of P^{CQ} (and hence into the ordinary subgoals of P) to be μ_1' followed by μ, and μ_2' followed by μ, respectively. We claim that μ_1 and μ_2 prove the containment of P in Q_1. That is, we need to prove the following implication.

$$\beta \Rightarrow (\mu_1(X) > 5 \wedge \mu_1(Z) < 8) \vee (\mu_2(X) > 5 \wedge \mu_2(Z) < 8)$$

To prove the satisfaction of the implication, we argue on how the right-hand side is implied due to direct implications (a) and due to coupling (b):

(a) (Direct implication) μ maps the unary predicate's $U_{>5}$ variable X_1 into a variable $\mu(X_1)$, such that $\mu(X_1) > 5$ is implied by the ACs in P (i.e., it is implied by β). However, $\mu_1(X) = \mu(X_1)$ is true by construction. Hence $\mu_1(X) > 5$ is implied by the ACs in P. The same claim holds for $U_{<8}$, hence $\mu_2(Z) < 8$ is implied by the ACs in P.

(b) (Coupling) By construction, however, we have $\mu_1(Z) = \mu_2(X)$.

Summarizing (a) and (b), we have: $\beta \Rightarrow \mu_1(X) > 5$ and $\beta \Rightarrow \mu_2(Z) < 8$ and $\mu_1(Z) = \mu_2(X)$. Therefore β implies the right-hand side of the above implication.

The "only if" direction. If two mappings are necessary and sufficient to prove containment of P in Q_1, then we show that there is a containment mapping from the subgoals of Q'_1 into the subgoals of P^{CQ}. Let μ_1 and μ_2 be the two mappings. We know that:

$$\beta \Rightarrow (\mu_1(X) > 5 \wedge \mu_1(Z) < 8) \vee (\mu_2(X) > 5 \wedge \mu_2(Z) < 8) \quad (3)$$

Assuming that two mappings are necessary and sufficient to prove containment of P in Q_1, we shall show in the rest of the proof the following: If Entailment 3 is satisfied, then there is a containment mapping from the subgoals of Q'_1 into the subgoals of P^{CQ}. To prove this claim, we argue as follows on the possibilities of satisfaction of the right-hand side of Entailment 3.

First, we use the fact that exactly two mappings are needed. Hence one mapping will not do. Therefore, (i) one of the $\mu_1(X) > 5$ and $\mu_1(Z) < 8$ is not implied by the left-hand side. (Otherwise mapping μ_1 would suffice for the implication to be satisfied.) (ii) one of the $\mu_2(X) > 5$ and $\mu_2(Z) < 8$ is not implied by the left-hand side. (Otherwise, mapping μ_2 would suffice for the implication to be satisfied.) It leaves the possibility of either (a) $\mu_1(X) > 5$ is not implied, and $\mu_2(Z) < 8$ is not implied; or (b) $\mu_1(Z) < 8$ is not implied, and $\mu_2(X) > 5$ is not implied. (All other possibilities would never end up in satisfying the implication.) In case (a), for the implication to be satisfied, the following should hold: $\mu_1(X) = \mu_2(Z)$. In case case (b), similarly, $\mu_1(Z) = \mu_2(X)$ should hold.

Now, it is easy to verify that, in both cases, μ_1 and μ_2 can be viewed as a mapping μ that maps:

$$e(X_1, Y_1), e(Y_1, X_2), e(X_2, Y_2), e(Y_2, Z_1)$$

into the ordinary subgoals of P, such that $X_1 > 5$ and $Z_1 < 8$ are implied by the ACs in P. As $\beta \Rightarrow \mu(X_1) > 5$ and $\beta \Rightarrow \mu(Z_1) < 8$ are encoded by the unary predicates $U_{>5}(\mu(X_1)), U_{<8}(\mu(Z_1))$ in P^{CQ}, mapping μ also maps the unary predicates. Thus it is a containment mapping from the subgoals of Q'_1 into the subgoals of P^{CQ}. $\square$

In the proof of the "only if" direction, we reasoned on the possibilities of satisfaction of Entailment 3. As the right-hand side of the implication was concrete, we could reason by complete enumeration of the possibilities. In the general case, however, we need the following lemma.

LEMMA 5.2. *Let $Q_1 = Q_{10} + \beta_1$ be a CQAC-SI1 query, and $Q_2 = Q_{20} + \beta_2$ be a CQAC-SI query. Suppose $\mu_1, \ldots, \mu_n$ are containment mappings from Q_{10} to Q_{20}, such that*

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

Then there is a $\mu_i(\beta_1)$ with each inequality, except possibly one inequality, directly implied by an inequality of β_2. $\square$

In Example 5.1, it turns out that the trick that works for whenever exactly two mappings prove containment can be generalized for any number of mappings by means of a Datalog program:

$$\begin{array}{ll} Q_1^{datalog}() & :\!-\ e(X,Y), e(Y,Z), I_{>5}(X), U_{<8}(Z) \\ I_{>5}(Z) & :\!-\ e(X,Y), e(Y,Z), I_{>5}(X) \\ I_{>5}(Z) & :\!-\ U_{>5}(X) \end{array}$$

The unary predicates in the program are used to encode conditions for the satisfaction of the implication. The EDB unary predicate $U_{<8}$ (and similarly the $U_{>5}$) encodes direct implications. That is, for any containment mapping, the variable in the argument of $U_{<8}$ maps on a variable X, such that $X < 8$ is implied by the ACs in Q_2. The IDB unary predicates encode both coupling, and (in the initialization rule) direct implication. That is, whenever we unfold the recursive rule into $I_{>5}$ in the body, we ensure that we get an additional mapping that offers a coupling inequality to $X < 8$.

5.3 Constructing Queries $Q_1^{datalog}$ and Q_2^{CQ}

In this subsection, we give a procedure that takes as input any CQAC-SI1 query Q_1, and outputs a datalog program $Q_1^{datalog}$. We also give a procedure that takes as input any CQAC-SI query Q_2, and outputs a CQ Q_2^{CQ} without arithmetic comparisons. Finally, we prove that Q_2 is contained in Q_1 if and only if Q_2^{CQ} is contained in $Q_1^{datalog}$.

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 (same as 5.1):

$$\begin{array}{lll} Q_1() & :\!- & e(X,Y), e(Y,Z), X > 5, Z < 8 \\ Q_2() & :\!- & e(A,B), e(B,C), e(C,D), e(D,E), \\ & & A > 6, E < 7 \end{array}$$

We first construct Q_2^{CQ} for Q_2 as follows. We introduce new unary IDB predicates [32], two for each constant c in Q_2, namely $I_{>c}$ and $I_{<c}$. We also introduce EDB predicates $U_{>c}$ and $U_{<c}$, two for each constant c in Q_1. 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 $X_i\theta c_i$ in β_2, we add a unary predicate subgoal $I_{\theta c_i}(X_i)$. We add a set of initialization *linking* rules that link the inequalities in Q_2 to the inequalities in Q_1. For each pair of constants c and c', where c is a constant of Q_1 and c' a constant of Q_2 such that $X\theta c$ is implied by $X\theta c'$, and there is an inequality $X'\theta c$ in Q_1, we add an initialization linking rule:

$$I_{\theta c'}(X) : -U_{\theta c}(X)$$

Note that Q_2^{CQ} is equivalent to a CQ. The following is the Q_2^{CQ} for Q_2 in our running example. (Note this Q_2^{CQ} is a different representation of the conjunctive query Q^{CQ} in Section 5.2.)

$$\begin{array}{lll} Q_2^{CQ} & :\!- & e(A,B), e(B,C), e(C,D), e(D,E), \\ & & I_{>6}(A), I_{<7}(E) \\ I_{>6}(A) & :\!- & U_{>5}(A) \\ I_{<7}(E) & :\!- & U_{<8}(E) \end{array}$$

We construct the following datalog program $Q_1^{datalog}$ for Q_1.

$$
\begin{array}{ll}
Q_1^{datalog} & \text{:- } e(X,Y), e(Y,Z), I_{>5}(X), I_{<8}(Z) \\
J_{<8}(Z) & \text{:- } e(X,Y), e(Y,Z), I_{>5}(X) \text{ – mapping rule} \\
J_{>5}(X) & \text{:- } e(X,Y), e(Y,Z), I_{<8}(Z) \text{ – mapping rule} \\
I_{<8}(X) & \text{:- } J_{>5}(X) \text{ – coupling rule} \\
I_{>5}(X) & \text{:- } J_{<8}(X) \text{ – coupling rule} \\
I_{>5}(A) & \text{:- } U_{>5}(A) \text{ – initialization rule} \\
I_{<8}(E) & \text{:- } U_{<8}(E) \text{ – initialization rule}
\end{array}
$$

We give the details of the construction of $Q_1^{datalog}$. We first construct a single *query rule* (the first one in the program). We then construct three kinds of rules: *mapping rules*, *coupling rules*, and *initialization* rules. We introduce new unary IDB predicates, two pairs for each constant c in Q_1, namely ($I_{>c}$, $I_{<c}$) and ($J_{>c}$, $J_{<c}$). For each pair of one inequality $X\theta c$ and one IDB predicate atom $I_{\theta c}(X)$ ($J_{\theta c}(X)$ respectively), we refer to each other as the *associated I-atom* (*associated J-atom* respectively) or the *associated AC*.

The query rule copies in its body all subgoals of Q_1 and replaces each AC of Q_1 by its associated I-atom. We get one mapping rule for each single inequality e in Q_1. The body is a copy of the body of the query rule, only that the I-atom associated to e is deleted. The head is associated to the J-atom associated to e. For every pair of constants $c_1 \leq c_2$ contained in Q_1, we construct two coupling rules. One rule is $I_{<c_2}(X)$:- $J_{>c_1}(X)$, and the other is $I_{>c_1}(X)$:- $J_{<c_2}(X)$.

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:

$$
\begin{array}{ll}
Q_1^{datalog} \text{ :- } & e(X,Y), e(Y,Z), e(X_1,Y_1), e(Y_1,X), \\
& U_{>5}(X_1), U_{<8}(Z)
\end{array}
$$

This CQ maps to Q_2^{CQ}, thus shows the containment.

THEOREM 5.1. *Let Q_1 be a CQAC-SI1 query and Q_2 be a CQAC-SI query. Then $Q_2 \sqsubseteq Q_1$ iff $Q_2^{CQ} \sqsubseteq Q_1^{datalog}$.* □

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.* □

5.4 An Algorithm for Finding MCRs

We use the result in the previous subsection to construct an algorithm that produces an MCR for a CQAC-SI1 query using CQAC-SI views, where some view variables can be nondistinguished The derived MCR is a datalog program with semi-interval comparisons. The result in this subsection is an immediate consequence of the results in the previous subsection and the results in [14].

The prove the correctness of the algorithm, we need the following lemma.

LEMMA 5.3. *Let Q be a CQAC-SI query. Let P be a CR of Q using views. Then there is a union of CRs of Q that contains P, and uses only SI ACs.* □

The following theorem proves the correctness of the algorithm.

THEOREM 5.3. *Each datalog query with comparison predicates that is contained in Q is contained in the MCR produced by the algorithm.* □

Input: • Q: a CQAC-SI1 query.
• V: a set of CQAC-SI views.
Output: An MCR of Q using V.

Method:
1. Construct the datalog query $Q^{datalog}$ for Q.
2. For each view $v_i \in V$, construct a new view v_i^{CQ}.
3. For each unary predicate $U_{\theta c}$, construct a view $u_{\theta c}$ as: $u_{\theta c}(X)$:- $U_{\theta c}(X)$.
4. Find an MCR P (using the results in [14]) for the datalog query $Q^{datalog}$ using the views v_i^{CQ}'s and $u_{\theta c}$'s.
5. Replace each v_i^{CQ} by v_i, and each $u_{\theta c}(X)$ in P by AC $X\theta c$.
6. Return the result as an MCR of Q.

Figure 4: Constructing an MCR of a CQAC-SI1 query using CQAC-SI views.

PROOF. Suppose P_D is a datalog query with comparisons, and P_D is contained in Q. For each CQ P_D^1 with comparison predicates that is produced by P_D, we shall prove that P_D^1 is contained in P. Because of Lemma 5.3, it suffices to prove that for any CQ P_1 with SI predicates that is contained in Q, it is contained in the MCR (denoted P) generated by the algorithm. As P_1 is contained in Q, P_1^{exp} is contained in Q, hence (by Theorem 5.1) $(P_1^{exp})^{CQ}$ is contained in $Q^{datalog}$. By construction of the MCR in step 4 in the algorithm, $(P_1)^{CQ}$ is contained in the MCR of $Q^{datalog}$. Since the translation of this MCR into the MCR of Q using the original views maps the unary predicates of the former one-to-one to the associated comparisons of the latter, P_1 is contained in the MCR of Q produced by the algorithm. □

6. CONCLUSION

In this paper we studied the problem of answering a query using views in the presence of arithmetic comparisons. A conjunctive query with arithmetic comparisons is called a CQAC query. We first gave the decidable results in the case of finding equivalent rewritings, and the case of finding maximally-contained rewritings (MCRs) when all view variables are distinguished. Then we developed a novel algorithm, called RewirteLSIQuery, for finding an MCR (in the space of finite unions of CQACs) for a query using views, where the query has left-semi-interval comparisons only (or right-semi-interval comparisons only). This algorithm shares the basic steps as two existing algorithms in the literature. It is novel in dealing with nondistinguished view variables, and satisfying comparisons in the query.

In the case where some view variables are nondistinguished, we showed that an MCR can be a recursive datalog program. Then we studied one case of the problem, where the views contain semi-interval comparisons (called CQAC-SI views), and the query contains (possibly) a single left-semi-interval inequality and several right-semi-interval inequalities and its symmetric case (called a CQAC-SI1 query). In this case we showed that there is an MCR w.r.t. the language of datalog with semi-interval predicates, whereas there is no MCR w.r.t. the language of finite unions of CQACs. We developed an algorithm for finding an MCR in this case.

The results in this paper indicate that the presence of arithmetic comparisons in the query introduce most of the complications in the problem of finding rewritings. In that respect, it might not be too hard to extend the results in

Section 5 to the case views contain any comparison predicates. Whereas even for views without comparison predicates, when the query contains any kind of semi-interval predicates, it seems difficult to find a satisfactory solution to the problem of finding maximally-contained rewritings. For equivalent rewritings, it will be interesting to explore efficient heuristics. See [19] for more discussions on finding rewritings of queries using views.

Acknowledgments

We thank Jeff Ullman for providing the results in Theorem 3.1. We also thank the anonymous reviewers for their valuable comments on the draft of this paper.

7. REFERENCES

[1] S. Abiteboul and O. M. Duschka. Complexity of answering queries using materialized views. In *PODS*, pages 254–263, 1998.

[2] F. Afrati, C. Li, and P. Mitra. Answering queries using views with arithmetic comparisons (full version). *Technical report, Information and Computer Science, UC Irvine*, 2002.

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

[4] R. J. Bayardo Jr. et al. Infosleuth: Semantic integration of information in open and dynamic environments (experience paper). In *SIGMOD*, pages 195–206, 1997.

[5] C. Beeri, A. Y. Levy, and M.-C. Rousset. Rewriting queries using views in description logics. In *PODS*, pages 99–108. ACM Press, 1997.

[6] D. Calvanese, G. D. Giacomo, and M. Lenzerini. Answering queries using views over description logics knowledge bases. In *PODS*, pages 386–391, July - August 2000.

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

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

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

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

[11] S. S. Chawathe et al. The TSIMMIS project: Integration of heterogeneous information sources. *IPSJ*, pages 7–18, 1994.

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

[13] O. M. Duschka. Query planning and optimization in information integration. *Ph.D. Thesis, Computer Science Dept., Stanford Univ.*, 1997.

[14] O. M. Duschka and M. R. Genesereth. Answering recursive queries using views. In *PODS*, pages 109–116, 1997.

[15] D. Florescu, A. Levy, D. Suciu, and K. Yagoub. Optimization of run-time management of data intensive web-sites. In *Proc. of VLDB*, pages 627–638, 1999.

[16] G. Grahne and A. O. Mendelzon. Tableau techniques for querying information sources through global schemas. In *ICDT*, pages 332–347, 1999.

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

[18] L. M. Haas, D. Kossmann, E. L. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *Proc. of VLDB*, pages 276–285, 1997.

[19] A. Y. Halevy. Theory of answering queries using views. *SIGMOD Record*, 29(4):40–47, 2000.

[20] Z. Ives, D. Florescu, M. Friedman, A. Levy, and D. Weld. An adaptive query execution engine for data integration. In *SIGMOD*, pages 299–310, 1999.

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

[22] 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.

[23] A. Levy. Answering queries using views: A survey. *Technical report, Computer Science Dept., Washington Univ.*, 2000.

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

[25] A. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of VLDB*, pages 251–262, 1996.

[26] C. Li. Query processing and optimization in information-integration systems. *Ph.D. Thesis, Computer Science Dept., Stanford Univ.*, 2001.

[27] C. Li, M. Bawa, and J. D. Ullman. Minimizing view sets without losing query-answering power. In *ICDT*, pages 99–113, 2001.

[28] P. Mitra. An algorithm for answering queries efficiently using views. In *Proceedings of the Australasian Database Conference*, 2001.

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

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

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

[32] J. D. Ullman. *Principles of Database and Knowledge-base Systems, Volumes II: The New Technologies.* Computer Science Press, New York, 1989.

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

[34] J. D. Ullman. Personal communication, 2001.

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

[36] X. Zhang and M. Ozsoyoglu. Implication and referential constraints: A new formal reasoning. In *Knowledge and Data Engineering*, volume 9(6), pages 894–910, 1997.