Answering Queries Using Views in the Presence of Functional Dependencies

Jun Hong[1], Weiru Liu[1], David Bell[1], and Qingyuan Bai[2]

[1] School of Computer Science, Queen's University Belfast, Belfast BT7 1NN, UK
[2] School of Computing and Mathematics, University of Ulster, Newtownabbey, Co. Antrim BT37 0QB, UK

Abstract. This paper is concerned with the problem of answering queries using views in the presence of functional dependencies. Previous algorithms for answering queries using views, such as the MiniCon algorithm, have not taken into account the presence of functional dependencies. As a consequence, these algorithms may miss query rewritings in the presence of such dependencies. In this paper, we present an extension of the MiniCon algorithm to handle the presence of functional dependencies while still retaining the main properties of the algorithm and its computational advantage over the other algorithms.

1 Introduction

Data integration from multiple disparate data sources over the Internet has recently attracted a lot of attention in both the database and AI communities [1–4]. Data integration deals with pre-existing and autonomous data sources that have been created independently. It aims to provide a uniform interface to the underlying data sources, which allows users to make queries using the interface in terms of a mediated schema rather than interacting directly with the relevant sources using their individual schemas and combining the data from them. One main stage of data integration is query reformulation in which a user query over the mediated schema is reformulated into queries over the data-source schemas. A typical approach to query reformulation is called *local as view*, in which data sources are described by views over the mediated schema. The objective of query reformulation in this approach is to reformulate the user query using the given views (data source descriptions). The problem of query reformulation using the local-as-view approach is closely related to the broader problem of answering queries using views.

In this paper, we consider the problem of answering conjunctive queries using a large set of conjunctive views in the presence of functional dependencies. In the context of data integration, a number of algorithms, such as the MiniCon algorithm [5], have been developed for query reformulation. However, the presence of functional dependencies in the mediated schema has not been taken into account in these algorithms. As a consequence, these algorithms may miss query rewritings in the presence of such dependencies.

Example 1. Consider the following mediated schema that is used throughout this paper: $student(S, P, Y)$, $taught(P, D)$, and $program(P, C)$. The $student$ relation describes

M. Jackson et al. (Eds.): BNCOD 2005, LNCS 3567, pp. 70–81, 2005.

the degree program P a student S takes and the year Y the student is in. The $taught$ relation shows the department D in which a degree program P is taught. The $program$ relation states the program code C of a degree program P. In the mediated schema, we also assume that a student takes only one degree program and is in a specific year, a degree program is taught in only one department, and a degree program has a unique program code. We therefore have the following functional dependencies in the mediated schema: $student : S \rightarrow P$, $S \rightarrow Y$; $taught : P \rightarrow D$; $program : P \rightarrow C$.

Suppose we have three data sources described by three views:
$v_1(S', Y', D')$:- $student(S', P', Y'), taught(P', D')$.
$v_2(S', P')$:- $student(S', P', Y')$.
$v_3(P', C')$:- $program(P', C')$.

v_1, v_2 and v_3 provide data showing the year and department a student is in, the degree program a student takes, and the program code of a degree program, respectively.

Assume that a user asks which degree program a student takes and in which year the student is: $q(S, P, Y)$:- $student(S, P, Y)$. The following is a correct rewriting of the query: $q'(S', P', Y')$:- $v_1(S', Y', D'), v_2(S', P')$. The rewriting is correct only because the functional dependencies $S \rightarrow P$ and $S \rightarrow Y$ hold in the mediated schema.

Example 2. Suppose that a user asks in which department the degree program $cs401$ is taught: $q(D)$:- $taught(P, D), program(P, C), C = cs401$. The following is a correct rewriting of the query:

$q'(D')$:- $v_1(S', Y', D'), v_2(S', P'), v_3(P', C'), C' = cs401$.

The rewriting is correct only because the functional dependencies, $S \rightarrow P$ and $P \rightarrow D$, hold in the mediated schema. In particular, we have the transitive functional dependency, $S \rightarrow D$.

The previous algorithms for answering queries using views, such as the MiniCon algorithm, however fail to generate the above two rewritings since they do not take into account the presence of functional dependencies in the mediated schema. In this paper, we present an extension of the MiniCon algorithm for answering queries using views in the presence of functional dependencies. The extended MiniCon algorithm retains the main properties of the MiniCon algorithm and its computational advantage over the other algorithms. The paper is organised as follows. Section 2 describes the notation used in the paper and formally defines the problem. Section 3 gives a brief review of the MiniCon algorithm. Section 4 describes our extension of the MiniCon algorithm. Section 5 briefly discusses related work. We finally conclude in Section 6.

2 Preliminaries

Definition 1. (Mediated Schema, Query and View) *A mediated schema consists of a set of database relations over which user queries can be made and views describing data sources can be defined. A query is a conjunctive query of the form:*

$q(\overline{X})$:- $r_1(\overline{X}_1)$,..., $r_n(\overline{X}_n)$

over the mediated schema, where $\overline{X}, \overline{X}_1, ..., \overline{X}_n$ *are tuples containing either variables or constants and* $\overline{X} \subseteq \overline{X}_1 \cup ... \cup \overline{X}_n$. *The variables in* $\overline{X}$ *are the distinguished variables of the query and all the other variables are existential variables. A view is a named query describing a data source.*

Definition 2. (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 instance D, the answer of evaluating Q_1 over D, $Q_1(D)$, is a subset of the answer of evaluating Q_2 over D, $Q_2(D)$, that is $Q_1(D) \subseteq Q_2(D)$. Q_1 is equivalent to Q_2, denoted by $Q_1 \equiv Q_2$, if $Q_1 \sqsubseteq Q_2$ and $Q_2 \sqsubseteq Q_1$.*

Definition 3. (Contained Rewriting and Equivalent Rewriting) *Let Q be a query over a mediated schema, $\mathcal{V} = V_1, ..., V_n$ be a set of views over the same mediated schema, and $\mathcal{L}$ be a query language. The query Q' in $\mathcal{L}$ using $\mathcal{V}$ is a contained rewriting of Q if $Q'(\mathcal{V}) \sqsubseteq Q$, and an equivalent rewriting of Q if $Q'(\mathcal{V}) \equiv Q$.*

In the context of data integration, since data sources are often pre-existing and autonomous and have been created independently, it is often not possible for us to generate an equivalent rewriting of a user query. Instead we want to be able to generate the maximally-contained rewriting that provides all the possible answers from a given set of data sources.

Definition 4 (Maximally-Contained Rewriting). *Let Q be a query over a mediated schema, $\mathcal{V} = V_1, ..., V_n$ be a set of views over the same mediated schema, and $\mathcal{L}$ be a query language. The query Q' in $\mathcal{L}$ using $\mathcal{V}$ is a maximally-contained rewriting of Q if (1) $Q'(\mathcal{V}) \sqsubseteq Q$, and (2) there is no query Q'' in $\mathcal{L}$ using $\mathcal{V}$ that is not equivalent to Q', such that $Q'(\mathcal{V}) \sqsubseteq Q''(\mathcal{V}) \sqsubseteq Q$.*

Definition 5 (Functional Dependencies). *A functional dependency $r : a_1, ..., a_n \rightarrow b$ in the mediated schema, where $a_1, ..., a_n$ and b refer to attributes in the relation r, states that for every two tuples t and u in r if $t.a_i = u.a_i$ for $i = 1, ..., n$, then $t.b = u.b$.*

In the presence of functional dependencies in the mediated schema, query containment, query equivalence, contained rewritings, equivalent rewritings, and maximally-contained rewritings can be defined accordingly, taking into account the presence of such dependencies. For simplicity of the paper, we do not introduce any new notation to denote these. But whenever we talk about contained rewritings and maximally-contained rewritings, we always make it clear whether the presence of functional dependencies in the mediated schema has been taken into account.

The Problem: Given a conjunctive query Q over the mediated schema with a set of functional dependencies $\mathcal{F}$, and a set of conjunctive views $\mathcal{V} = V_1, ..., V_n$ also over the mediated schema describing a set of data sources $\mathcal{S} = S_1, ..., S_n$, the problem of answering conjunctive queries using conjunctive views in the presence of functional dependencies is to generate every conjunctive query Q' over $\mathcal{V}$, which is a contained rewriting of Q in the presence of $\mathcal{F}$ such that the union of all the contained rewritings of Q is a maximally-contained rewriting of Q using $\mathcal{V}$ in the presence of $\mathcal{F}$.

3 The MiniCon Algorithm

The MiniCon algorithm [5] is one of the algorithms for answering queries using views developed in the context of data integration. It generates all the contained rewritings of a given query, Q, whose union forms a maximally-contained rewriting of Q. In order to

do this, it first finds every view that covers a minimal set of subgoals in Q as required and then combines every set of selected views that covers pair-wise disjoint subsets of subgoals in Q to generate a conjunctive rewriting that is contained in Q. Given a mapping τ from $Vars(Q)$ to $Vars(V)$, where $Vars(Q)$ and $Vars(V)$ denote the sets of variables in a query Q and a view V respectively, a view subgoal g' is said to cover a query subgoal g if $\tau(g) = g'$.

To find a view that covers a minimal set of subgoals in Q as required, the MiniCon algorithm first finds a view V containing a subgoal g' that a subgoal g in Q can be mapped to by a partial mapping from g to g'. A partial mapping from g to g' can be found by finding a unifier θ from g to g', i.e., θ is a variable mapping from g to g' such that $\theta(g) = \theta(g')$. In the meanwhile, the unifier θ also needs to meet the requirement that the distinguished query variables in the query subgoal g are mapped to the distinguished view variables in the view subgoal g'. Once it finds the partial mapping, it then considers the joins between the view V and some of the other subgoals in Q and finds out whether any of the other subgoals in Q need to be mapped to subgoals in V, given that g will be mapped to g'. If so the minimal set of such subgoals is obtained. The requirement for including any of the other query subgoals in the minimal set of subgoals in Q that need to be mapped to subgoals in V is that if any existential query variable in the query subgoal g is part of a join predicate between g and the other query subgoal, and it has not been mapped to a distinguished view variable.

The minimal set of subgoals in Q and the corresponding mapping information are contained in a so called *MiniCon Description* (MCD). If it turns out that a view V does not cover the minimal set of subgoals in Q as required, no MCD will be generated for Q over V. The MCD for Q over V ensures that V covers the minimal set of subgoals in Q that need to be mapped to subgoals in V so that V can be used in a non-redundant rewriting of the corresponding subgoals in Q. Therefore, the MiniCon algorithm deals with combinations of relevant views, each covering a set of subgoals in Q, as candidate rewritings. In the second phase, the MCDs that cover pair-wise disjoint sets of subgoals in Q are combined to generate the rewritings.

The MiniCon algorithm, in particular, considers a mapping from a query to a specialization of a view if no mapping from the query to the view itself exists, where some of the distinguished variables in the view may have been equated. Every MCD has an associated *head homomorphism*. A head homomorphism h on a view V is a mapping from $Vars(V)$ to $Vars(V)$ that is identity on the existential variables, but may equate distinguished variables. A head homomorphism on a view maps it to one of its specialisations.

Definition 6 (MiniCon Descriptions). *A MCD C for a query Q over a view V is a tuple of the form*

$$(h_C, V(Y)_C, \varphi_C, G_C)$$

where:

- *h_C is a head homomorphism on V,*
- *$V(Y)_C$ is the result of applying h_C to the head of V, i.e., $Y = h_C(A)$, where A are the head variables of V,*

- *φ_C is a partial mapping from $Vars(Q)$ to $h_C(Vars(V))$,*
- *G_C is a subset of the subgoals in Q which are covered by some subgoals in $V(Y)_C$ using the mapping φ_C.*

In the above definition, φ_C is a mapping from a set of variables in Q to a set of specialized variables in $h_C(V)$ obtained by applying the head homomorphism h_C to the original set of variables in V. G_C is the minimal set of subgoals of Q that are covered by $h_C(V)$ as required, given φ_C.

Property 1 below specifies the exact conditions that need to be satisfied when deciding whether an MCD can be used in a non-redundant rewriting of the query and which query subgoals should be included in G_C. The MiniCon algorithm considers only those MCDs in which h_C is the least restrictive head homomorphism necessary in order to unify the minimal set of subgoals in the query with subgoals in a view.

Property 1. Let C be an MCD for a query Q over a view V. Then C can only be used in a non-redundant rewriting of Q if the following conditions hold:

C1: For each distinguished variable X of Q which is in the domain of φ_C, $\varphi_C(X)$ is a distinguished variable in $h_C(V)$.

C2: If $\varphi_C(X)$ is an existential variable in $h_C(V)$, then for every g, subgoal of Q, that includes X (1) all the variables in g are in the domain of φ_C, and (2) $\varphi_C(g) \in h_C(V)$.

Clause C1 makes sure that every distinguished query variable in the query is substituted by a distinguished view variable in a view that is used in a rewriting of the query. Clause C2 guarantees that if a query variable X is part of a join predicate in the query, which is not enforced by the view, then $\varphi_C(X)$ must be a distinguished view variable so the join predicate can be applied in the rewriting.

Property 2 below states the conditions that need to be satisfied when the MiniCon algorithm combines MCDs to generate non-redundant rewritings of a query so that only the MCDs that cover pair-wise disjoint subsets of subgoals of the query are combined.

Property 2. Given a query q, a set of views $\mathcal{V}$, and the set of MCDs $\mathcal{C}$ for q over $\mathcal{V}$, the only combinations of MCDs that can result in non-redundant rewriting of q are of the form C_1,...,C_l, where

D1. $G_{C_1} \cup ... \cup G_{C_l} = Subgoals(q)$, and

D2. for every $i \neq j$, $G_{C_i} \cap G_{C_j} = \emptyset$.

Example 3. Suppose we have the same mediated schema as in Example 1 and 2, and the following set of data sources:

$v_2(S', P') : -student(S', P', Y')$.

$v_4(S', Y') : -student(S', P', Y')$.

$v_5(P', D') : -taught(P', D')$.

$v_6(S', D') : -student(S', P', Y'), taught(P', D')$.

Consider the following query: $q(S, D) : -student(S, P, Y), taught(P, D)$.

The MiniCon algorithm creates the following MCDs[1]:

$V(Y)_C$	φ_C	G_C
$v_2(S', P')$	$S \to S'$ $P \to P'$ $Y \to Y'$	1
$v_5(P', D')$	$P \to P'$ $D \to D'$	2
$v_6(S', D')$	$S \to S'$ $P \to P'$ $Y \to Y'$ $D \to D'$	1,2

Combining the above MCDs, the MiniCon algorithm generates the following two rewritings only:

$q_1(S', D') : -v_2(S', P'), v_5(P', D')$.

$q_3(S', D') : -v_6(S', D')$.

4 Extending the MiniCon Algorithm

The MiniCon algorithm does not take into account the presence of functional dependencies in the mediated schema. As we indicated in Section 1, it sometimes misses query rewritings in the presence of such dependencies. In this section, we describe how the MiniCon algorithm can be extended to take into account the presence of functional dependencies and solve the problem of missing query rewritings.

Continue with the examples given in Section 1. In Example 1, we have the query:
$q(S, P, Y) : -student(S, P, Y)$.

We also have the following three data sources:
$v_1(S', Y', D') : -student(S', P', Y'), taught(P', D')$.
$v_2(S', P') : -student(S', P', Y')$.
$v_3(P', C') : -program(P', C')$.

The MiniCon algorithm, however, cannot generate the following rewriting:
$q'(S', P', Y') : -v_1(S', Y', D'), v_2(S', P')$.

Though we can have a partial mapping so that the only subgoal in q can be covered by the *student* subgoal in v_1. It is easy to see that not all the distinguished variables in the query subgoal can be mapped to the distinguished variables in v_1. So Clause C1 of Property 1 is violated. No MCD for q over v_1 can be used in a non-redundant rewriting of q. However, we can construct a joint view $v_{1,2}$ of v_1 and v_2 that has all the distinguished variables in either v_1 or v_2 as its distinguished variables, and all the subgoals in either v_1 or v_2 as its subgoals. The joint view provides all the distinguished variables that the distinguished variables in the query subgoal can be mapped to. We can therefore have an MCD for q over $v_{1,2}$ covering the only subgoal in q, which satisfies Clause C1 of Property 1 and can be used to generate a non-redundant rewriting of q. Furthermore, when the functional dependencies $S' \to P'$ and $S' \to Y'$ hold in the mediated schema, the joint view $v_{1,2}$ is equivalent to the join of v_1 and v_2 because the

[1] These are simplified MCDs in which the head homomorphisms on the views are omitted, where each homomorphism simply maps a view variable to itself

join is a lossless-join decomposition of $v_{1,2}$. The join of v_1 and v_2 can then be used to rewrite $v_{1,2}$ in the rewriting to get q'.

In Example 2, we have the query:
$q(D) : -taught(P, D), program(P, C), C = cs401.$

The MiniCon algorithm again fails to generate the following rewriting:
$q'(D) : -v_1(S, D), v_2(S, P), v_3(P, C), C = cs401.$

First of all, we can have a partial mapping so that the $taught$ subgoal in q is covered by the $taught$ subgoal in v_1 and the only distinguished variable in the query subgoal can be mapped to a distinguished variable in v_1. So Clause C1 of Property 1 is satisfied. However, the existential variable P in the $taught$ subgoal in q is in the join predicate with the $program$ subgoal in q. But the join predicate has not been enforced in v_1, and the P variable in v_1, which the P variable in q is mapped to, is not a distinguished variable in v_1 either. So Clause C2 of Property 1 is violated. No MCD for q over v_1, can be used in a non-redundant rewriting of q.

Again we can use the joint view $v_{1,2}$ of v_1 and v_2, in which the P variable is a distinguished variable. We can therefore have an MCD for q over $v_{1,2}$ covering the $taught$ subgoal in q, which satisfies both Clause C1 and C2 of Property 1 and can be used in a non-redundant rewriting of q. It is easy to see that another MCD for q over v_3 covering the $program$ subgoal in q can also be used in a non-redundant rewriting of q. The rewriting q' of q can be generated by first combining both MCDs for q over $v_{1,2}$ and v_3 respectively. Furthermore, the joint view $v_{1,2}$ is equivalent to the join of v_1 and v_2 only when functional dependencies $S' \rightarrow P'$, $P' \rightarrow D'$ and $S' \rightarrow D'$ hold in the mediated schema. Note that the third functional dependency is a transitive functional dependency which can be derived from the first two functional dependencies. So the join of v_1 and v_2 can be used to rewrite $v_{1,2}$ in the generated rewriting to get q'.

In the above examples, what we have revealed is the following. Though we can have a partial mapping so that a subgoal in a query q can be covered by a subgoal in a view v_1, no MCD for q over v_1 can be used in a non-redundant rewriting of q because one of the clauses of Property 1 is violated. However, in the presence of functional dependencies in the mediated schema, it may be possible to create a joint view $v_{1,2}$ of v_1 and another view v_2, over which no MCD for q can be used in a non-redundant rewriting of q either, so that (1) the MCD for q over $v_{1,2}$ satisfies both clauses of Property 1 and therefore can be used in a non-redundant rewriting of q; (2) the joint view $v_{1,2}$ is equivalent to the join of v_1 and v_2, which can then be used to rewrite $v_{1,2}$.

In Section 4.1, we describe how to form an MCD for a query Q over a joint view, which can be used in a non-redundant rewriting of Q. In Section 4.2, we describe how to combine MCDs over either single or joint views to generate the conjunctive rewritings of a query. Our extension of the MiniCon algorithm retains the main properties of the MiniCon algorithm and its computational advantage over the other algorithms.

4.1 Forming MCDs over Joint Views

We first formally define the joint view.

Definition 7. *Given a set of views $v_1(\overline{X}_1)$, $v_2(\overline{X}_2)$, ..., and $v_n(\overline{X}_n)$, their joint view, $v_{1,2,...,n}(\overline{X})$, is formed by having all the distinguished variables in the given views as its distinguished variables and all the subgoals in the given views as its subgoals.*

When forming the joint view from the given views, we make sure that the subgoals in different views with the same predicate are unified to get a single subgoal. As a result, some variables in different views may be mapped to a representative variable in the joint view, where we choose a distinguished variable as the representative variable whenever possible. In Examples 1 and 2, we have the following two views:

$v_1(S', Y', D') : -student(S', P', Y'), taught(P', D')$.

$v_2(S', P') : -student(S', P', Y')$.

and we can form a joint view:

$v_{1,2}(S', Y', D', P') : -student(S', P', Y'), taught(P', D')$.

Note that two $student$ subgoals in v_1 and v_2 are unified to get a single $student$ subgoal in the joint view $v_{1,2}$, in which the variables S', Y', D' and P' are all distinguished variables. Proposition 1 below specifies the exact conditions that we need to consider when we decide whether the join of a set of single views is equivalent to the joint view of the corresponding views.

Proposition 1. *Let $v_{1,2,...,n}(\overline{X})$ be the joint view of views $v_1(\overline{X}_1)$, $v_2(\overline{X}_2)$, ..., and $v_n(\overline{X}_n)$. Given that there exists a set of variables $X_1, ..., X_m$, where $X_1, ..., X_m \in \overline{X}_1$, $X_1, ..., X_m \in \overline{X}_2$, ..., and $X_1, ..., X_m \in \overline{X}_n$, and for any other variable X', where $X' \in \overline{X}_i$ for $1 \leq i \leq n$, the functional dependency $X_1, ..., X_m \rightarrow X'$ holds in $v_i(\overline{X}_i)$, then $v_{1,2,...,n}(\overline{X})$ is equivalent to the join of $v_1(\overline{X}_1), v_2(\overline{X}_2)$, ..., and $v_n(\overline{X}_n)$, that is, $v_{1,2,...,n}(\overline{X}) \equiv v_1(\overline{X}_1), v_2(\overline{X}_2)$, ..., $v_n(\overline{X}_n)$.*

It is straightforward that in the presence of the corresponding functional dependencies, the join of $v_1(\overline{X}_1), v_2(\overline{X}_2)$, ..., and $v_n(\overline{X}_n)$ is a lossless-join decomposition of the joint view $v_{1,2,...,n}(\overline{X})$. So we have the equivalence $v_{1,2,...,n}(\overline{X}) \equiv v_1(\overline{X}_1), v_2(\overline{X}_2)$, ..., $v_n(\overline{X}_n)$.

In Examples 1 and 2, as functional dependencies $S' \rightarrow Y'$, $S' \rightarrow D'$, and $S' \rightarrow P'$ hold in the mediated schema, we have the following equivalence:

$v_{1,2}(S', Y', D', P') \equiv v_1(S', Y', D'), v_2(S', P')$.

Property 3 below specifies the exact conditions that we need to consider when we decide which views can be used to form a joint view over which an MCD for a query q can be used in a non-redundant rewriting of q.

Property 3. Let $\mathcal{F}$ be a set of functional dependencies in the mediated schema, q be a query, $v_1(\overline{X}_1)$ be a view containing a subgoal that a subgoal in q can be mapped to but no MCD for q over v_1 satisfies both Clause C1 and C2 of Property 1 hence can be used in a non-redundant rewriting of q; Let $v_2(\overline{X}_2)$, ..., and $v_n(\overline{X}_n)$ be some other views over each of which no MCD for q can be used in a non-redundant rewriting of q, and $v_{1,2,...,n}(\overline{X})$ be a joint view of $v_1(\overline{X}_1)$, $v_2(\overline{X}_2)$, ..., and $v_n(\overline{X}_n)$; Let $C_{1,2,...n}$ be an MCD for q over $v_{1,2,...,n}(\overline{X})$. $C_{1,2,...,n}$ can only be used in a non-redundant rewriting of q if the following conditions hold:

C1: For each distinguished variable X of q which is in the domain of $\varphi_{C_{1,2,...,n}}$, $\varphi_{C_{1,2,...,n}}(X)$ is a distinguished variable in $h_{C_{1,2,...,n}}(v_{1,2,...n})$.

C2: If $\varphi_{C_{1,2,...,n}}(X)$ is an existential variable in $h_{C_{1,2,...,n}}(v_{1,2,...,n})$, then for every g, subgoal of q, that includes X, the following conditions must be satisfied: (1) all the variables in g are in the domain of $\varphi_{C_{1,2,...,n}}$, and (2) $\varphi_{C_{1,2,...,n}}(g) \in h_{C_{1,2,...,n}}(v_{1,2,...,n})$.

C3: $v_{1,2,...,n}(\overline{X}) \equiv v_1(\overline{X}_1), v_2(\overline{X}_2), ..., v_n(\overline{X}_n)$ holds in the presence of functional dependencies $\mathcal{F}$.

Clause C1 guarantees that for each distinguished variable X of q, which is in the domain of $\varphi_{C_{1,2,...,n}}$, $\varphi_{C_{1,2,...,n}}(X)$ is a distinguished variable in $h_{C_{1,2,...,n}}(v_{1,2,...,n})$. Clause C2 guarantees that if a variable X of q is part of a join predicate which is not enforced by the joint view $v_{1,2,...,n}$, then X must be a distinguished variable of $v_{1,2,...,n}$ so the join predicate can be applied in the rewriting. C3 guarantees that $v_{1,2,...,n}$ is equivalent to the join of $v_1(\overline{X}_1), v_2(\overline{X}_2), ..., v_n(\overline{X}_n)$ that can then be used to rewrite $v_{1,2,...,n}$. The extended MiniCon algorithm enforces the conditions in Property 3 to generate only those MCDs that satisfy these conditions and all the MCDs generated are used to form conjunctive rewritings.

In Example 1, we have the query: $q(S, P, Y) : -student(S, P, Y)$. and two views v_1 and v_2:

$v_1(S', Y', D') : -student(S', P', Y'), taught(P', D')$.

$v_2(S', P') : -student(S', P', Y')$.

each of which has a subgoal that the $student$ subgoal in q can be mapped to. But neither of v_1 and v_2 actually satisfies Clause C1 of Property 1. However, we can have the following joint view:

$v_{1,2}(S', Y', D', P') : -student(S', P', Y'), taught(P', D')$.

and a mapping from q to $v_{1,2}$:

$\varphi_{C_{1,2}} = \{S \rightarrow S', P \rightarrow P', Y \rightarrow Y'\}$

It is easy to see that every distinguished variable in q has been mapped to a distinguished variable in $v_{1,2}$. So Clause C1 of Property 3 is satisfied. Clause C2 does not apply. We also have functional dependencies: $S' \rightarrow Y'$, $S' \rightarrow D'$, and $S' \rightarrow P'$ in the mediated schema. So $v_{1,2}$ is equivalent to the join of v_1 and v_2 and Clause C3 is also satisfied. Now an MCD for q over $v_{1,2}$ can be used in a non-redundant rewriting of q. Furthermore, the join of v_1 and v_2 can be used to rewrite $v_{1,2}$. Therefore, we can have the following rewriting:

$q'(S', P', Y') : -v_1(S', Y', D'), v_2(S', P')$.

In Example 2, we have the query: $q(D)$:- $taught(P, D)$, $program(P, C)$, $C = cs401$. and the view v_1 that covers the $taught$ subgoal in q but does not satisfy Clause C2 of Property 1: $v_1(S', Y', D')$:- $student(S', P', Y')$, $taught(P', D')$. Again we can have the joint view $v_{1,2}$ $v_{1,2}(S', Y', D', P')$:- $student(S', P', Y')$, $taught(P', D')$. and a mapping from q to $v_{1,2}$: $\varphi_{C_{1,2}} = \{P \rightarrow P', D \rightarrow D'\}$

Now the variable P' in $v_{1,2}$ is a distinguished view variable. So Clause C2 of Property 3 is satisfied. In the presence of functional dependencies $S' \rightarrow Y'$, $S' \rightarrow D'$ and $S' \rightarrow P'$ in the mediated schema, Clause C3 of Property 3 is also satisfied. It is easy to see that another view v_3 covers the $program$ subgoal in q and can also be used in a non-redundant rewriting of q. Therefore we have the following rewriting:

$q'(D')$:- $v_1(S', Y', D')$, $v_2(S', P')$, $v_3(P', C')$, $C' = cs401$.

Given a subgoal in the query, the extended MiniCon algorithm first finds every view containing a subgoal that the query subgoal can be mapped to and checks whether the view satisfies Property 1. If so an MCD for the query over the view is created, which can then be used in a non-redundant rewriting of the query. In this phrase, the extended MiniCon algorithm is the same as the MiniCon algorithm. Otherwise, if a view can be

found which contains a subgoal that the query subgoal can be mapped to but the view does not satisfy either Clause C1 or C2 of Property 1, the algorithm finds other views to form, together with the given view, a joint view that satisfies Property 3. Clause C1 and C2 of Property 3 are the same as Clause C1 and C2 of Property 1 while Clause C3 of Property 3 further ensures that the joint view is equivalent to the join of the corresponding views.

When finding other views, together with the given view, to form a joint view, attention is paid to those views that can help to satisfy either Clause C1 or C2 of Property 1 which the given view failed to satisfy. We then also make sure that the joint view that consists of the selected views and the given view satisfies Clause C3 of Property 3. A joint view formed this way can therefore satisfy Property 3 and an MCD for the query over the joint view can be created, which can then be used in a non-redundant rewriting of the query. The joint view is added to the set of existing views.

4.2 Combining MCDs over Either Single or Joint Views

In the secodn phrase, the extended MiniCon algorithm finds valid combinations of MCDs formed in the first phrase and creates conjunctive rewritings of the query. The maximally-contained rewriting of the query is a union of conjunctive rewritings.

Property 4 specifies the exact conditions a combination of MCDs must satisfy so that it can be used to create a conjunctive rewriting of the query. The extended MiniCon algorithm enforces the conditions in Property 4 to combine only those MCDs that satisfy these conditions.

Property 4. Given a query q, a set of views $\mathcal{V}$, a set of functional dependencies $\mathcal{F}$ in the mediated schema, and the set of MCDs $\mathcal{C}$ formed by the first phase of the extended algorithm for q over $\mathcal{V}'$ that may also contain joint views apart from the single views in $\mathcal{V}$ in the presence of $\mathcal{F}$, the only combinations of MCDs that can result in non-redundant rewriting of q are of the form C_1,...,C_l, where

D1. $G_{C_1} \cup ... \cup G_{C_l} = Subgoals(q)$, and

D2. for every $i \neq j$, $G_{C_i} \cap G_{C_j} = \emptyset$.

For creating the rewriting q', the extended MiniCon algorithm works the exactly the same as the second phase of the MiniCon algorithm, simply treating joint views as single views. In the last step of the extended MiniCon algorithm, it however needs to replace every joint view in q' with its correct rewriting.

Theorem 1 states the properties of the extended MiniCon algorithm.

Theorem 1. *Given a conjunctive query q and a set of conjunctive views $\mathcal{V}$, in the presence of functional dependencies $\mathcal{F}$ in the mediated schema, the extended MiniCon algorithm is sound in the sense that every conjunctive rewriting q' that is generated by the algorithm is contained in q. In terms of completeness, the algorithm can generate the union of conjunctive rewritings that is a maximally-contained rewriting of q using $\mathcal{V}$ in the presence of $\mathcal{F}$ only if there exists such a maximally-contained rewriting. Sometimes, such a maximally-contained rewriting may not exist and recursive rewritings may be necessary in order to obtain a maximally-contained rewriting.*

The proofs of Properties 3 and 4 follow the correctness proof of the extended MiniCon algorithm. In terms of computational complexity of the extended MiniCon algorithm, creating extra joint views does not involve a significant increase of computational complexity compared to the MiniCon algorithm. Even in the worst case, in the first phase, the running time of the extended MiniCon algorithm is only roughly a number of times of the running time of the MiniCon algorithm. In the second phase, the running time of the extended MiniCon algorithm is virtually the same as that of the MiniCon algorithm. The correctness proof and complexity analysis of the extended MiniCon algorithm are omitted in this paper due to space limitation. For details, refer to the extended version of the paper.

5 Related Work

The problem of answering queries using views has relevance to a wide variety of data management problems [6]. In the context of data integration, a number of algorithms, such as the bucket algorithm [1], the inverse-rules algorithm [7, 8] and the MiniCon algorithm [5], have been developed for the problem of reformulating conjunctive queries using conjunctive views. However, these algorithms have not taken into account the presence of functional dependencies in the mediated schema. As a consequence, these algorithms may miss query rewritings in the presence of these integrity constraints.

Some algorithms have recently been developed for answering queries using views in the presence of functional dependencies [9–11]. These algorithms can in general be viewed as the extensions of the inverse-rules algorithm, and they inherit the performance costs of the inverse-rules algorithm. In [5], it has been proven that the inverse-rules algorithm does not scale up and is significantly outperformed by the scalable MiniCon algorithm. In this paper, we have presented an extension of the MiniCon algorithm to handle the presence of functional dependencies while retaining the main properties of the MiniCon algorithm and its significantly lower performance costs.

In addition to these algorithms, algorithms have been developed for conjunctive queries with comparison predicates [12, 13], recursive queries [10], queries over disjunctive views [14], queries over conjunctive views with negation [15], queries and views with grouping and aggregation [16, 17], queries over semi-structured data [18, 19], and OQL queries [20]. Duschka et al. [10] showed that in the presence of functional and full dependencies there does not always exist a non-recursive maximally-contained query rewriting. An algorithm [10] has been developed that deals with limitations on data sources, which are described by a set of allowed binding patterns. In this case it is known that recursive query rewritings may be necessary [3]. The algorithm constructs a recursive maximally-contained query rewriting.

6 Conclusions

In this paper, we have considered the problem of answering queries using views in the presence of functional dependencies. We have presented an extension of the MiniCon algorithm to deal with the functional dependencies in the mediated schema. The underlying idea is that in the presence of functional dependencies, some views can be joined

with other views to form joint views for which the corresponding MCDs can be used in the non-redundant rewritings of the query, thus avoiding the problem of missing queries in the presence of functional dependencies that the previous algorithms may have. Our extension of the MiniCon algorithm retains the main properties of the algorithm. The extension does not involve any significant increase in performance costs and retains the computational competitiveness of the MiniCon algorithm over the other algorithms.

In future work, we will further explore the possibilities of extending the MiniCon algorithm to deal with other types of integrity constraints in the mediated schema, such as inclusion dependencies and domain dependencies.

References

1. Levy, A.Y., Rajaraman, A., Ordille, J.J.: Querying heterogeneous information sources using source descriptions. In: VLDB. (1996) 251–262
2. Duschka, O.M., Genesereth, M.R.: Query planning in informaster. In: Proc. of the ACM Symposium on Applied Computing. (1997) 109–111
3. Kwok, C.T., Weld, D.S.: Planning to gather information. In: AAAI. (1996) 32–39
4. Lambrecht, E., Kambhampati, S., Gnanaprakasam, S.: Optimizing recursive information gathering plans. In: IJCAI. (1999) 1204–1211
5. Pottinger, R., Halevy, A.: Minicon: A scalable algorithm for answering queries using views. VLDB Journal **10** (2001) 182–198
6. Halevy, A.: Answering queries using views: a survey. VLDB Journal **10** (2001) 270–294
7. Qian, X.: Query folding. In: ICDE. (1996) 48–55
8. Duschka, O.M., Genesereth, M.R.: Answering recursive queries using views. In: PODS. (1997) 109–116
9. Duschka, O.M., Levy, A.Y.: Recursive plans for information gathering. In: IJCAI. (1997) 778–784
10. Duschka, O.M., Genesereth, M.R., Levy, A.Y.: Recursive query plans for data integration. Journal of Logic Programming **43** (2000) 49–73
11. Gryz, J.: Query rewriting using views in the presence of functional and inclusion dependencies. Information Systems **24** (1999) 597–612
12. Yang, H.Z., Larson, P.A.: Query transformation for psj-queries. In: VLDB. (1987) 245–254
13. Afrati, F.N., Li, C., Mitra, P.: Answering queries using views with arithmetic comparisons. In: PODS. (2002) 209–220
14. Duschka, O.M., Genesereth, M.R.: Query planning with disjunctive sources. In: Proc. of AAAI Workshop on AI and Information Integration. (1998)
15. Flesca, S., Greco, S.: Rewriting queries using views. IEEE Transactions on Knowledge and Data Engineering **13** (2001) 980–995
16. Gupta, A., Harinarayan, V., Quass, D.: Aggregate-query processing in data warehousing environments. In: VLDB. (1995) 358–369
17. Srivastava, D., Dar, S., Jagadish, H.V., Levy, A.Y.: Answering sql queries using materilized views. In: VLDB. (1996) 318–329
18. Papakonstantinou, Y., Vassalos, V.: Query rewriting for semi-structured data. In: SIGMOD. (1999) 455–466
19. Calvanese, D., Giacomo, G.D., Lenzerini, M., Vardi, M.: Rewriting of regular expressions and regular path queries. In: PODS. (1999) 194–204
20. Florescu, D., Raschid, L., Valduriez, P.: A methodology for query reformulation in cis using semantic knowledge. International Journal of Intelligent and Cooperative Information Systems **5** (1996) 431–468