Minimizing View Sets without Losing Query-Answering Power (Extended Version)

Chen Li, Mayank Bawa, and Jeffrey D. Ullman
Department of Computer Science
Stanford University, CA 94305
{chenli,bawa,ullman}@db.stanford.edu

Abstract

The problem of answering queries using views has been studied extensively, due to its relevance in a wide variety of data-management applications. In these applications, we often need to select a subset of views to maintain, due to limited resources. In this paper, we show that traditional query containment is *not* a good basis for deciding whether or not a view should be selected. Instead, we should minimize the view set without losing *query-answering power.* To formalize this notion, we first introduce the concept of "p-containment." That is, a view set $\mathcal{V}$ is *p-contained* in another view set $\mathcal{W}$, if $\mathcal{W}$ can answer all the queries that can be answered by $\mathcal{V}$. We show that p-containment and the traditional query containment are *not* related; i.e., one does not imply the other. We then discuss how to minimize a view set while retaining its query-answering power. We develop the idea further by considering p-containment of two view sets with respect to a given set of queries, and consider their relationship in terms of maximally-contained rewritings of queries using the views.

1 Introduction

The problem of answering queries using views [AGK99, Dus97, CGLV99, GT00, LMSS95, Qia96] has been studied extensively, because of its relevance to a wide variety of data management problems, such as information integration, data warehousing, and query optimization. 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? Recently, Levy compiled a good survey [Lev00] about the different approaches to this problem.

In the context of query optimization, computing a query using previously materialized views can speed up query processing, because part of the computation necessary for the query may have been done while computing the views. In a data warehouse, views can preclude costly access to the base relations and help answer queries quickly. In web-site designs, precomputed views can be used to improve the performance of web-sites [FLSY99]. Before choosing an optimal design, we must assure that the chosen views can be used to answer the queries we expect to receive at the web-site. A system that caches answers locally at the client can avoid accesses to base relations at the server. Cached result of a query can be thought of as a materialized view, with the query as its view definition. The client could use the cached answers from previous queries to answer future queries.

However, the benefits presented by views are not without costs. Materialized views often compete for limited resources. Thus, it is critical to select views carefully. For instance, in an information-integration system [Ull97], a view may represent a set of web pages at an autonomous source. The mediator [Wie92] in these systems often needs to crawl these web pages periodically to refresh the cached data in its local repository [CGM00]. In such a scenario, the cost manifests itself as the bandwidth needed for such crawls and the efforts in maintaining the cache up-to-date. Correspondingly, in a query-optimization and database-design scenario, the materialized views may have part of the computation necessary for the query. When a user poses a query, we need to decide how to answer the query using the materialized views. By selecting an optimal subset of views to materialize, we can reduce the computation needed to decide how to answer typical queries. In a client-server architecture with client-side caching, storing all answers to past queries may need a large storage space and will add to the maintenance costs. Since the client needs to deal with an evolving set of queries, any of these can be used to answer future queries. Thus, redundant views need not be cached.

The following example shows that views can have redundancy to make such a minimization possible, and that traditional query containment is *not* a good basis for deciding whether a view should be selected or not. Instead, we should consider the *query-answering* power of the views.

EXAMPLE 1.1 Suppose we have a client-server system with client-side caching for improving performance, since server data accesses are expensive. The server has the following base relation about books:

$$book(Title, Author, Pub, Price)$$

For example, the tuple $\langle$`databases`, `smith`, `prenhall`, `$60`$\rangle$ in the relation means that a book titled `databases` has an author `smith`, is published by Prentice Hall (`prenhall`), and has a current price of $60. Assume that the client has seen the following three queries, the answers of which have been cached locally. The cached data (or views), denoted by the view set $\mathcal{V} = \{V_1, V_2, V_3\}$, are:

$$\begin{array}{lll} V_1: & v_1(T,A,P) & \text{:- } book(T,A,B,P) \\ V_2: & v_2(T,A,P) & \text{:- } book(T,A,prenhall,P) \\ V_3: & v_3(A_1,A_2) & \text{:- } book(T,A_1,prenhall,P_1), book(T,A_2,prenhall,P_2) \end{array}$$

The view V_1 has author-title-price information about all books in the relation, while the view V_2 includes this information only about books published by Prentice Hall. The view V_3 has coauthor pairs for books published by Prentice Hall. Since the view set has redundancy, we might want to eliminate a view to save costs of its maintenance and storage. At the same time, we want to be assured that such an elimination does not cause increased server accesses in response to future queries.

Clearly, view V_2 is contained in V_1, i.e., V_1 includes all the tuples in V_2, so we might be tempted to select $\{V_1, V_3\}$, and eliminate V_2 as a redundant view. However, with this selection, we cannot answer the query:

$$Q_1 : q_1(T,P) \text{ :- } book(T, smith, prenhall, P)$$

which asks for titles and prices of books written by `smith` and published by `prenhall`. The reason is that even though V_1 includes author-title-price information about all books in the base relation, the publisher attribute is projected out in the head of the view V_1. Thus, using V_1 only, we cannot tell which books are published by `prenhall`. On the other hand, the query Q_1 can be answered trivially using V_2:

$$q_1(T,P) \text{ :- } v_2(T, smith, P)$$

In other words, by dropping V_2 we have lost some *power* to answer queries. In addition, note that even though view V_3 is not contained in V_1 and V_2, it can be eliminated from $\mathcal{V}$ without changing the query-answering power of $\mathcal{V}$. The reason is that V_3 can be computed from V_2 as follows:

$$P_1 : v_3(A_1,A_2) \text{ :- } v_2(T,A_1,P_1), v_2(T,A_2,P_2)$$

To summarize, we should not select $\{V_1, V_3\}$ but $\{V_1, V_2\}$, even though the former includes all the tuples in $\mathcal{V}$, while the latter does not. The rationale is that the latter is as "powerful" as $\mathcal{V}$ while the former is not. (We will give a formal proof in Section 3.) *Caution:* One might hypothesize from this example that only projections in view definitions cause such a mis-match, since we do not lose any "data" in the body of the view. We show in Section 3 that this hypothesis is wrong. □

In this paper we discuss how to minimize a view set without losing its query-answering power. We first introduce the concept of *p-containment* between two sets of views, where "p" stands for query-answering *power* (Sections 2 and 3). A view set $\mathcal{V}$ is *p-contained* in another view set $\mathcal{W}$, or $\mathcal{W}$ is *at least as powerful* as $\mathcal{V}$, if $\mathcal{W}$ can answer all the queries that can be answered using $\mathcal{V}$. Two view sets are called *equipotent* if they have the same power to answer queries. As shown in Example 1.1, two view sets may have the same tuples, yet have different query-answering power. That is, traditional view containment [CM77, SY80] does not imply p-containment. The example further shows that the reverse direction is also *not* implied.

Given a view set $\mathcal{V}$ on base relations, we show how to find a minimal subset of $\mathcal{V}$ that is equipotent to $\mathcal{V}$ (Section 4). As one might suspect, a view set can have multiple equipotent minimal subsets. However, it would be helpful to postulate the necessary properties of the set for the equipotent minimal subset to be unique. We deduce the conditions for ensuring uniqueness and provide some heuristics for speeding up the process for finding a minimal, equipotent subset.

In some scenarios, users are restricted in the queries they can ask. In such cases, equipotence may be determined *relative* to the expected (possibly infinite) set of queries. In Section 5, we investigate the above questions of equipotence testing given this extra constraint. In particular, we consider infinite query sets defined by parameterized queries, and develop algorithms for testing this relative p-containment.

In information-integration systems, we often need to consider not only equivalent rewritings of a query using views, but also maximally-contained rewritings (MCR's). Analogous to p-containment, which requires equivalent rewritings, we introduce the concept of *MCR-containment* that is defined using maximally-contained rewritings (Section 6). Surprisingly, we show that p-containment implies MCR-containment, and vice-versa.[1]

The different containments between two view sets that are discussed in this paper are summarized in Table 1. We start our discussions with conjunctive queries, and generalize the results to more general queries in Section 7.

Containment	**Definition**	**How to test**
v-containment $\mathcal{V} \sqsubseteq_v \mathcal{W}$	For any database, a tuple in a view in $\mathcal{V}$ is in a view in $\mathcal{W}$.	Check if each view in $\mathcal{V}$ is contained in some view of $\mathcal{W}$.
p-containment $\mathcal{V} \preceq_p \mathcal{W}$	If a query is answerable by $\mathcal{V}$, then it is answerable by $\mathcal{W}$.	Check if each view in $\mathcal{V}$ is answerable by $\mathcal{W}$.
relative p-containment $\mathcal{V} \preceq_{\mathcal{Q}} \mathcal{W}$	For each query Q in a given set of queries $\mathcal{Q}$, if Q is answerable by $\mathcal{V}$, then Q is answerable by $\mathcal{W}$.	Test by the definition if $\mathcal{Q}$ is finite. See Section 5.3 for infinite queries generated by parameterized queries.
MCR-containment $\mathcal{V} \preceq_{MCR} \mathcal{W}$	For each query Q, for any maximally -contained rewriting $MCR(Q, \mathcal{V})$ (resp. $MCR(Q, \mathcal{W})$) of Q using $\mathcal{V}$ (resp. $\mathcal{W}$), $MCR(Q, \mathcal{V}) \sqsubseteq MCR(Q, \mathcal{W})$.	Same as testing if $\mathcal{V} \preceq_p \mathcal{W}$, since $\mathcal{V} \preceq_p \mathcal{W} \Leftrightarrow \mathcal{V} \preceq_{MCR} \mathcal{W}$.

Table 1: Different containments between two sets of conjunctive views: $\mathcal{V}$ and $\mathcal{W}$.

[1] This result was sufficiently surprising to us to deserve an extra footnote.

1.1 Related work

There has been a lot of work on the problem of selection of views to materialize in a data warehouse. In [IK93, HGMW^{+}95, Wid95, CW91], a data warehouse is modeled as a repository of integrated information available for querying and analysis. The materialized views are stored with an intention of making the frequent queries fast. The system accesses base relations for a query which cannot be answered using the materialized views. Each of the materialized views can be huge, resulting in the need for careful selection under the constraint of limited disk space [HRU96, GHRU97, RSS96, YKL97, BPT97, TS97]. Moreover, base relations can change over time. The updates have to be propagated to the materialized views resulting in a maintenance cost [GHRU97, GM99b]. The study in this setting has, therefore, emphasised on modeling the view-selection problem as cost-benefit analysis. Thus, for a given set of queries, various sets of sub-queries are considered for materialization. Redundant views in a set which increase cost are deduced using query containment, and an optimal subset is chosen.

Such a model is feasible when all queries can be answered in the worst case by accessing the base relations, and not by views alone. This assumption is incorporated in the model by replicating base relations at the warehouse, without taking the costs of such a step into account. Thus, the base relations themselves are considered to be normalized, independent, and minimal. However, when real-time access to base relations is prohibitive, such an approach can lead to wrong conclusions, as was seen in Example 1.1. In such a scenario, it is essential to ensure the *computability* of queries using *only* maintained views.

Our work is directed towards scenarios where the following assumptions hold. (1) Real-time access to base relations is prohibitive, or possibly denied, and (2) cached views are expensive to maintain over time, because of the high costs of propogating changes from base relations to views. Therefore, while minimizing a view set, it is important to retain its *query-answering* power. We believe the power of answering queries and the benefit/costs of a view set are orthogonal issues, and their interplay would make an interesting work in its own right.

Recently, [CG00] has proposed solutions to the following problem of *database reformulation:* given a query on base relations, how can we materialize views to answer the query? The authors show that there can be infinite number of view sets that can answer the same query. Thus, their work starts with a given query but no views. In our framework, we assume that a set of views are given, and queries can be arbitrary. We would like to deduce a minimal subset of views that can answer all queries answerable by the original set.

2 Background

In this section, we review some concepts about answering queries using views [LMSS95]. We assume the reader is familiar with the survey [Lev00]. Let $r_1, \ldots, r_m$ be m *base relations* in a database. We first consider queries on the database in the following conjunctive form:[2]

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

In each subgoal $g_i(\bar{X}_i)$, predicate g_i is a base relation, and every argument in the subgoal is either a variable or a constant. We consider views defined on the base relations by safe conjunctive queries, i.e., every variable in a query's head appears in its body. Note that we take the closed-world assumption [AD98], since the views are computed from existing database relations. We shall use names beginning

[2] In Section 7 we shall extend our results of conjunctive queries to more general queries.

with lower-case letters for constants and relations, and names beginning with upper-case letters for variables.

Definition 2.1 (**query containment and equivalence**) A query Q_1 is *contained* in a query Q_2, denoted by $Q_1 \sqsubseteq Q_2$, if for any database D of the base relations, the set of tuples computed for Q_1 is a subset of those computed for Q_2, i.e., $Q_1(D) \subseteq Q_2(D)$. The two queries are *equivalent* if $Q_1 \sqsubseteq Q_2$ and $Q_2 \sqsubseteq Q_1$. □

Definition 2.2 (**expansion of a query using views**) The *expansion* of a query P on a set of views $\mathcal{V}$, denoted by P^{exp}, is obtained from P by replacing all the views in P with their corresponding base relations. Existentially quantified variables in a view are replaced by fresh variables in P^{exp}. □

Definition 2.3 (**rewritings and equivalent rewritings**) Given a query Q and a view set $\mathcal{V}$, a query P is a *rewriting* of query Q using $\mathcal{V}$ if P uses only the views in $\mathcal{V}$, and $P^{exp} \sqsubseteq Q$. P is an *equivalent rewriting* of Q using $\mathcal{V}$ if P^{exp} and Q are equivalent. We say a query Q is *answerable* by $\mathcal{V}$ if there exists an equivalent rewriting of Q using $\mathcal{V}$. □

In Example 1.1, P_1 is an equivalent rewriting of the query Q_1 using view V_2, because the expansion of P_1:

$$P_1^{exp} : q_1(T, R) \text{ :- } book(T, smith, P, R)$$

is equivalent to Q_1. Thus, query Q_1 is answerable by V_2, but it is not answerable by $\{V_1, V_3\}$.

Several algorithms have been developed for answering queries using views, such as the bucket algorithm [LRO96, GM99a], the inverse-rule algorithm [Qia96, DG97], and the algorithms in [Mit99, PL00]. See [LMSS95, AD98] for a study of the complexity of answering queries using views. In particular, it has been shown that the problem of rewriting a query using views is $\mathcal{NP}$-hard. Our goal is to minimize a view set without losing its power to answer queries.

3 Comparing the query-answering power of view sets

Before discussing how to minimize a view set without losing its query-answering power, we first introduce the concept of *p-containment*. It captures the fact that one view set may be more powerful than another view set, in terms of the possible queries they can answer. We also compare p-containment with traditional query containment.

3.1 p-containment

Definition 3.1 (**p-containment and equipotence**) A view set $\mathcal{V}$ is *p-contained* in another view set $\mathcal{W}$, or "$\mathcal{W}$ is at least as powerful as $\mathcal{V}$," denoted by $\mathcal{V} \preceq_p \mathcal{W}$, if any query answerable by $\mathcal{V}$ is also answerable by $\mathcal{W}$. Two view sets are *equipotent*, denoted by $\mathcal{V} \asymp_p \mathcal{W}$, if $\mathcal{V} \preceq_p \mathcal{W}$, and $\mathcal{W} \preceq_p \mathcal{V}$. □

In Example 1.1, the two view sets $\{V_1, V_2\}$ and $\{V_1, V_2, V_3\}$ are equipotent, since the latter can answer all the queries that can be answered by the former, and vice-versa. (We will give a formal proof shortly.) However, the two view sets, $\{V_1, V_3\}$ and $\{V_1, V_2, V_3\}$, are not equipotent, since the latter can answer the query Q_1, which cannot be answered by the former. The following theorem suggests an algorithm for testing p-containment.

Theorem 3.1 *Let $\mathcal{V}$ and $\mathcal{W}$ be two view sets. $\mathcal{V} \preceq_p \mathcal{W}$ iff for every view $V \in \mathcal{V}$, if treated as a query, V is answerable by $\mathcal{W}$.*[3] □

The importance of this theorem is that given two view sets $\mathcal{V}$ and $\mathcal{W}$, we can test $\mathcal{V} \preceq_p \mathcal{W}$ simply by checking if every view in $\mathcal{V}$ is answerable by $\mathcal{W}$. That is, we can just consider a finite set of queries, even though $\mathcal{V} \preceq_p \mathcal{W}$ means that $\mathcal{W}$ can answer *all* the infinite number of queries that can be answered by $\mathcal{V}$. We can use the algorithms in [DG97, GM99a, LRO96, Qia96] to do the checking. It can be shown that the problem of checking p-containment is $\mathcal{NP}$-hard. It is also easy to see that the relationship "$\preceq_p$" is reflexive, antisymmetric, and transitive.

EXAMPLE 3.1 As we saw in Example 1.1, view V_3 is answerable by the view V_2. By Theorem 3.1, we have $\{V_1, V_2, V_3\} \preceq_p \{V_1, V_2\}$. Clearly the other direction is also true, so $\{V_1, V_2\} \asymp_p \{V_1, V_2, V_3\}$. On the other hand, V_2 cannot be answered using $\{V_1, V_3\}$, which means $\{V_1, V_2, V_3\} \not\preceq_p \{V_1, V_3\}$. □

We are interested in the relationship between p-containment and the traditional concept of query containment. Before making the comparisons, we first generalize the latter to a concept called *v-containment.*

3.2 v-containment

We use v-containment to capture the fact that one view set may store more tuples than another view set. The motivation for introducing v-containment, rather than using the traditional concept of query containment (as in Definition 2.1), is to cover the cases where the views in a set have different schemas.

Definition 3.2 (**v-containment and v-equivalence**) A view set $\mathcal{V}$ is *v-contained* in another view set $\mathcal{W}$, denoted by $\mathcal{V} \sqsubseteq_v \mathcal{W}$, if the following holds. For any database D of the base relations, if tuple t is in $V(D)$ for a view $V \in \mathcal{V}$, then there exists a view $W \in \mathcal{W}$, such that $t \in W(D)$. The two sets are *v-equivalent*, if $\mathcal{V} \sqsubseteq_v \mathcal{W}$, and $\mathcal{W} \sqsubseteq_v \mathcal{V}$. □

In Example 1.1, the two view sets $\{V_1, V_2, V_3\}$ and $\{V_1, V_3\}$ are v-equivalent, while their views have different schemas. The definition of v-containment allows the views in the sets to have different schemas as well, unlike conventional query containment. Given two view sets $\mathcal{V}$ and $\mathcal{W}$, we want to test $\mathcal{V} \sqsubseteq_v \mathcal{W}$. The following theorem shows that this test can be done easily for conjunctive views.

Theorem 3.2 *Let $\mathcal{V}$ and $\mathcal{W}$ be two sets of conjunctive views. Then $\mathcal{V} \sqsubseteq_v \mathcal{W}$ iff for every view $V \in \mathcal{V}$, there is a view $W \in \mathcal{W}$, such that $V \sqsubseteq W$.* □

For more complicated queries, such as conjunctive queries with arithmetic comparisons, unions of conjunctive queries, and datalog queries, it becomes more challenging to test v-containment. For example, [Gup94] shows that if conjunctive views with arithmetic comparisons are considered, two views can "team up" to contain one view, while neither of them contains the view by itself.

[3] *Due to space limitations, we do not provide all the proofs of the lemmas and theorems. Some proofs are given in the appendix.*

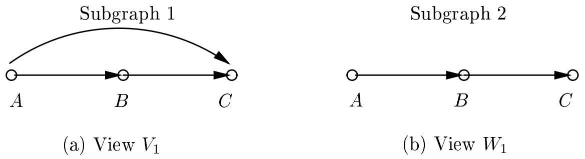
3.3 Comparison between p-containment and v-containment

Example 1.1 shows that v-containment does not imply p-containment, and vice-versa. One might guess that if we do not allow projections in the view definitions (i.e., all the variables in the body of a view appear in the head), then v-containment could imply p-containment. However, the following example shows that this guess is incorrect.

EXAMPLE 3.2 Let $e(X_1, X_2)$ be a base relation, where a tuple $e(x, y)$ means that there is an edge from vertex x to vertex y in a graph. Consider the following two view sets:

$$\begin{array}{llll} \mathcal{V} = \{V_1\}, & V_1: & v_1(A,B,C) & \text{:- } e(A,B), e(B,C), e(A,C) \\ \mathcal{W} = \{W_1\}, & W_1: & w_1(A,B,C) & \text{:- } e(A,B), e(B,C) \end{array}$$

As illustrated by Figure 1, view V_1 stores all the subgraphs shown in Figure 1(a), while view W_1 stores all the subgraphs shown in Figure 1(b). Although, the two views do not have projections in their definitions, $\mathcal{V} \sqsubseteq_v \mathcal{W}$, and $\mathcal{V} \not\preceq_p \mathcal{W}$, since V_1 cannot be answered using W_1. □



(a) View V_1 (b) View W_1

Figure 1: Diagram for the two views in Example 3.2

The following example shows that p-containment does not imply v-containment, even if the views in the sets have the same schemas.

EXAMPLE 3.3 Let $r(X_1, X_2)$ and $s(Y_1, Y_2)$ be two base relations on which two view sets are defined:

$$\begin{array}{llll} \mathcal{V} = \{V_1\}, & V_1: & v_1(A,C) & \text{:- } r(A,B), s(B,C) \\ \mathcal{W} = \{W_1, W_2\}, & W_1: & w_1(A,B) & \text{:- } r(A,B) \\ & W_2: & w_2(B,C) & \text{:- } s(B,C) \end{array}$$

Clearly $\mathcal{V} \not\sqsubseteq_v \mathcal{W}$, but $\mathcal{V} \preceq_p \mathcal{W}$, since there is a rewriting of V_1 using $\mathcal{W}$: $v_1(A,C)$:- $w_1(A,B), w_2(B,C)$. □

4 Minimizing a view set without losing power

In this section, we discuss how to minimize a view set without losing its power to answer queries. We show that a view set can have multiple equipotent minimal subsets, and give a condition that guarantees the uniqueness of the minimum. We also provide heuristics for speeding up the process for finding an equipotent minimal subset.

4.1 Equipotent minimal subsets of views

Definition 4.1 (**equipotent minimal subsets**) A subset M of a view set $\mathcal{V}$ is an *equipotent minimal subset* (EMS for short) of $\mathcal{V}$ if $M \asymp_p \mathcal{V}$, and for any $V \in M$: $M - \{V\} \not\asymp_p \mathcal{V}$. □

Informally, an equipotent minimal subset of a view set $\mathcal{V}$ is a minimal subset that is as powerful as $\mathcal{V}$. For instance, in Example 1.1, the view set $\{V_1, V_2\}$ is an EMS of $\{V_1, V_2, V_3\}$. We can compute an EMS of $\mathcal{V}$ using the following Shrinking algorithm.

> Algorithm Shrinking initially sets $M = \mathcal{V}$. For each view $V \in M$, it checks if V is answerable by the views $M - \{V\}$. If so, it removes V from M. It repeats this process until no more views can be removed from M, and returns the resulting M as an EMS of $\mathcal{V}$.

It can be shown that the problem of finding an EMS is $\mathcal{NP}$-hard. The following example shows that, as suspected, a view set may have multiple EMS's.

EXAMPLE 4.1 Suppose $r(A, B)$ is a base relation, on which the following three views are defined:

$$\begin{array}{lll} V_1: & v_1(A) & \text{:- } r(A,B) \\ V_2: & v_2(B) & \text{:- } r(A,B) \\ V_3: & v_3(A,B) & \text{:- } r(A,X), r(Y,B) \end{array}$$

Let $\mathcal{V} = \{V_1, V_2, V_3\}$. As shown by the following rewritings, $\mathcal{V}$ has two EMS's: $\{V_1, V_2\}$, and $\{V_3\}$.

$$\begin{array}{llll} \text{rewrite } V_1 \text{ using } V_3: & P_1: & v_1(A) & \text{:- } v_3(A,B) \\ \text{rewrite } V_2 \text{ using } V_3: & P_2: & v_2(B) & \text{:- } v_3(A,B) \\ \text{rewrite } V_3 \text{ using } \{V_1, V_2\}: & P_3: & v_3(A,B) & \text{:- } v_1(A), v_2(B) \end{array}$$

For instance, P_3 is an equivalent rewriting of V_3 because its expansion, $v_3(A,B)$:- $r(A,B'), r(A',B)$, can be shown equivalent to V_3. □

4.2 Uniqueness guarantee for EMS

In many applications, each view is associated with a cost, such as its storage space or the number of web pages that need to be crawled for the view [CGM00]. We often need to find an EMS that is optimal, i.e., the total cost of selected views is minimum. However, since views may have multiple EMS's, we want to know in which cases the algorithm Shrinking will not be stuck at a local minima. The following theorem shows a property of a view set that can guarantee the uniqueness of EMS.

Theorem 4.1 *A view set $\mathcal{V}$ has a unique EMS if and only if the following property holds. For any subset X of $\mathcal{V}$, for any two views V_1 and V_2 in X, if $X - \{V_1\} \asymp_p X - \{V_2\} \asymp_p \mathcal{V}$, then $X - \{V_1, V_2\} \asymp_p \mathcal{V}$. Intuitively, uniqueness of EMS holds iff for any subset of $\mathcal{V}$, we can remove two views if each of them is redundant.* □

The proof of the theorem is in the appendix. In cases where a view set $\mathcal{V}$ has multiple EMS's, the following heuristics can help us find its EMS's efficiently.

1. If there is a view $V \in \mathcal{V}$ such that query V cannot be answered by other views in $\mathcal{V}$, then V must be in all the EMS's of $\mathcal{V}$. In particular, if a view V uses a relation that is not used by any other views, then V cannot be answered by other views in $\mathcal{V}$. Thus, V is in all the EMS's of $\mathcal{V}$.
2. Let $\mathcal{F}$ be a family of views that has the following property: each view in $\mathcal{F}$ involves only one subgoal, and no variables in the body of the view are projected out in the head. That is, only selections are allowed in the views of $\mathcal{F}$. For two view sets $\mathcal{V}$ and $\mathcal{W}$ of $\mathcal{F}$, we can show that $\mathcal{V} \sqsubseteq_v \mathcal{W}$ implies $\mathcal{V} \preceq_p \mathcal{W}$, and vice-versa. Therefore, given a view set $\mathcal{V} \subseteq \mathcal{F}$, it must have a unique EMS, which can be computed by minimizing $\mathcal{V}$ using the traditional minimization techniques in [SY80].

5 Testing p-containment relative to a query set

Till now, we have considered p-containment between two view sets with respect to a "universal" set of queries, i.e., users can ask *any* query on the base relations. However, in some scenarios, users are restricted in the queries they can ask. In this section, we consider the relationship between two view sets with respect to a given set of queries. In particular, we consider infinite query sets defined by finite, parameterized queries.

5.1 Relative p-containment

Definition 5.1 (**relative p-containment**) Given a (possibly infinite) set of queries $\mathcal{Q}$, a view set $\mathcal{V}$ is p-contained in a view set $\mathcal{W}$ w.r.t. $\mathcal{Q}$, denoted by $\mathcal{V} \preceq_{\mathcal{Q}} \mathcal{W}$, if and only if for any query $Q \in \mathcal{Q}$ that is answerable by $\mathcal{V}$, Q is also answerable by $\mathcal{W}$. The two view sets are equipotent w.r.t. $\mathcal{Q}$, denoted by $\mathcal{V} \asymp_{\mathcal{Q}} \mathcal{W}$, if $\mathcal{V} \preceq_{\mathcal{Q}} \mathcal{W}$ and $\mathcal{W} \preceq_{\mathcal{Q}} \mathcal{V}$. □

EXAMPLE 5.1 Assume we have two relations $car(Make, Dealer)$ and $loc(Dealer, City)$ that store information about cars, their dealers, and the cities where the dealers are located. Consider the following two queries and three views:

$$\begin{array}{llll}
\text{Queries:} & Q_1\text{:} & q_1(D,C) & \text{:- } car(toyota,D), loc(D,C) \\
 & Q_2\text{:} & q_2(D,C) & \text{:- } car(honda,D), loc(D,C) \\
\text{Views:} & W_1\text{:} & w_1(D,C) & \text{:- } car(toyota,D), loc(D,C) \\
 & W_2\text{:} & w_2(D,C) & \text{:- } car(honda,D), loc(D,C) \\
 & W_3\text{:} & w_3(M,D,C) & \text{:- } car(M,D), loc(D,C)
\end{array}$$

Let $\mathcal{Q} = \{Q_1, Q_2\}$, $\mathcal{V} = \{W_1, W_2\}$, and $\mathcal{W} = \{W_3\}$. Then $\mathcal{V}$ and $\mathcal{W}$ are equipotent w.r.t. $\mathcal{Q}$, since the following equivalent rewritings show that both Q_1 and Q_2 can be answered by $\mathcal{V}$ as well as $\mathcal{W}$, respectively.

$$\begin{array}{llll}
\text{Rewritings using } \mathcal{V}\text{:} & Q_1\text{:} & q_1(D,C) & \text{:- } w_1(D,C) \\
 & Q_2\text{:} & q_2(D,C) & \text{:- } w_2(D,C) \\
\text{Rewritings using } \mathcal{W}\text{:} & Q_1\text{:} & q_1(D,C) & \text{:- } w_3(toyota,D,C) \\
 & Q_2\text{:} & q_2(D,C) & \text{:- } w_3(honda,D,C)
\end{array}$$

Note that $\mathcal{V}$ and $\mathcal{W}$ are not equipotent in general. □

Given a view set $\mathcal{V}$ and a query set $\mathcal{Q}$, we define an *equipotent minimal subset* (EMS) of $\mathcal{V}$ w.r.t. $\mathcal{Q}$ as follows. A subset M of $\mathcal{V}$ is an EMS of $\mathcal{V}$ w.r.t. $\mathcal{Q}$ if $M \asymp_{\mathcal{Q}} \mathcal{V}$, and for any $V \in M$: $M - \{V\} \not\asymp_{\mathcal{Q}} \mathcal{V}$. We can compute an EMS of $\mathcal{V}$ w.r.t. $\mathcal{Q}$ in the same way as in Section 4.1, if we have a method to test relative p-containment. This testing is straightforward when $\mathcal{Q}$ is finite. By definition, we can check for each query $Q_i \in \mathcal{Q}$ that is answerable by $\mathcal{V}$, whether Q_i is also answerable by $\mathcal{W}$. However, if $\mathcal{Q}$ is infinite, testing relative p-containment becomes more challenging, since we cannot use this enumerate-and-test paradigm for all the queries in $\mathcal{Q}$. In the rest of this section we consider ways to test relative p-containment w.r.t. infinite query sets generated by finite parameterized queries.

5.2 Parameterized queries

A *parameterized query* is a conjunctive query that contains *placeholders* in the argument positions of its body, in addition to constants and variables. A placeholder is denoted by an argument name beginning with a "$" sign. The following is an example.

EXAMPLE 5.2 Consider the following parameterized query Q on the two relations in Example 5.1:

$$Q : q(D) \text{ :- } car(\$M, D), loc(D, \$C)$$

This query represents all the following queries: a user gives a car make m for the placeholder $\$M$, and a city c for the placeholder $\$C$, and asks for the dealers of the make m in the city c. For instance, the following are two instances of Q:

$$\begin{array}{lll} I_1: & q(D) & \text{:- } car(toyota, D), loc(D, sf) \\ I_2: & q(D) & \text{:- } car(honda, D), loc(D, sf) \end{array}$$

which respectively ask for the dealers of Toyota and Honda in San Francisco. □

In general, each *instance* of a parameterized query Q is obtained by assigning a constant from the corresponding domain to each placeholder. If a placeholder appears in different argument positions, then the same constant must be used in these positions. Let $IS(Q)$ denote the set of all instances of the query Q. We assume that the domains of placeholders are infinite (independent of an instance of the base relations), causing $IS(Q)$ to be infinite. Thus we can represent an infinite set of queries using a finite set of parameterized queries.

EXAMPLE 5.3 Consider the following three views:

$$\begin{array}{lll} V_1: & v_1(M, D, C) & \text{:- } car(M, D), loc(D, C) \\ V_2: & v_2(M, D) & \text{:- } car(M, D), loc(D, sf) \\ V_3: & v_3(M) & \text{:- } car(M, D), loc(D, sf) \end{array}$$

Clearly, view V_1 can answer all instances of Q, since it includes information for cars and dealers in all cities. View V_2 cannot answer all instances, since it has only the information about dealers in San Francisco. But it can answer instances of the following more restrictive parameterized query, which replaces the placeholder $\$C$ by sf:

$$Q' : q(D) \text{ :- } car(\$M, D), loc(D, sf)$$

That is, the user can only ask for information about dealers in San Francisco. Finally, view V_3 cannot answer any instance of Q, since it does not have the *Dealer* attribute in its head. □

Given a finite set of parameterized queries $\mathcal{Q}$ and two view sets $\mathcal{V}$ and $\mathcal{W}$, we want to test $\mathcal{V} \preceq_{\mathcal{Q}} \mathcal{W}$. The example above suggests the following test strategy:

1. Deduce *all* instances of $\mathcal{Q}$ that can be answered by $\mathcal{V}$.
2. Test if $\mathcal{W}$ can answer *all* such instances.

In the next two subsections we show how to perform each of these steps. We show that all answerable instances of a parameterized queries for a given view set can be represented by a finite set of parameterized queries. We give an algorithm for deducing this set, and an algorithm for the second step. Although our discussion is based on one parameterized query, the results can be easily generalized to a finite set of parameterized queries.

5.3 Complete answerability of a parameterized query

We first consider the problem of testing whether all instances of a parameterized query can be answered by a view set. If all instances of a parameterized query Q can be answered by a view set $\mathcal{V}$, we say that Q is *completely answerable* by $\mathcal{V}$.

Definition 5.2 (**canonical instance**) Let Q be a parameterized query and $\mathcal{V}$ be a view set. A *canonical instance* of Q (given $\mathcal{V}$) is an instance of Q, in which each placeholder is replaced by a new distinct constant that does not appear in Q and $\mathcal{V}$. □

Theorem 5.1 *Let Q be a parameterized query, and $\mathcal{V}$ be a view set. Q is completely answerable by $\mathcal{V}$ if and only if $\mathcal{V}$ can answer a canonical instance of Q (given $\mathcal{V}$).* □

The proof is in the appendix. The theorem also suggests an algorithm TestComp for testing whether all instances of a parameterized query Q can be answered by a view set $\mathcal{V}$.

> Algorithm TestComp first constructs a canonical instance Q_c of Q (given $\mathcal{V}$). Then it tests if Q_c can be answered using $\mathcal{V}$ by calling an algorithm of answering queries using views, such as those in [LRO96, GM99a, Qia96, DG97]. It outputs "yes" if $\mathcal{V}$ can answer Q_c; otherwise, it outputs "no."

EXAMPLE 5.4 Consider the parameterized query Q in Example 5.3. To test whether view V_1 can answer all instances of Q, we use two new distinct constants m_0 and c_0 to replace the two placeholders $\$M$ and $\$C$, and obtain the following canonical instance:

$$Q_c : q(D) \text{ :- } car(m_0, D), loc(D, c_0)$$

Clearly Q_c can be answered by view V_1, because of the following equivalent rewriting of Q_c:

$$P_c : q(D) \text{ :- } v_1(m_0, D, c_0)$$

By Theorem 5.1, view V_1 can answer all instances of Q. In addition, since V_2 cannot answer Q_c (which is also a canonical instance of Q given V_2), it cannot answer *some* instances of Q. The same argument holds for V_3. □

5.4 Partial answerability of a parameterized query

As shown by the view V_2 and the query Q in Example 5.3, even if a view set cannot answer all instances of a parameterized query, it can still answer some of its instances. In general, we want to know what instances can be answered by the view set, and whether these instances can also be represented as a set of more "restrictive" parameterized queries. A parameterized query Q_1 is more *restrictive* than a parameterized query Q if every instance of Q_1 is also an instance of Q. For example, query $q(D) \text{ :- } car(\$M, D), loc(D, sf)$ is more restrictive than query $q(D) \text{ :- } car(\$M, D), loc(D, \$C)$, since the former requires the second argument of the *loc* subgoal to be sf, while the latter allows any constant for its corresponding placeholder $\$C$. For another example, query $q(M, C) \text{ :- } car(M, \$D_1), loc(\$D_1, C)$ is more restrictive than query $q(M, C) \text{ :- } car(M, \$D_1), loc(\$D_2, C)$, since the former has one placeholder in two argument positions, while the latter allows two different constants to be assigned to its two placeholders.

Clearly all the parameterized queries that are more restrictive than Q can be generated by adding the following two types of restrictions:

1. Type I: Some placeholders must be assigned the same constant. Formally, let $\{\$A_1, \ldots, \$A_k\}$ be *some* placeholders in Q. We can put a restriction $\$A_1 = \cdots = \$A_k$ on the query Q. That is, we can replace all these k placeholders with one placeholder.
2. Type II: For a placeholder $\$A_i$ in Q and a constant c in Q or $\mathcal{V}$, we put a restriction $\$A_i = c$ on Q. That is, the user can only assign constant c to this placeholder in an instance.

Consider all the possible (finite) combinations of these two types of restrictions. For example, suppose Q has two placeholders, $\{\$A_1, \$A_2\}$, and Q and $\mathcal{V}$ have one constant c. Then we consider the following restriction combinations: $\{\}$, $\{\$A_1 = \$A_2\}$, $\{\$A_1 = c\}$, $\{\$A_2 = c\}$, and $\{\$A_1 = \$A_2 = c\}$. Note that we allow a combination to have restrictions of only one type. In addition, each restriction combination is consistent, in the sense that it does not have a restriction $\$A_1 = \$A_2$ and two restrictions $\$A_1 = c_1$ and $\$A_2 = c_2$, while c_1 and c_2 are two different constants in Q and $\mathcal{V}$. For each restriction combination RC_i, let $Q(RC_i)$ be the parameterized query that is derived by adding the restrictions in RC_i to Q. Clearly $Q(RC_i)$ is a parameterized query that is more restrictive than Q. Let $\Phi(Q, \mathcal{V})$ denote all these parameterized queries that are more restrictive than Q.

Suppose I is an instance of Q that can be answered by $\mathcal{V}$. We can show that there exists a parameterized query $Q_i \in \Phi(Q, \mathcal{V})$, such that I is a canonical instance of Q_i. By Theorem 5.1, Q_i is completely answerable by $\mathcal{V}$. Therefore, we have proved the following theorem:

Theorem 5.2 *All instances of a parameterized query Q that are answerable by a view set $\mathcal{V}$ can be generated by a finite set of parameterized queries that are more restrictive than Q, such that all these parameterized queries are completely answerable by $\mathcal{V}$.* □

We propose the following algorithm GenPartial. Given a parameterized query Q and a view set $\mathcal{V}$, the algorithm generates all the parameterized queries that are more restrictive than Q, such that they are completely answerable by $\mathcal{V}$, and they generate all the instances of Q that are answerable by $\mathcal{V}$.

> Algorithm GenPartial first generates all the restriction combinations, and creates a parameterized query for each combination. Then it calls the algorithm TestComp to check if this parameterized query is completely answerable by $\mathcal{V}$. It outputs all the parameterized queries that are completely answerable by $\mathcal{V}$.

5.5 Testing p-containment relative to finite parameterized queries

Now we give an algorithm for testing p-containment relative to parameterized queries. Let $\mathcal{Q}$ be a query set with only one parameterized query Q. Let $\mathcal{V}$ and $\mathcal{W}$ be two view sets. The algorithm tests $\mathcal{V} \preceq_{\mathcal{Q}} \mathcal{W}$ as follows. First call the algorithm GenPartial to find all the more restrictive parameterized queries of Q that are completely answerable by $\mathcal{V}$. For each of them, call the algorithm TestComp to check if it is also completely answerable by $\mathcal{W}$. By definition, $\mathcal{V} \preceq_{\mathcal{Q}} \mathcal{W}$ iff all these parameterized queries that are completely answerable by $\mathcal{V}$ are also completely answerable by $\mathcal{W}$. The algorithm can be easily generalized to the case where $\mathcal{Q}$ is a finite set of parameterized queries.

6 MCR-containment

So far we have considered the query-answering power of views with respect to equivalent rewritings of queries. In information-integration systems, we often need to consider maximally-contained rewritings of a query using views. In this section, we introduce the concept of *MCR-containment*, which describes the relative power of two view sets in terms of their maximally-contained rewritings of queries. Surprisingly, MCR-containment is essentially the same as p-containment.

Definition 6.1 (maximally-contained rewritings) P is a *maximally-contained rewriting* of a query Q using a set of views $\mathcal{V}$ if the following hold: (1) P is a finite union of conjunctive queries using only the views in $\mathcal{V}$; (2) For any database, the answer computed by P is a subset of the answer to Q; and (3) No other unions of conjunctive queries that satisfy the two conditions above can properly contain P. □

Intuitively, a maximally-contained rewriting (henceforth called MCR for short) is a plan that uses only the views in $\mathcal{V}$ and computes the maximal answer to the query Q. If Q has two MCR's, by definition, they must be equivalent. It is known [Li99] that when arbitrary datalog programs are considered to compute the maximal answer to a conjunctive query, we can deduce a *finite* union of conjunctive queries as an MCR. That is, we do not need to consider datalog rewritings. Clearly if Q is answerable by $\mathcal{V}$, then an equivalent rewriting of Q using $\mathcal{V}$ is also an MCR of Q.

EXAMPLE 6.1 Consider the following query Q and view V on the two relations in Example 5.1:

$$\begin{array}{lll} Q: & q(M,D,C) & \text{:- } car(M,D), loc(D,C) \\ V: & v(M,D,sf) & \text{:- } car(M,D), loc(D,sf) \end{array}$$

Suppose we have the access to view V only. Then $q(M,D,sf)$:- $v(M,D,sf)$ is an MCR of the query Q using the view V. That is, we can give the user only the information about car dealers in San Francisco as an answer to the query, but not anything more. □

Definition 6.2 (**MCR-containment**) A view set $\mathcal{V}$ is *MCR-contained* in another view set $\mathcal{W}$, denoted by $\mathcal{V} \preceq_{MCR} \mathcal{W}$, if for any query Q, we have $MCR(Q,\mathcal{V}) \sqsubseteq MCR(Q,\mathcal{W})$, where $MCR(Q,\mathcal{V})$ and $MCR(Q,\mathcal{W})$ are MCR's of Q using $\mathcal{V}$ and $\mathcal{W}$, respectively.[4] The two sets are *MCR-equipotent*, denoted by $\mathcal{V} \asymp_{MCR} \mathcal{W}$, if $\mathcal{V} \preceq_{MCR} \mathcal{W}$, and $\mathcal{W} \preceq_{MCR} \mathcal{V}$. □

The following theorem shows that MCR-containment is essentially the same as p-containment.

Theorem 6.1 *For two view sets $\mathcal{V}$ and $\mathcal{W}$, $\mathcal{V} \preceq_p \mathcal{W}$ if and only if $\mathcal{V} \preceq_{MCR} \mathcal{W}$.* □

Proof: "If": Suppose $\mathcal{V} \preceq_{MCR} \mathcal{W}$. Consider each view $V \in \mathcal{V}$. Clearly V itself is an MCR of the query V using $\mathcal{V}$, since it is an equivalent rewriting of V. Let $MCR(V,\mathcal{W})$ be an MCR of V using $\mathcal{W}$. Since $\mathcal{V} \preceq_{MCR} \mathcal{W}$, we have $V \sqsubseteq MCR(V,\mathcal{W})$. On the other hand, by the definition of MCR's, $MCR(V,\mathcal{W}) \sqsubseteq V$. Thus $MCR(V,\mathcal{W})$ and V are equivalent, and $MCR(V,\mathcal{W})$ is an equivalent rewriting of V using $\mathcal{W}$. By Theorem 3.1, $\mathcal{V} \preceq_p \mathcal{W}$.

"Only if": Suppose $\mathcal{V} \preceq_p \mathcal{W}$. By Theorem 3.1, every view has an equivalent rewriting using $\mathcal{W}$. For any query Q, let $MCR(Q,\mathcal{V})$ and $MCR(Q,\mathcal{W})$ be MCR's of Q using $\mathcal{V}$ and $\mathcal{W}$, respectively. We replace each view in $MCR(Q,\mathcal{V})$ with its corresponding rewriting using $\mathcal{W}$, and obtain a new rewriting MCR' of query Q using $\mathcal{W}$, which is equivalent to $MCR(Q,\mathcal{V})$. By the definition of MCR's, we have $MCR' \sqsubseteq MCR(Q,\mathcal{W})$. Thus $MCR(Q,\mathcal{V}) \sqsubseteq MCR(Q,\mathcal{W})$, and $\mathcal{V} \preceq_{MCR} \mathcal{W}$. ∎

7 Minimizing view sets of general queries

Until now, we have discussed minimizations of conjunctive views using the concept of p-containment. In this section we extend the earlier results to more general queries, and propose a framework for minimizing view sets without losing their query-answering power.

[4] We extend the query-containment notation "$\sqsubseteq$" in Definition 2.1 to unions of conjunctive queries in the obvious way.

Let $\mathcal{F}$ be one of the following three families of queries: conjunctive queries with arithmetic comparisons, unions of conjunctive queries, and datalog queries. Suppose $Q \in \mathcal{F}$ is a query on base relations, and $\mathcal{V}$ is a set of views in $\mathcal{F}$ on the same base relations.

Definition 7.1 (**general answerability**) We say Q is *answerable* by $\mathcal{V}$ if there is a query $P \in \mathcal{F}$ that uses only the views in $\mathcal{V}$, such that for any database D of the base relations, the answer computed by P on the instance of the views $\mathcal{V}(D)$ is equivalent to the answer computed by Q on the base relations. □

Definition 7.2 (**general p-containment and equipotence**) Let $\mathcal{V}$ and $\mathcal{W}$ be two sets of views in $\mathcal{F}$. We say $\mathcal{V}$ is *p-contained* in $\mathcal{W}$, denoted by $\mathcal{V} \preceq_p \mathcal{W}$, if any query in $\mathcal{F}$ that is answerable by $\mathcal{V}$ is also answerable by $\mathcal{W}$. The two view sets are *equipotent*, denoted by $\mathcal{V} \asymp_p \mathcal{W}$, if $\mathcal{V} \preceq_p \mathcal{W}$, and $\mathcal{W} \preceq_p \mathcal{V}$. □

7.1 Testing general p-containment

Theorem 7.1 *Under the definition of general p-containment, Theorem 3.1 holds. That is, for two sets of views $\mathcal{V}$ and $\mathcal{W}$, $\mathcal{V} \preceq_p \mathcal{W}$ if and only if for every view $V \in \mathcal{V}$, V is answerable by $\mathcal{W}$.* □

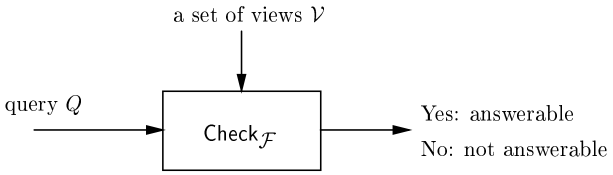


Figure 2: The algorithm $\mathsf{Check}_{\mathcal{F}}$ for a family $\mathcal{F}$ of queries.

Suppose we have an algorithm $\mathsf{Check}_{\mathcal{F}}$ that can test the *answerability* of a query with respect to a set of views in the family $\mathcal{F}$. That is, given a query Q and a view set $\mathcal{V}$, as shown in Figure 2, the algorithm $\mathsf{Check}_{\mathcal{F}}$ tests if Q is answerable by $\mathcal{V}$. By Theorem 7.1, we can test $\mathcal{V} \preceq_p \mathcal{W}$ as follows: for each view $V \in \mathcal{V}$, call the algorithm $\mathsf{Check}_{\mathcal{F}}$ to test whether V is answerable by $\mathcal{W}$. $\mathcal{V} \preceq_p \mathcal{W}$ is true iff all the views in $\mathcal{V}$ pass the test. It should be observed here, that algorithm $\mathsf{Check}_{\mathcal{F}}$ does not exist for the family of datalog queries [Dus97].

7.2 Minimizing view sets using general equipotence

Given a view set $\mathcal{V}$, similarly to Definition 4.1, an equipotent minimal subset (EMS) of $\mathcal{V}$ is a minimal subset of $\mathcal{V}$ that is equipotent to $\mathcal{V}$. To compute an EMS of $\mathcal{V}$, we check for each view $V \in \mathcal{V}$ if V is answerable by $\mathcal{V} - \{V\}$ by calling the algorithm $\mathsf{Check}_{\mathcal{F}}$. If so, remove V from $\mathcal{V}$. We keep shrinking the view set until no views can be removed without changing the query-answering power, and the final set is an EMS of $\mathcal{V}$. As shown in Example 4.1, a view set can have multiple EMS's.

In some scenarios we need to answer queries using views in the presence of constraints (e.g., functional dependencies [Cod70], multivalued dependencies [Fag77, Del78]) and binding limitations on views [RSU95, DL97, Ull89, LC00, LYV^{+}98, YLUGM99]. We can generalize our results by requiring that the algorithm $\mathsf{Check}_{\mathcal{F}}$ take constraints and binding limitations into consideration. [RSU95, Gry99] give algorithms for answering conjunctive queries using conjunctive views in these scenarios; [Dus97] provides answers for rewriting datalog queries using conjunctive views.

8 Conclusion

In this paper we showed that when minimizing a set of views, we should consider the query-answering power of the views, rather than using the traditional query-containment concept for selection of views. We developed the concept of p-containment, and showed that it is not related to the traditional query containment. We discussed how to minimize a view set without losing its query-answering power. We also considered p-containment of two view sets with respect to a given (possibly infinite) set of queries, and with respect to maximally-contained rewritings of queries using the views. We developed algorithms for testing these containments, and discussed their relationships. Currently we are working on some open problems in our framework, including ways to find an EMS of a view set efficiently, to find a v-equivalent minimal subset of a view set efficiently, and to find cases where v-containment can imply p-containment, and vice-versa.

Acknowledgments: We thank Arvind Arasu for his valuable comments on this material.

References

[AD98] Serge Abiteboul and Oliver M. Duschka. Complexity of answering queries using materialized views. In *Proc. of ACM Symposium on Principles of Database Systems (PODS)*, pages 254–263, 1998.

[AGK99] Foto N. Afrati, Manolis Gergatsoulis, and Theodoros G. Kavalieros. Answering queries using materialized views with disjunctions. *International Conference on Database Theory (ICDT)*, pages 435–452, 1999.

[BPT97] E. Baralis, S. Paraboschi, and E. Teniente. Materialized view selection in a multidimensional database. In *Proc. of VLDB*, 1997.

[CG00] Rada Chirkova and Michael R. Genesereth. Linearly bounded reformulations of conjunctive databases. *Proceedings of the Sixth International Conference on Rules and Objects in Databases (DOOD)*, 2000.

[CGLV99] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. Query answering using views for data integration over the Web. *WebDB*, pages 73–78, 1999.

[CGM00] Junghoo Cho and Hector Garcia-Molina. Synchronizing a database to improve freshness. *Proc. of ACM SIGMOD*, 2000.

[CM77] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. *STOC*, pages 77–90, 1977.

[Cod70] E. F. Codd. A relational model of data for large shared data banks. *CACM*, 13(6):377–387, 1970.

[CW91] S. Ceri and J. Widom. Deriving production rules for incremental view maintenance. In *Proc. of VLDB*, 1991.

[Del78] Claude Delobel. Normalization and hierarchical dependencies in the relational data model. *TODS*, 3(3):201–222, 1978.

[DG97] Oliver M. Duschka and Michael R. Genesereth. Answering recursive queries using views. In *Proc. of ACM Symposium on Principles of Database Systems (PODS)*, pages 109–116, 1997.

[DL97] Oliver M. Duschka and Alon Y. Levy. Recursive plans for information gathering. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI-97*, 1997.

[Dus97] Oliver M. Duschka. Query planning and optimization in information integration. *Ph.D. Thesis, Computer Science Dept., Stanford Univ.*, 1997.

[Fag77] Ronald Fagin. Multivalued dependencies and a new normal form for relational databases. *TODS*, 2(3):262–278, 1977.

[FLSY99] Daniela Florescu, Alon Y. Levy, Dan Suciu, and Khaled Yagoub. Optimization of run-time management of data intensive web-sites. In *Proc. of VLDB*, pages 627–638, 1999.

[GHRU97] Himanshu Gupta, Venky Harinarayan, Anand Rajaraman, and Jeff Ullman. Index selection in olap. In *International Conference on Data Engineering (ICDE)*, 1997.

[GM99a] Gösta Grahne and Alberto O. Mendelzon. Tableau techniques for querying information sources through global schemas. In *International Conference on Database Theory (ICDT)*, pages 332–347, 1999.

[GM99b] Himanshu Gupta and Inderpal Mumick. Selection of views to materialize under a maintenance-time constraint. In *International Conference on Database Theory (ICDT)*, 1999.

[Gry99] Jarek Gryz. Query rewriting using views in the presence of functional and inclusion dependencies. *Information Systems*, 24(7):597–612, 1999.

[GT00] Stéphane Grumbach and Leonardo Tininini. On the content of materialized aggregate views. In *Proc. of ACM Symposium on Principles of Database Systems (PODS)*, pages 47–57, 2000.

[Gup94] Ashish Gupta. Partial information based integrity constraint checking. *Ph.D. Thesis, Computer Science Dept., Stanford Univ.*, 1994.

[HGMW$^+$95] J. Hammer, Hector Garcia-Molina, Jennifer Widom, Wilburt Labio, and Y. Zhuge. The stanford data warehousing project. In *IEEE Data Eng. Bulletin, Special Issue on Materialized Views and Data Warehousing*, 1995.

[HRU96] Venky Harinarayan, Anand Rajaraman, and Jeff Ullman. Implementing data cubes efficiently. In *Proc. of ACM SIGMOD*, 1996.

[IK93] W.H. Inmon and C. Kelley. Rdb/vms: Developing the data warehouse. In *QED Publishing Group, Boston, Massachussetts*, 1993.

[LC00] Chen Li and Edward Chang. Query planning with limited source capabilities. *International Conference on Data Engineering (ICDE)*, pages 401–412, 2000.

[Lev00] Alon Y. Levy. Answering queries using views: A survey. In `http://www.cs.washington.edu/homes/alon/site/files/view-survey.ps`, 2000.

[Li99] Chen Li. Testing query containment in the presence of limited access patterns. *Technical report, Computer Science Dept., Stanford Univ.*, 1999.

[LMSS95] Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. In *Proc. of ACM Symposium on Principles of Database Systems (PODS)*, pages 95–104, 1995.

[LRO96] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of VLDB*, pages 251–262, 1996.

[LYV$^+$98] Chen Li, Ramana Yerneni, Vasilis Vassalos, Hector Garcia-Molina, Yannis Papakonstantinou, Jeffrey D. Ullman, and Murty Valiveti. Capability based mediation in TSIMMIS. In *Proc. of ACM SIGMOD*, pages 564–566, 1998.

[Mit99] Prasenjit Mitra. An algorithm for answering queries efficiently using views. In *Technical Report, Stanford University*, 1999.

[PL00] Rachel Pottinger and Alon Levy. A scalable algorithm for answering queries using views. In *Proc. of VLDB*, 2000.

[Qia96] Xiaolei Qian. Query folding. *International Conference on Data Engineering (ICDE)*, pages 48–55, 1996.

[RSS96] Kenneth A. Ross, Divesh Srivastava, and S. Sudarshan. Materialized view maintenance and integrity constraint checking: Trading space for time. In *Proc. of ACM SIGMOD*, 1996.

[RSU95] Anand Rajaraman, Yehoshua Sagiv, and Jeffrey D. Ullman. Answering queries using templates with binding patterns. In *Proc. of ACM Symposium on Principles of Database Systems (PODS)*, pages 105–112, 1995.

[SY80] Yehoshua Sagiv and Mihalis Yannakakis. Equivalences among relational expressions with the union and difference operators. *Journal of the ACM*, 27(4):633–655, 1980.

[TS97] D. Theodoratos and T. Sellis. Data warehouse configuration. In *Proc. of VLDB*, 1997.

[Ull89] Jeffrey D. Ullman. *Principles of Database and Knowledge-base Systems, Volumes II: The New Technologies.* Computer Science Press, New York, 1989.

[Ull97] Jeffrey D. Ullman. Information integration using logical views. *International Conference on Database Theory (ICDT)*, pages 19–40, 1997.

[Wid95] Jennifer Widom. Research problems in data warehousing. In *Proc. of the Intl. Conf. on Information and Knowledge Management*, 1995.

[Wie92] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.

[YKL97] J. Yang, K. Karlapalem, and Q. Li. Algorithms for materialized view design in data warehousing environment. In *Proc. of VLDB*, 1997.

[YLUGM99] Ramana Yerneni, Chen Li, Jeffrey D. Ullman, and Hector Garcia-Molina. Optimizing large join queries in mediation systems. *International Conference on Database Theory (ICDT)*, pages 348–364, 1999.

A Appendix

A.1 The proof of Theorem 3.1

Theorem A.1 *Let $\mathcal{V}$ and $\mathcal{W}$ be two view sets. $\mathcal{V} \preceq_p \mathcal{W}$ iff for every view $V \in \mathcal{V}$, if treated as a query, V is answerable by $\mathcal{W}$.* □

Proof: "If": Clearly, each view $V \in \mathcal{V}$, if treated as a query, can be answered using the view V itself. By the definition of $\mathcal{V} \preceq_p \mathcal{W}$, view V is also answerable by $\mathcal{W}$. "Only If": For any query Q that is answerable using $\mathcal{V}$, assume Q can be written as

$$ans(\bar{X}) \text{ :- } v_{i_1}(\bar{X}_1) \ \& \ \cdots \ \& \ v_{i_k}(\bar{X}_k)$$

where each v_{i_j} is the head of a view V_{i_j} in $\mathcal{V}$, which is equivalent to some conjunctive query using the views in $\mathcal{W}$. Unify $v_{i_j}(X_j)$ with the head of that conjunctive query to get a body that replaces $v_{i_j}(X_j)$. Then Q is equivalent to the resulting conjunctive query using the views in $\mathcal{W}$. ■

A.2 The proof of Theorem 4.1

Theorem A.2 *A view set $\mathcal{V}$ has a unique EMS if and only if the following property holds. For any subset X of $\mathcal{V}$, for any two views V_1 and V_2 in X, if $X - \{V_1\} \asymp_p X - \{V_2\} \asymp_p \mathcal{V}$, then $X - \{V_1, V_2\} \asymp_p \mathcal{V}$. Intuitively, uniqueness of EMS holds iff for any subset of $\mathcal{V}$, we can remove two views if each of them is redundant.* □

Proof: "If": Assume $\mathcal{V}$ has a unique EMS M. For any $X \subseteq \mathcal{V}$, any view $V \in X$, if $X - \{V\} \asymp_p \mathcal{V}$, then $M \subseteq X - \{V\}$. Otherwise, we can "shrink" the set of views $X - \{V\}$ to obtain another EMS, which is different from M. Therefore, for any two views V_1 and V_2 in X, if $X - \{V_1\} \asymp_p X - \{V_2\} \asymp_p \mathcal{V}$, then $M \subseteq X - \{V_1\}$ and $M \subseteq X - \{V_2\}$. Thus $M \subseteq X - \{V_1, V_2\}$, and we have $X - \{V_1, V_2\} \asymp_p \mathcal{V}$.

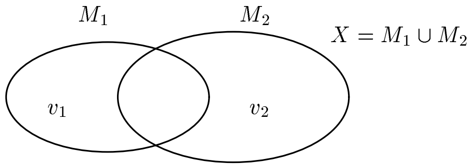


Figure 3: Diagram for the proof of Theorem 4.1.

"Only If": Suppose $X - \{V_1\} \asymp_p X - \{V_2\} \asymp_p \mathcal{V}$ implies $X - \{V_1, V_2\} \asymp_p \mathcal{V}$ for any subset X of $\mathcal{V}$. Assume $\mathcal{V}$ has two different EMS's M_1 and M_2. Since both are minimal, as shown in Figure 3, there exists a view $V_1 \in M_1 - M_2$. Let $X = M_1 \cup M_2$. Since $M_1 \neq M_2$, and $M_2 \not\subseteq M_1$ (since M_1 is minimal), there exists $V_2 \in X - M_1$. We can show that $X - \{V_1\} \asymp_p X - \{V_2\} \asymp_p \mathcal{V}$, so $X - \{V_1, V_2\} \asymp_p \mathcal{V}$. Remove V_2 from X. We repeat this process by removing redundant views in $M_2 - M_1$ from X, and then prove that $M_1 - \{V_1\}$ is also an EMS of $\mathcal{V}$. Then M_1 cannot be minimal. ■

A.3 The proof of Theorem 5.1

Theorem A.3 *Let Q be a parameterized query Q and $\mathcal{V}$ be a view set. Q is completely answerable by $\mathcal{V}$ if and only if $\mathcal{V}$ can answer a canonical instance of Q given $\mathcal{V}$.* □

Proof: Let Q_c be a canonical instance of Q given $\mathcal{V}$. The "Only if" part is obvious, since Q_c is an instance of Q. To prove the "If" part, we need to show that if there exists a rewriting P_c of the instance Q_c using $\mathcal{V}$, then for any instance I of query Q, there exists a rewriting P_I of I using $\mathcal{V}$. As shown in Figure 4, the rewriting P_I is constructed as follows. For each placeholder $\$A_i$ in Q, suppose it is replaced by a_i in Q_c, and by b_i in the instance I. Replace each a_i in the rewriting P_c with b_i, and we do this replacement for every placeholder in Q. Let P_I be the new rewriting after the replacements.

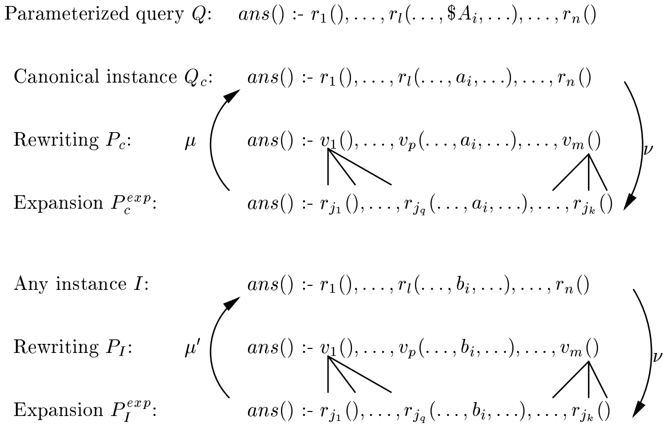

Parameterized query Q: $ans()$:- $r_1(), \ldots, r_l(\ldots, \$A_i, \ldots), \ldots, r_n()$
Canonical instance Q_c: $ans()$:- $r_1(), \ldots, r_l(\ldots, a_i, \ldots), \ldots, r_n()$
Rewriting P_c: $ans()$:- $v_1(), \ldots, v_p(\ldots, a_i, \ldots), \ldots, v_m()$
μ
ν
Expansion P_c^{exp}: $ans()$:- $r_{j_1}(), \ldots, r_{j_q}(\ldots, a_i, \ldots), \ldots, r_{j_k}()$
Any instance I: $ans()$:- $r_1(), \ldots, r_l(\ldots, b_i, \ldots), \ldots, r_n()$
Rewriting P_I: $ans()$:- $v_1(), \ldots, v_p(\ldots, b_i, \ldots), \ldots, v_m()$
μ'
ν'
Expansion P_I^{exp}: $ans()$:- $r_{j_1}(), \ldots, r_{j_q}(\ldots, b_i, \ldots), \ldots, r_{j_k}()$

Figure 4: Diagram for the proof of Theorem 5.1.

Now we prove that P_I is an equivalent rewriting of I using $\mathcal{V}$. Consider the expansion P_c^{exp} of the rewriting P_c. Since P_c^{exp} is equivalent to Q_c, there exist a containment mapping ν from Q_c to P_c^{exp}, and a containment mapping μ of the other direction. Note that a containment mapping must map a constant to the same constant, and map a variable to either a constant, or another variable. In addition, the expansion P_I^{exp} of P_I can be obtained from P_c^{exp} by replacing every occurrence of a_i with b_i.

We can construct a new mapping ν' from I to P_I^{exp} as follows. ν' is the same as ν except that for each constant b_i in I that replaces a placeholder $\$A_i$ of Q, we let $\nu'(b_i) = b_i$. By the construction of P_c and P_I, we can show that ν' is a containment mapping from I to P_I^{exp}. Similarly we can derive from μ a containment mapping μ' from P_I^{exp} to I. Thus P_I is an equivalent rewriting of I using $\mathcal{V}$. ∎