Answering Queries Using Limited External Query Processors*

(Extended Abstract)

Alon Y. Levy
AT&T Research
levy@research.att.com

Anand Rajaraman
Stanford University
anand@cs.stanford.edu

Jeffrey D. Ullman
Stanford University
ullman@cs.stanford.edu

Abstract

When answering queries using external information sources, their contents can be described by views. To answer a query, we must rewrite it using the set of views presented by the sources. When the external information sources also have the ability to answer some (perhaps limited) sets of queries that require performing operations on their data, the set of views presented by the source may be infinite (albeit encoded in some finite fashion). Previous work on answering queries using views has only considered the case where the set of views is finite. In order to exploit the ability of information sources to answer more complex queries, we consider the problem of answering conjunctive queries using infinite sets of views. Our first result is that an infinite set of views can be partitioned into a finite number of equivalence classes, such that picking one view from every nonempty class is sufficient to determine whether the query can be answered using the views. Second, we show how to compute the set of equivalence classes for sets of views encoded by a datalog program. Furthermore, we extend our results to the case when the query and the views use the built-in predicates $<$, $\leq$, $=$, and $\neq$, and they are interpreted over a dense domain.

Finally, we extend our results to conjunctive queries and views with the built-in predicates $<$, $\leq$, and $=$ interpreted over the integers. In doing so we present a result of independent interest, namely, an algorithm to minimize such queries.

*The work of the second and third authors was supported by NSF grant IRI–92–23405, ARO grant DAAH04–95–1–0192, and USAF contract F33615–93–1–1339.

1 Introduction

The problem of answering queries using views has recently received considerable attention because of its applications in Global Information Systems [LSK95, RSU95, LRO96], Mobile Computing [BI94, HSW94], view adaptation [GMR95], maintaining physical data independence [TSI94] and speeding up query processing [YL87, CKPS95].

The problem arises naturally in systems that provide access to multiple heterogeneous information sources using *mediators* (e.g., TSIMMIS [CGMH+94, PGGMU95], HERMES [SAB+95], Information Manifold [LSK95, LRO96], SIMS [ACHK94], Internet Softbot [EW94]). In such systems, information sources are treated as limited external query processors that can answer some but not all possible queries over their data. The query capabilities of each information source are represented by a set $\mathcal{V}$ of *views* over some set of base relations. A query is solved by a program that uses the views to obtain answers from the information sources. For the solution to be correct, it must produce the same answer as the query for any set of tuples in the base relations.

Most previous work on answering queries using views (e.g., [YL87, LMSS95, RSU95, CKPS95]) assumes the set $\mathcal{V}$ of views to be finite. Papakonstantinou et al. [PGGMU95] considers the possibility of an infinite set of views supported by one source and shows that infinite sets of views naturally model sources with the ability to answer complex queries about their contents. In such cases it is important to be able to exploit the local processing power of the information source in order to reduce the amount of data transmitted over the network. In this paper we consider the view-rewriting problem when there is an infinite set of views, and the query and the views are conjunctive. Our first result shows that *any* infinite set of views can be partitioned into a finite set of equivalence classes, so that all views in an equivalence class are equivalent with respect to rewritings of the particular query Q we wish to answer. If we can pick one representative view from each nonempty equivalence class (preferably, the shortest), we can then use known techniques (such as those in [LMSS95]) to find a rewriting.

Several useful families of views can be encoded by using a Datalog program. We show that for such sets of views it is possible to compute which equivalence classes they have and pick one from every nonempty class. We then show that the set of equivalence classes remains finite and computable even when the views are allowed to contain the built-in comparison predicates $<$, $\leq$, $=$, and $\neq$, and they are interpreted over a dense domain. We show that the problem is decidable also when the query has built-in predicates, provided that all the subgoals of built-in predicates are *local*. A subgoal g of a built-in predicate is local if there is some other subgoal in the query of a non built-in predicate that contains all the variables of g.

Finally, we extend our results to queries and views with the built-in predicates $<$, $\leq$, and $=$ interpreted over the integers. In doing so we present a result of independent interest, namely, an algorithm to minimize such conjunctive queries. The previously known minimization algorithm [Klu88] requires that the built-in predicates be interpreted over a dense domain.

Our algorithm for rewriting queries using an infinite set of views assumes that the arity of the views is bounded. In order to handle some cases in which the arity is not bounded, we consider the view rewriting problem in the presence of *parameterized view definitions*. A parameterized view is a conjunctive query that contains *placeholders* in argument positions in the body of the view, in addition to variables and constants. The parameterized view represents the set of all view definitions obtained by assigning a constant to each place holder. Using parameterized view definitions we can encode the capabilities of information sources that may accept an unbounded number of inputs as part of the query, while still maintaining a bounded arity for the views computed by the sources.

The case in which the set of views is encoded by a datalog program was considered in [PGGMU95]. Our work generalizes [PGGMU95] in two ways. First, we allow the rewriting of the query using the views to contain multiple views from the infinite set $\mathcal{V}$, whereas [PGGMU95] try to find a *single* view from $\mathcal{V}$ to which additional post-processing can be applied to make it as "close" to the original query as possible. Second, we consider the case in which the rewriting is required to be equivalent to the query, whereas [PGGMU95] requires only that the solution contain the query.

2 Preliminaries

In our discussion we refer to the relations used in the query as the *database relations*. We consider conjunctive queries, which may in addition contain the *built-in comparison predicates* $=$, $\neq$, $<$ and $\leq$. We use $V, V_1, \ldots, V_m$ to denote views that are defined on the database relations. Views are also defined by conjunctive queries.

Definition 2.1 A query Q' is a *rewriting* of Q using $\mathcal{V}$ if

- Q and Q' are equivalent (i.e., produce the same answer for any instantiation of the database relations), and
- The predicates used in Q' are only names of views in $\mathcal{V}$ or built-in comparison predicates. $\square$

The *view-rewriting* problem is the following: Given a query Q and a set of views $\mathcal{V}$, does there exist a rewriting of Q using $\mathcal{V}$?

Given a rewriting Q', the *expansion* of Q' is the query in which the subgoals of the views are replaced by their respective definitions. That is, if $V(\bar{X})$ is a subgoal of Q' we replace it by the result of unifying $V(\bar{X})$ with the head of the definition of V. Variables of V that do not appear in the head of V are replaced by variables occurring nowhere else in the expansion. The expansion of Q' (which is equivalent to Q') is used to check that Q' is equivalent to Q.

Example 2.1 Suppose *red* and *blue* are relations that represent the set of red and blue arcs in a directed graph. Informally, the view U_1 describes the set of red cycles of length 2 and V_1 describes the set of blue arcs that are followed by a red arc. Query Q_1 asks for the set of blue arcs whose destination node lies on a red cycle of length 2. Query Q_2 asks for the set of blue arcs whose destination node lies on a red cycle of length 3.

$$
\begin{array}{llll}
U_1: & u_1(X) & :- & red(X,Y) \ \& \ red(Y,X) \\
V_1: & v_1(X,Y) & :- & blue(X,Y) \ \& \ red(Y,Z_1) \\
Q_1: & q_1(A,B) & :- & blue(A,B) \ \& \ red(B,C) \ \& \\
 & & & red(C,B) \\
Q_2: & q_2(A,B) & :- & blue(A,B) \ \& \ red(B,C) \ \& \\
 & & & red(C,D) \ \& \ red(D,B)
\end{array}
$$

The query Q_1 can be rewritten using views U_1 and V_1 as follows:

$$Q_1' : q_1(A,B) \quad :- \quad v_1(A,B) \ \& \ u_1(B)$$

To see that Q_1' is a rewriting of Q_1 we consider its expansion:

$$
\begin{array}{llll}
Q_1'': & q_1(A,B) & :- & blue(A,B) \ \& \ red(B,Z_1) \ \& \\
 & & & red(B,Z_2) \ \& \ red(Z_2,B)
\end{array}
$$

and we show that there is a *containment mapping* [CM77] from Q_1 to Q_1'' and from Q_1'' to Q_1. It can also be shown that there is no rewriting of Q_2 using the views U_1 and V_1. □

Levy et al. [LMSS95] show that the problem of finding a rewriting of a query using a finite set of views is NP-complete when neither the query nor the views contain built-in subgoals. In some applications we are faced with the view-rewriting problem where the set of views $\mathcal{V}$ may be infinite. For example, suppose we are answering queries using only external sources of information. If the external sources have, in addition to data, the ability to answer complex queries about their data, then their capabilities are described by a set of views they can compute (e.g., [PGGMU95, LRO95]), which is typically an infinite set.

Example 2.2 Consider the example of red and blue arcs from Example 2.1 and define the views U_i and V_i as follows:

$$
\begin{array}{llll}
U_i: & u_i(X) & :- & red(X,Y_1) \ \& \ red(Y_1,Y_2) \ \& \ \ldots \\
 & & & \& \ red(Y_i,X) \\
V_i: & v_i(X,Y) & :- & blue(X,Y) \ \& \ red(Y,Z_1) \ \& \ \ldots \\
 & & & \& \ red(Z_{i-1},Z_i)
\end{array}
$$

Informally, U_i describes red cycles of length $i+1$ and V_i describes blue arcs that are followed by red paths of length i. Suppose an external source can answer all the queries defined by the views $U_1, U_2, \ldots$ and $V_1, V_2, \ldots$. Then Q_1 can be rewritten using U_1 and V_1 exactly as in Example 2.1. Moreover, we can also rewrite query Q_2 as follows:

$$Q_2' : q_2(A,B) :- v_1(A,B) \ \& \ u_2(B)$$

□

In this paper we consider the view-rewriting problem when $\mathcal{V}$ may be infinite, as in Example 2.2. We assume that the arity of any view in $\mathcal{V}$ is bounded by some constant m. If the arity of view V is k, we assume unless otherwise mentioned that the head variables of a view V are $Z_1, \ldots, Z_k$ in that order. In particular, note that two views of the same arity have identical variables in the head, and no view has repeated variables in the head.

A *variable mapping* is a function that maps a set of variables $\bar{X}$ to a set $\bar{Y}$. A set of variable mappings $\psi_1, \ldots, \psi_n$, whose domains are different but perhaps overlapping, are said to be *pairwise consistent* if there does not exist a variable X and integers i and j such that $\psi_i(X) \neq \psi_j(X)$. If they are consistent, we define their *union mapping,* ψ, to be the variable mapping whose domain is the union of the domains of $\psi_1, \ldots, \psi_n$, and $\psi(X) = Y$ if there exists some i, $1 \leq i \leq n$ such that $\psi_i(X) = Y$. Variable mappings are defined to be the identity mapping on predicate symbols and constants, and so we can apply a variable mapping to a subgoal in a query with the obvious meaning. A *complete mapping* ψ from a query Q_1 to Q is a variable mapping from the variables of Q_1 to the variables of Q, such that for every subgoal g in the body of Q_1, $\psi(g)$ is in the body of Q. A *containment mapping* from Q_1 to Q is a complete mapping from Q_1 to Q that also maps the head of Q_1 to the head of Q. Finally, a *partial mapping* from a query Q_1 to Q is a variable mapping from some subset of the variables of Q_1 to the variables of Q.

3 Rewriting Queries Using Infinitely Many Views

We begin by considering the case in which the query Q and the views $\mathcal{V}$ are conjunctive queries and do not have built-in subgoals. Built-in subgoals are considered in Section 7. Without loss of generality, we can assume that if $V_1 \in \mathcal{V}$, and V_2 is the result of equating two variables that appear in the head of V_1 and removing one copy of the duplicate variable from the head, then $V_2 \in \mathcal{V}$. Our first result shows that any infinite set of views $\mathcal{V}$ can be partitioned into a *finite* number of equivalence classes, such that determining whether there is a rewriting of a query Q using $\mathcal{V}$ can be done by checking only one view from every nonempty equivalence class. The importance of this is result is that given a set $\mathcal{V}$, we can focus on the problem of computing these equivalence classes, which may have a more direct solution. In Section 4 we show how to compute the equivalence classes for a set of views encoded by a Datalog program.

The intuition behind our solution is the following. Suppose Q' is a conjunctive query using views in $\mathcal{V}$, and we want to check whether Q' is a rewriting of Q. To do so we consider the query expansion Q'' of Q', and show that Q is equivalent to Q''. This step requires that we find containment mappings from Q to Q'' and vice versa. If two views V_1 and V_2 can participate in the same ways in containment mappings between Q and Q'' then they will be interchangeable in rewritings of Q. In what follows we formally define an equivalence relation among views that captures this intuition.

Example 3.1 In Example 2.2, we can show that for any i, the following is a rewriting of Q_1:

$$Q_1^i : q_1(A, B) :- v_i(A, B) \ \& \ u_1(B)$$

Informally, whenever there is a red cycle, we can construct red paths of arbitrary length by repeatedly cycling through the arcs in the cycle. The views V_i and V_j differ only in the length of the red path that follows a blue arc. Therefore, the subgoals $v_i(A, B)$, for $i = 1, 2, \ldots$ behave in exactly the same way with respect to rewritings of query Q_1. $\Box$

The equivalence relation between views is defined by considering *signatures* of variable mappings between the views and the query.

Definition 3.1 Let V be a view of arity m and Q be a query. Let ψ be a complete mapping from the variables of V to the variables of the query Q. The Q-signature of ψ, denoted by $sig_Q(\psi)$ is $\{(Z_1, T_1), \ldots, (Z_m, T_m)\}$, where $T_i = \psi(Z_i)$.

Definition 3.2 Let Q be a query and V be a view, and let ψ be a partial mapping from the variables of Q to the variables of V. The V-signature of ψ, denoted by $sig_V(\psi)$ is the tuple $(\psi_{head}, \bar{g})$, where:

- ψ_{head} is the restriction of ψ to variables of Q that are mapped to head variables of V,
- $\bar{g}$ is the subset of the subgoals of Q that are mapped by ψ to subgoals in the body of V.

Definition 3.3 Let V_1 and V_2 be two views in $\mathcal{V}$ of the same arity, and let Q be a query. We say that V_1 and V_2 are rewriting-equivalent w.r.t. the query Q, denoted by $V_1 \equiv_Q V_2$, if

C1. $\{sig_Q(\psi) \mid \psi \in Maps(V_1)\} = \{sig_Q(\psi) \mid \psi \in Maps(V_2)\}$, where $Maps(V_i)$ denotes the set of complete mappings from V_i to Q, and

C2. $\{sig_V(\psi) \mid \psi \in Pmaps(V_1)\} = \{sig_V(\psi) \mid \psi \in Pmaps(V_2)\}$, where $Pmaps(V_i)$ denotes the set of partial mappings from the variables of Q to the variables of V_i.

Since there are only a finite number of signatures for a given query Q, Definition 3.3 partitions the set $\mathcal{V}$ to a finite number of equivalence classes. Intuitively, C1 guarantees that V_1 and V_2 can be interchanged without changing the possible variable mappings from a rewriting to Q, while C2 guarantees that the same holds for variables mappings from Q to a rewriting.

Example 3.2 For even i, the unique complete mapping from V_i to Q_1 is:

$$X \to A,\ Y \to B, Z_1 \to C,\ Z_2 \to B,\ \ldots,$$

$$Z_{i-1} \to C,\ Z_i \to B$$

For odd i, the the unique complete mapping from V_i to Q_1 is:

$$X \to A,\ Y \to B,\ Z_1 \to C,\ Z_2 \to B,\ \ldots$$

$$Z_{i-1} \to B,\ Z_i \to C$$

That is, odd red arcs get mapped to the subgoal $red(B, C)$ and even red arcs get mapped to the subgoal $red(C, B)$. The Q-signature is identical in each case: $\{(X, A), (Y, B)\}$. Thus each V_i has the same set of Q-signatures.

Consider the V-signatures for V_2. For clarity we show only maximal variable mappings, that is, variable mappings that cannot be extended to any more variables. Let us number the subgoals of Q_1 1, 2, and 3, from left to right. There are four maximal partial mappings from Q_1 to V_2:

1. $\phi_1 = \{A \rightarrow X, B \rightarrow Y, C \rightarrow Z_1\}$, mapping subgoals $\{1, 2\}$.
2. $\phi_2 = \{C \rightarrow Y, B \rightarrow Z_1\}$, mapping subgoal 3.
3. $\phi_3 = \{B \rightarrow Z_1, C \rightarrow Z_2\}$, mapping subgoal 2.
4. $\phi_4 = \{C \rightarrow Z_1, B \rightarrow Z_2\}$, mapping subgoal 3.

The set of V-signatures of V_1 is therefore

$$\{(\{(A,X),(B,Y)\},\{1,2\}),\ (\{(C,Y)\},\{3\}),$$
$$(\phi,\{2\}),\ (\phi,\{3\})\}$$

For V_i, $i > 2$, the number of maximal partial mappings increases but the set of V-signatures remains the same. Since all the V_i, $i > 2$ have the same set of Q-signatures and V-signatures, they are rewriting-equivalent with respect to query Q_1. V_1 alone has a different set of V-signatures from the rest of the V_i's and is in an equivalence class by itself. □

The following theorem shows that views that are rewriting equivalent can be interchanged in a rewriting of a query Q using $\mathcal{V}$. Therefore, it is enough to consider only one representative view from every rewriting-equivalence class in order to decide whether there is a rewriting of Q using $\mathcal{V}$.

Theorem 3.1 *Suppose*

$$Q_1 : q_1(\bar{X}) : -V_1(\bar{X}_1)\,\&\,\ldots\,\&\,V_n(\bar{X}_n)$$

and

$$Q_2 : q_2(\bar{X}) : -U_1(\bar{X}_1)\,\&\,\ldots\,\&\,U_n(\bar{X}_n)$$

are conjunctive queries using views in $\mathcal{V}$, and that for i, $1 \le i \le n$, $V_i \equiv_Q U_i$. Then, Q_1 is equivalent to Q if and only if Q_2 is equivalent to Q.

Proof: Let Q_1' (Q_2') be the expansions of Q_1 (Q_2). Note that two subgoals in Q_1' that belong to expansions of different V_i's can only share variables that occur in Q_1 (and similarly for Q_2 and Q_2'). Recall that Q_1 and Q_1' are equivalent queries, and similarly Q_2 and Q_2'.

Assume Q_1 and Q are equivalent queries, and so there are containment mappings ψ_1 from Q to Q_1' and ψ_2 from Q_1' to Q. To show that Q_2 and Q are equivalent queries it suffices to construct two containment mappings: ψ_1' from Q to Q_2' and ψ_2' from Q_2' to Q. We begin with ψ_1'.

Let ψ_1^i be the restriction of ψ_1 to the variables of Q that are mapped to variables in the expansion of V_i. Since $V_i \equiv_Q U_i$, there must be a partial mapping ϕ_i from the variables of Q to the body of U_i, such that $sig_V(\psi_1^i) = sig_V(\phi_i)$. We define ψ_1' to be the union of $\phi_1, \ldots, \phi_n$.

First note that ψ_1' is well defined because (1) the ϕ_i's are identical on variables appearing in Q_1, and (2) if $\psi_1(X)$ does not appear in Q_1, then $\psi_1(X)$ appears in only *one* of the expansions of the subgoals of Q_1. Therefore $\psi_1'(X)$ is determined by exactly one of the ϕ_i's.

Second, note that ψ_1' is a containment mapping from Q to Q_2' because:

- Every atom of Q is mapped to some atom in Q_2', (a subgoal of Q that was mapped to the expansion of V_i will now be mapped to the expansion of U_i), and
- The mappings ψ_1 and ψ_1' are identical on the variables in Q_1, and therefore, ψ_1' maps the head of Q to the head of Q_2'.

We now construct ψ_2'. Since the $V_i \equiv_Q U_i$, there are complete mappings $\tau_1, \ldots, \tau_n$ from $U_1, \ldots, U_n$ to Q respectively, such that $sig_Q(\tau_i) = sig_Q(\psi_2^i)$, where ψ_2^i is the restriction of ψ_2 to variables that appear in the expansion of V_i. As before, we define ψ_2' to be the union of the mappings $\tau_1, \ldots, \tau_n$. Note that ψ_2' is well defined, because on the variables in Q_2, τ_i is identical to ψ_2^i for every i, $1 \le i \le n$. Furthermore, ψ_2' maps the head of Q_2' to the head of Q. Finally, ψ_2' is a containment mapping because every subgoal in the expansion of one of the U_i's is mapped to a subgoal in Q. This follows because if $sig_Q(\psi_2^i) = sig_Q(\tau_i)$, then τ_i is a complete mapping from the variables of U_i to the variables of Q.

The containment mappings ψ_1' and ψ_2' show that Q and Q_2' are equivalent queries. □

The following corollary follows from Theorem 3.1 and from the fact that if a query Q with n subgoals has a rewriting, then it has one with at most n view literals [LMSS95].

Corollary 3.2 *Let $\mathcal{V}$ be a set of conjunctive views, and Q be a conjunctive query. If there is a procedure to pick one representative from each nonempty rewriting-equivalence class of views in $\mathcal{V}$, then the problem of determining whether there is a rewriting of Q using $\mathcal{V}$ is decidable.*

4 Encoding Sets of Views by Datalog Programs

For infinite sets of views to be practically useful, we need a mechanism to compactly encode an infinite set of views. In [PGGMU95] it is shown that several interesting sets of views can be encoded using datalog programs. In this section we show how an infinite set of views is encoded using a datalog program. We then give an algorithm for computing the equivalence classes of the previous section for a set of views $\mathcal{V}$ encoded by a datalog program P, thus showing that for any such set, the view-rewriting problem is decidable.

A datalog program P encodes a set of conjunctive views for every IDB predicate as follows. Consider an IDB predicate p of P. A *finite expansion* of p using P is a finite sequence of rule unfoldings starting with a rule whose head is p. A finite expansion can be viewed as a symbolic derivation tree. The root of the tree is a goal-node labeled with an atom of the predicate p. The single child of a goal-node g is a rule-node g_r. The rule-node g_r is labeled with the result of unifying a rule r of P with the atom labeling g. The children of g_r are goal-nodes labeled with the subgoals in the label of g_r. All the leaves of a symbolic derivation tree are nodes whose atoms are of EDB predicates. A finite expansion of p can be viewed as a conjunctive query whose subgoals are all the leaves of the symbolic derivation tree. We say that the set of views encoded by P for a set of distinguished IDB predicates I is the set including all the finite expansions of predicates in I. The set $\mathcal{V}_P$ denotes the set of all views encoded by P (i.e., when all the IDB predicates are distinguished).

Example 4.1 The set of views $V_1, V_2, \ldots$ can be encoded by the following datalog program with the distinguished predicate v:

$$r_1 : v(X,Y) :- \mathit{blue}(X,Y) \,\&\, \mathit{rp}(Y)$$
$$r_2 : \mathit{rp}(Z) :- \mathit{red}(Z,W)$$
$$r_3 : \mathit{rp}(Z) :- \mathit{red}(Z,W) \,\&\, \mathit{rp}(W) \qquad \square$$

It is important to note that equivalent datalog programs do not necessarily encode the same set of views. In example 4.1 the rule r_3 is redundant. However, if we remove it, we will encode only a subset of the views encoded by the program.

In this section we assume that P has no built-in predicates, and that no head or subgoal in the rules of P has two occurrences of the same variable. We later consider programs with built-in predicates, where all these restrictions will be removed. Note that these assumptions entail that all unifications (when creating expansions) are trivial. Furthermore, by preprocessing the program P with the query-tree [LS92] we can ensure that the program also encodes all conjunctive views resulting from equating variables in the heads of views encoded by P. We denote the equivalence class to which a view V belongs by a pair (S_V^1, S_V^2), where S_V^1 is the set of Q-signatures of complete mappings from V to Q, and S_V^2 is the set of V-signatures of partial mappings from Q to V.

Our algorithm performs a bottom-up evaluation of the program P over an abstract interpretation. For each predicate of P we compute a set of *adornments* of the form (S_1, S_2), where S_1 is a set of Q-signatures and S_2 is a set of V-signatures. Computing an adornment (S_1, S_2) for an IDB predicate p means that some view that is a finite expansion of p belongs to the equivalence class denoted by (S_1, S_2). To describe the computation we need to specify the adornments for the EDB predicates, and to specify how to apply a rule r of P on the domain of adornments.

Adornments of EDB predicates: Every EDB predicate e of P has a single adornment, (S_e^1, S_e^2) defined as follows. Let V be defined as

$$V : v(X_1, \ldots, X_l) :- e(X_1, \ldots, X_l)$$

where $X_1, \ldots, X_l$ are distinct variables. The set S_e^1 is defined to be $\{sig_Q(\psi) \mid \psi \in Maps(V)\}$, where $Maps(V)$ is the set of complete mappings from V to Q. The set S_e^2 is $\{sig_V(\psi) \mid \psi \in Pmaps(V)\}$, where $Pmaps(V)$ is the set of partial variable mappings from Q to V.

Applying a rule: Let r be a rule in P of the form: $r : h(\bar{X}) :- g_1(\bar{X}_1), \ldots, g_l(\bar{X}_l)$, where m_b is the arity of the head predicate, and m_i is the arity of g_i. Suppose $(S_1^1, S_1^2), \ldots, (S_l^1, S_l^2)$ are adornments computed for the predicates $g_1, \ldots, g_l$, respectively. We compute an adornment (S_h^1, S_h^2) for the head predicate h as follows.

The set S_h^1 is computed by considering all the possible combinations of signatures from the S_i^1's. Specifically, suppose that $s_i \in S_i^1$ for i, $1 \leq i \leq l$. Recall that s_i is a mapping from the variables $Z_1, \ldots, Z_{m_i}$ to the variables of Q. Let ϕ_i be the mapping in which the j'th variable of $\bar{X}_i$ is mapped to $s_i(Z_j)$. If the mappings $\phi_1, \ldots, \phi_n$ are pairwise consistent, we define their union mapping ϕ. Let ψ_{head} be the restriction of ϕ to the variables $\bar{X}$, and let s be the mapping from $Z_1, \ldots, Z_{m_h}$ to Q such that $s(Z_j) = \psi(X_j)$, where X_j is the j'th variable of $\bar{X}$. The mapping s is added to S_h^1.

The set S_h^2 is computed by considering all the possible combinations of signatures from the S_i^2's. Specifically, suppose that $t_i \in S_i^2$ for i, $1 \leq i \leq l$. Recall that each of the t_i's is a pair of the form $(\psi_i, \bar{g}_i)$, where ψ_i is a partial mapping from the variables of Q to $Z_1, \ldots, Z_{m_i}$, and g_i is a subset of the subgoals of Q.

Let ϕ_i be the mapping with the same domain as ψ_i such that if $\psi_i(X) = Z_j$, then $\phi_i(X)$ is the j'th variable of $\bar{X}_i$. If the mappings $\phi_1, \ldots, \phi_n$ are pairwise consistent, we denote by ϕ their union mapping, and by ϕ_{head} the restriction of ϕ to variables of Q that are mapped to head variables of r. Finally, we denote by ψ_h the mapping on the variables of Q such that $\psi_h(X) = Z_j$ if $\phi_h(X)$ is the j'th variable of the head of r. If there is no variable X that appears in more than one of the g_i's and is *not* in ϕ_{head}, then we add the pair $(\psi_h, \bar{g}_1 \cup \ldots \cup \bar{g}_n)$ to S_h^2.

Finally, if $(\psi, \bar{g}_1)$ and $(\psi, \bar{g}_2)$ are both in S_h^2 and $\bar{g}_2$ is a strict subset of $\bar{g}_1$, we remove $(\psi, \bar{g}_2)$ from S_h^2.

The bottom-up evaluation of P terminates because there are only a finite number of adornments that can be computed.

Example 4.2 Let us compute the adornment S_v^1 with respect to Q_1. There is only one variable mapping from the EDB predicate $blue(X,Y)$ to Q_1, giving $S_{blue}^1 = \{(X,A),(Y,B)\}$. There are two variable mappings from $red(Z,W)$ to Q_1, giving $S_{red}^1 = \{\{(Z,B),(W,C)\},\{(Z,C),(W,B)\}\}$. Using the rule r_2 for the recursive predicate rp, we project out the Z components of each element in S_{red}^1 and add the elements $\{(Z,B)\}$ and $\{(Z,C)\}$ to S_{rp}^1. Further iterations using rule r_3 do not add new elements to S_{red}^1. Finally, we apply r_1 to obtain $S_v^1 = \{\{(X,A),(Y,B)\}\}$. □

Theorem 4.1 *Let $\mathcal{V}_P$ be the set of conjunctive views encoded for the IDB predicates of the datalog program P. The adornment (S^1, S^2) is computed for an IDB predicate of arity m of P if and only if there is some view $V \in \mathcal{V}_P$ of arity m, such that $(S_V^1, S_V^2) = (S^1, S^2)$.*

Theorem 4.2 *Let $\mathcal{V}_P$ be the set of conjunctive views defined by a datalog program P, and let Q be a conjunctive query. The question of whether there exists a rewriting of Q using views in $\mathcal{V}_P$ is decidable in nondeterministic doubly exponential time in the size of Q and $\mathcal{V}_P$.*

5 Parameterized View Definitions

In some applications the grammar describing the set of views computable by an external source actually describes a set of *parameterized views* [PGGMU95]. A parameterized view is a conjunctive query that contains *placeholders* in argument positions in the body of the view, in addition to variables and constants. The parameterized view V represents the set of all view definitions obtained by assigning a constant to each place holder. Parameterized views provide a method for describing views in which constraints are applied to variables that appear only in the body of the view and not in the head. Using parameterized view definitions we can also encode the capabilities of information sources that may accept an unbounded number of inputs as part of the query, while still maintaining a bounded arity for the views computed by the sources.

Example 5.1 Let us consider a bibliographic information source that can find titles of publications given the names of one or more authors. Suppose the base relations are $title(Book, T)$ and $author(Book, A)$. The queries accepted by the source are described by the set of parameterized views V_i, defined by:

$$V_i : v_i(T) :- \; title(B,T) \;\&\; author(B, *A_1) \;\&\; \ldots \;\& \\ author(B, *A_i)$$

Placeholders are denoted by argument names beginning with an asterisk (*). It is straightforward to construct a recursive Datalog program that can generate the above set of parameterized views. □

The following theorem states that when trying to answer a query Q, we do not have to consider instances of a parameterized view in which a placeholder is mapped to a constant that does not appear in Q. Consequently, if $\mathcal{V}$ is a set of parameterized views encoded by a datalog program, then the rewriting problem is decidable.

Theorem 5.1 *Let $\mathcal{V}$ be a set of parameterized conjunctive views, and Q be a conjunctive query. If there is a rewriting of Q using the parameterized views $V_1, \ldots, V_n$ in $\mathcal{V}$ then there is a rewriting in which all the placeholders in $V_1, \ldots, V_n$ are mapped to constants appearing in Q.*

In Section 6 we generalize this result to queries and views with built-in predicates, where placeholders can appear in built-in subgoals as well as in subgoals of base relations.

6 Queries and Views with Built-in Predicates

In this section we consider queries and views that use the *built-in predicates* $\leq$, $<$, $=$, and $\neq$. We assume that the built-in predicates are interpreted over a dense domain. Algorithms for containment and equivalence of conjunctive queries with built-in predicates were first considered by Klug [Klu88] and then in [LS93, GSUW94]. Klug showed that the containment problem for such queries is in Π_2^p. Van der Meyden [vdM92] later showed the containment and equivalence problems to be complete for Π_2^p. Given a conjunctive query Q with built-in predicates, its *core* is the subset of its subgoals whose predicate is not a built-in predicate.

The crucial question we must ask in order to extend our query-rewriting algorithm to queries with built-in predicates is the following: given a query Q, which constants must we consider using in rewritings of Q? In the case of conjunctive queries without built-in predicates, the answer is simple: the only constants we need to use are those that appear in Q itself. When the built-in predicates are interpreted over a dense domain, Klug showed that any pair of equivalent queries Q_1 and Q_2 use same set of *essential constants*. The theorem we state below follows from [Klu88].[1]

[1] Klug considers only queries that use the built-in predicates $<$, $\leq$, and $=$. However, his results can be extended to queries that also use the predicate $\neq$ for dense domains.

Theorem 6.1 *Let Q_1 and Q_2 be equivalent conjunctive queries with built-in predicates interpreted over a dense domain, and suppose Q_2 uses some constants that do not appear in Q_1. Then we can rewrite Q_2 leaving its core unchanged, to obtain an equivalent query Q_3 that uses only the constants in Q_1. Moreover, the number of subgoals in Q_3 is no more than the number of subgoals in Q_1.*

We use Theorem 6.1 to extend our results to the rewriting problem for queries and views with built-in predicates and placeholders in the case of dense domains. We assume that all views in $\mathcal{V}$ use constants from a finite set $\mathcal{C}$ (however, placeholders can still be assigned constants not in $\mathcal{C}$). The theorem below generalizes Theorem 3.1.

Theorem 6.2 *Let Q be a conjunctive query with built-in predicates interpreted over a dense domain, and let $\mathcal{V}$ be a (possibly infinite) set of parameterized conjunctive queries with built-in predicates. It is possible to partition $\mathcal{V}$ into a finite number of equivalence classes $\mathcal{E}$, such that if V_i and U_i belong to the same equivalence class in $\mathcal{E}$, for i, $1 \leq i \leq n$, and*

$$Q_1 : q_1(\bar{X}) : -V_1(\bar{X}_1) \,\&\, \ldots \,\&\, V_n(\bar{X}_n), C$$
$$Q_2 : q_2(\bar{X}) : -U_1(\bar{X}_1) \,\&\, \ldots \,\&\, U_n(\bar{X}_n), C$$

then Q_1 is equivalent to Q if and only if Q_2 is equivalent to Q.

The proof is based on refining Definition 3.3 to ensure that views in the same equivalence class participate in the same ways in containment mappings to and from the query, for *every* possible ordering of the variables. Theorem 6.1 and Theorem 6.2 entail the following theorem:

Theorem 6.3 *Let $\mathcal{V}$ be a (perhaps infinite) set of parameterized conjunctive views, and Q be a conjunctive query, both $\mathcal{V}$ and Q with built-in predicates and using a finite set of constants $\mathcal{C}$. If there is a procedure to pick one representative from each nonempty rewriting-equivalence class for views in $\mathcal{V}$, then the problem of determining whether there is a rewriting of Q using $\mathcal{V}$ is decidable.*

Finally, we show that if Q is a conjunctive query all of whose built-in subgoals are *local*, then the rewriting problem is decidable when the set of views is encoded by a datalog program. A built-in subgoal g is local if there is some non built-in subgoal g_1 in the body of the Q that contains all the variables in g.

Theorem 6.4 *Let $\mathcal{V}_P$ be the set of conjunctive views encoded by a datalog program P. Let Q be a conjunctive query all of whose built-in subgoals are local. The problem of finding a rewriting of Q using $\mathcal{V}$ is decidable.*

It should be noted that computing the equivalence classes in Theorem 6.2 is undecidable if the query Q can contain non-local built-in subgoals.

7 Queries with Built-in Predicates over the Integers

Klug's result, and therefore Theorem 7.2, applies only to dense domains (such as the reals), and not to nondense domains (such as the integers). In this section, we consider queries and views that do not contain the predicate $\neq$ and generalize Theorem 6.1 to the integer domain.

In what follows, the term "query" refers to a conjunctive query with built-in predicates $<$, $\leq$, and $=$ interpreted over the integers. We denote by $B(Q)$ the conjunction of built-in predicates in query Q. A *term* of a query Q is a variable or a constant that appears in Q. A complete ordering on Q is a conjunction of built-in atoms that completely determines all the ordering relations between the terms in Q and is consistent with $B(Q)$. Formally, D is said to be a *complete ordering* on Q if:

- $B(Q) \wedge D$ is satisfiable, and
- for every atomic formula g of the form $\alpha\theta\beta$, where α and β are terms of Q and $\theta \in \{=, \leq, <, \geq, >\}$, then either $D \models g$ or $D \models \neg g$.

We now prove a somewhat weaker version of Theorem 6.1 in the case of integers. Let $C(Q)$ denote the set of constants in Q, and let m be the number of variables in Q. If Q has no constants, we set $C(Q)$ to $\{0\}$. For each $c \in C(Q)$, define its *interval* I_c as follows:

$$I_c = \begin{cases} [c-m, c+m] & \text{if } c \text{ is the smallest constant} \\ & \text{in } C(Q) \\ [c, c+m] & \text{otherwise} \end{cases}$$

The interval of Q is defined to be $I(Q) = \bigcup_{c \in C(Q)} I_c$.

Lemma 7.1 *Let D be a total ordering on the variables of query Q. Then there is a mapping ϕ from the variables of Q to $I(Q)$ such that $\phi(B(Q))$ is consistent with D (that is, $\phi(B(Q)) \wedge D$ is satisfiable).*

Proof.(Sketch) Partition the terms (variables and constants) of Q into equivalence classes based on the total ordering D, such that $D \models \alpha = \beta$ for terms α and β in the same equivalence class. Define the total order $<$ on the equivalence classes as follows: $[\alpha] < [\beta]$ if $D \models \alpha < \beta$.

For a constant $c \in C(Q)$, we say that equivalence class $[\alpha]$ is in the *right neighborhood* of c if $[c] < [\alpha]$ and there is no constant d such that $c < d$ and $[d] < [\alpha]$. The left neighborhood of $[c]$ is defined symmetrically. Define the mapping ϕ as follows: For each variable $X \in [c]$, $\phi(X) = c$. Let $\{[\alpha_1], \ldots, [\alpha_k]\}$ be the right neighborhood of c, and let $[\alpha_1] < \ldots < [\alpha_k]$. For each variable $X \in [\alpha_i]$, define $\phi(X) = c + i$. Finally, we consider the left neighborhood $\{[\alpha_1], \ldots, [\alpha_r]\}$ of $[d]$, where d is the least constant in $C(Q)$. Let $[\alpha_r] < \ldots < [\alpha_1]$. For each variable $X \in [\alpha_i]$, define $\phi(X) = d - i$.

It is easy to show that the mapping ϕ is well-defined and satisfies the conditions of the lemma. $\square$

We are now ready to prove the analog of Theorem 6.1 for the integers.

Theorem 7.2 *Let Q_1 and Q_2 be equivalent conjunctive queries with built-in predicates $<$, $\leq$, and $=$ interpreted over the integers, and suppose Q_2 uses some constants that do not appear in Q_1. Then we can rewrite Q_2 leaving its core unchanged, to obtain an equivalent query Q_3 that uses only the constants in $I(Q_1)$. Moreover, the number of subgoals in Q_3 is no more than the number of subgoals in Q_1.*

Proof. (Sketch) We assume without loss of generality that $B(Q_1)$ is satisfiable, and so Q_1 is not the empty query on all databases. If Q_2 contains an atom of the form $X = c$, where c does not appear in Q_1, then Q_2 cannot contain Q_1.

Construct Q_3 to be Q_2 with the following modifications.

- Suppose Q_2 contains a subgoal of the form $X \leq c$ or $X < c$ for some constant c that does not appear in $I(Q_1)$. In Q_3, replace the subgoal by $X \leq d$, where d is the largest constant in $I(Q_1)$ such that $d \leq c$.

- Suppose Q_2 contains a subgoal of the form $c \leq X$ or $c < X$ for some constant c that does not appear in $I(Q_1)$. In Q_3, replace the subgoal by $d \leq X$, where d is the smallest constant in $I(Q_1)$ such that $d \geq c$.

Clearly $Q_3 \subseteq Q_2$ (since we have only "strengthened" conditions in Q_2 to obtain Q_3). Since Q_1 and Q_2 are equivalent, it suffices to show that $Q_1 \subseteq Q_3$.

Suppose to the contrary that $Q_1 \not\subseteq Q_3$. Then there exists a database D and a fact $\bar{a}$ such that $\bar{a} \in Q_1(D)$ but $\bar{a} \notin Q_3(D)$. The database D and the mapping from the variables of Q_1 to the constants of D induce a total ordering on the variables and constants of Q_1. Using Lemma 7.1 we can show that there exists a mapping ψ on the constants of D, such that $\psi(\bar{a}) \in Q_1(\psi(D))$, $\psi(\bar{a}) \notin Q_3(\psi(D))$, and such that $\psi(D)$ contains only constants in $I(Q_1)$. The construction of Q_3 entails that Q_2 and Q_3 are identical on databases that include only constants in $I(Q_1)$. Therefore, $\psi(\bar{a}) \notin Q_2(\psi(D))$, contradicting the fact that Q_1 and Q_2 are equivalent. $\square$

Given Theorem 7.2, all the results of Section 6 apply to queries, views, and rewritings that use the built-in predicates $=$, $<$, and $\leq$ interpreted over the integers.

7.1 Minimization over Integers

We now apply Theorem 7.2 to the problem of query minimization. The minimization problem we address is the following: given a conjunctive query with built-in subgoals Q, find a query Q' that has as few subgoals of ordinary (non-built-in) predicates as possible and no redundant built-in subgoals. Klug [Klu88] gave a minimization algorithm that is in Σ_3^p for the case of dense domains. Since the size of $I(Q)$ is polynomial in the size of Q, Theorem 7.2 can be used to construct a similar nondeterministic algorithm for the integers when queries do not use the predicate $\neq$.

Theorem 7.3 *Let Q be a conjunctive query with built-in predicates $<$, $\leq$, and $=$ interpreted over the integers. The problem of minimizing Q is in Σ_3^p.*

8 Conclusions

In this paper we considered the problem of rewriting queries using a possibly infinite set of views and sets of parameterized views. This problem is important when we need to answer queries using a collection of external information sources that can also answer complex queries about their contents. In this context, each information source can be modeled as being able to answer a possibly infinite set of views. We showed that an infinite set of views can be partitioned into a finite number of equivalence classes, such that the view rewriting problem can be completely solved by considering one representative from every non-empty equivalence class. The importance of this result is that we can now focus on computing the set of equivalence classes for a given set of views. The problem of computing the classes may have a more direct solution based on the specific encoding of the set of views. We described an algorithm for computing the set of equivalence classes for sets of views encoded by datalog programs. Our results extend also to views and queries with built-in comparison predicates.

We are currently extending our work in several directions. First, we are considering the case in which the views can only be used with specific binding patterns [RSU95]. Second, we are considering the case in which we need a rewritten query that is *contained* in the original query, and not necessarily equivalent to it. Finally, we are extending the results of Section 7 to queries that include the predicate $\neq$.

Acknowledgements

We thank Chandra Chekuri for helpful discussions on Section 7 and also for comments on a draft of this paper.

References

[ACHK94] Yigal Arens, Chin Y. Chee, Chun-Nan Hsu, and Craig A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal on Intelligent and Cooperative Information Systems*, 1994.

[BI94] Daniel Barbará and Tomasz Imieliński. Sleepers and workaholics: Caching strategies in mobile environments. In *Proceedings of SIGMOD-94*, pages 1–12, 1994.

[CGMH+94] Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey Ullman, and Jennifer Widom. The TSIMMIS project: Integration of heterogeneous information sources. In proceedings of IPSJ, Tokyo, Japan, October 1994.

[CKPS95] Surajit Chaudhuri, Ravi Krishnamurthy, Spyros Potamianos, and Kyuseok Shim. Optimizing queries with materialized views. In *Proceedings of International Conference on Data Engineering*, 1995.

[CM77] A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, pages 77–90, 1977.

[EW94] Oren Etzioni and Dan Weld. A softbot-based interface to the internet. *CACM*, 37(7):72–76, 1994.

[GMR95] Ashish Gupta, Inderpal Singh Mumick, and Kenneth A. Ross. Adapting materialized views after redefinitions. In *Proceedings of SIGMOD-95*, 1995.

[GSUW94] Ashish Gupta, Yehoshua Sagiv, Jeffrey D. Ullman, and Jennifer Widom. Constraint checking with partial information. In *Proceedings of the Thirteenth Symposium on Principles of Database Systems (PODS)*, pages 45–55, 1994.

[HSW94] Yixiu Huang, Prasad Sistla, and Ouri Wolfson. Data replication for mobile computers. In *Proceedings of SIGMOD-94*, pages 13–24, 1994.

[Klu88] A. Klug. On conjunctive queries containing inequalities. *Journal of the ACM*, pages 35(1): 146–160, 1988.

[LMSS95] Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. In *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, San Jose, CA*, 1995.

[LRO95] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. AT&T Bell Labs technical report. Submitted for publication, 1995.

[LRO96] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Query answering algorithms for information agents. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI–96*, 1996.

[LS92] Alon Y. Levy and Yehoshua Sagiv. Constraints and redundancy in Datalog. In *The Proceedings of the Eleventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, San Diego, CA.*, pages 67–80, 1992.

[LS93] Alon Y. Levy and Yehoshua Sagiv. Queries independent of updates. In *Proceedings of the 19th VLDB Conference, Dublin, Ireland*, pages 171–181, 1993.

[LSK95] Alon Y. Levy, Divesh Srivastava, and Thomas Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems, Special Issue on Networked Information Discovery and Retrieval*, 5 (2), September 1995.

[PGGMU95] Yannis Papakonstantinou, Ashish Gupta, Hector Garcia-Molina, and Jeffrey Ullman. A query translation scheme for rapid implementation of wrappers. In *In proceedings of the Conference on Deductive and Object Oriented Databases, DOOD-95, To appear*, 1995.

[RSU95] Anand Rajaraman, Yehoshua Sagiv, and Jeffrey D. Ullman. Answering queries using templates with binding patterns. In *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, San Jose, CA*, 1995.

[SAB+95] V.S. Subrahmanian, S. Adali, A. Brink, R. Emery, J. Lu, A. Rajput, T. Rogers, R. Ross, and C. Ward. HERMES: A heterogeneous reasoning and mediator system. Technical report, University of Maryland, 1995.

[TSI94] Odysseas G. Tsatalos, Marvin H. Solomon, and Yannis E. Ioannidis. The GMAP: A versatile tool for physical data independence. In *Proceedings of the 20th International VLDB Conference, Santiago, Chile*, pages 367–378, 1994.

[vdM92] Ron van der Meyden. The complexity of querying indefinite data about linearly ordered domains. In *The Proceedings of the Eleventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, San Diego, CA.*, pages 331–345, 1992.

[YL87] H. Z. Yang and P. A. Larson. Query transformation for PSJ-queries. In *Proceedings of the 13th International VLDB Conference*, pages 245–254, 1987.