Information Systems 35 (2010) 149–169

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Computing certain answers in the presence of dependencies ☆

Foto N. Afrati, Nikos Kiourtis *,[1]

Electrical and Computing Engineering, National Technical University of Athens, 157 73 Athens, Greece

ARTICLE INFO

Keywords:
Conjunctive queries
Views
Certain answers
Query rewritings
Dependencies

ABSTRACT

In this paper we consider conjunctive queries and views, and we investigate the problem of query answering using views in the presence of dependencies and in particular the problem of finding equivalent and maximally contained rewritings of a query using a set of views in the presence of dependencies. We present an efficient sound and complete algorithm CoreCover$\mathcal{C}$ which finds equivalent rewritings with the minimum number of subgoals in the presence of weakly acyclic local as view tuple generating dependencies ($\mathcal{C}^{w}_{\mathsf{LAV}}$). We also present an efficient algorithm Minicon$\mathcal{C}$ that finds maximally contained rewritings (MCRs) with respect to the language of finite unions of conjunctive queries (UCQ) of a UCQ query Q in the presence of $\mathcal{C}^{w}_{\mathsf{LAV}}$. We also prove that an MCR of a UCQ query Q with respect to UCQ computes all the certain answers of Q both in the absence and presence of a set of dependencies $\mathcal{C}$ if the chase of Q with $\mathcal{C}$ terminates (and, in the case of dependencies, if such an MCR exists).

1. Introduction

In many data-management applications, such as information integration, data warehousing, web-site designs, and query optimization, the problem of answering queries using views is of special significance. The presence of dependencies in query answering is also very important, because dependencies are essential for dealing with true concept mismatch between the data of the sources. In this paper, we investigate the problem of finding equivalent and maximally contained rewritings of a query Q using a set of views $\mathcal{V}$ under a set of dependencies $\mathcal{C}$. We also show under what conditions a maximally contained rewriting can be used to answer a query using views with the certain answers semantics.

The problem of finding equivalent rewritings is formally defined as follows: we have a database schema R, a set of CQ views $\mathcal{V}$ over schema R, a set of tgds and egds $\mathcal{C}$ over schema R and a CQ query Q over schema R. The solution to the problem is a conjunctive query Q' over the schema of views $\mathcal{V}$ such that $Q \equiv_{\mathcal{C}} Q'$, i.e. for all databases D that satisfy the constraints $\mathcal{C}$: $Q(D) = Q'(\mathcal{V}(D))$. Such a conjunctive query may not always exist, even in the absence of dependencies and even if certain strong conditions hold (see e.g. [5,24]). This problem is harder than the problem of finding equivalent rewritings when no dependencies are present, because the presence of some dependencies may assert the equivalence of some rewritings that are otherwise not equivalent if these dependencies are missing. This is illustrated in the following example:

Example 1. Suppose that a company has a database with the relations $\mathsf{Dept}_A(Name)$, $\mathsf{Dept}_B(Name)$ and $\mathsf{Staff}(Name, Tel, BTel)$, where Dept_A and Dept_B store the employees that work in Department A and Department B, respectively, and Staff stores the employees along with their home and business telephone numbers. Suppose we have the query

☆ A short version [3] of this paper appeared in the International Workshop on New Trends in Information Integration (NTII) 2008, Auckland, New Zealand.

* Corresponding author.
E-mail address: nkiourtis@gmail.com (N. Kiourtis).

[1] This paper is part of the 03ED176 research project, implemented within the framework of the "Reinforcement Programme of Human Research Manpower" (PENED) and co-financed by National and Community Funds (20% from the Greek Ministry of Development-General Secretariat of Research and Technology and 80% from E.U.-European Social Fund).

0306-4379/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2009.08.002

$Q(N,T) : -\mathsf{Dept_A}(N),\ \mathsf{Dept_B}(N),\ \mathsf{Staff}(N,T,T)$, that finds all employees with their home phone numbers that work in both departments and have listed their business number as their home number, i.e. those that work at home. Suppose also that we have a set $\mathcal{V}$ of two views: in view V_1 we have stored all employees with their home number that work in Department A, i.e. $V_1(N,T) : -\mathsf{Dept_A}(N),\ \mathsf{Staff}(N,T,B)$. In the view V_2 we have stored all employees with their business number that work in department B, i.e. $V_2(N,B) : -\mathsf{Dept_B}(N),\ \mathsf{Staff}(N,T,B)$. Thus, in this case, there is no equivalent rewriting of Q using $\mathcal{V}$. If, however, each employee has only one home and business number, which is captured by the functional dependencies $Name \rightarrow Tel$ and $Name \rightarrow BTel$, then the following rewriting $R(N,T) : -V_1(N,T),\ V_2(N,T)$ is an equivalent rewriting of Q. To illustrate this point, consider a database instance D that satisfies the functional dependencies. If $\mathsf{Staff}(n,t,b) \in D$ and $\mathsf{Staff}(n,t',b') \in D$, then it must hold that $t = t'$ and $b = b'$. Thus, if $(n,t) \in R(\mathcal{V}(D))$, then it must hold that $\mathsf{Dept_A}(n), \mathsf{Dept_B}(n), \mathsf{Staff}(n,t,t) \in D$, which yields $(n,t) \in Q(D)$ (the other direction is symmetrical).

The problem of finding maximally contained rewritings of a query using a set of views is another recognized open problem (see [4]), and is formally defined as follows: we have a database schema R, a set of CQ views $\mathcal{V}$ over schema R, a set of tgds and egds $\mathcal{C}$ over schema R and a UCQ query Q over schema R. The solution to the problem is a UCQ query $\mathcal{P}$ over the schema of views $\mathcal{V}$ such that: (i) $\mathcal{P}$ is a contained rewriting of Q under $\mathcal{C}$, (ii) if $\mathcal{P}'$ is another contained rewriting of Q under $\mathcal{C}$, then $\mathcal{P}' \sqsubseteq \mathcal{P}$.

In the presence of dependencies it may be the case that there exist only datalog maximally contained rewritings of a query, as in the following example:

Example 2. Suppose we have a database schema that contains one relation $schedule(Airline, Flight\_no, Date, Pilot, Aircraft)$. The set of views $\mathcal{V}$ contains one view $V(D,P,C) : -schedule(A,N,D,P,C)$, and we have a set $\mathcal{C} = \{d_1, d_2\}$ of two functional dependencies, where $d_1 : Pilot \rightarrow Airline$ and $d_2 : Aircraft \rightarrow Airline$. If we write the previous functional dependencies as egds we have that: $d_1 : schedule(A,N,D,P,C) \wedge schedule(A',N',D',P,C') \rightarrow A = A'$ and $d_2 : schedule(A,N,D,P,C) \wedge schedule(A',N',D',P',C) \rightarrow A = A'$. We consider the following query:

$$Q(P) : -schedule(A,N,D,mike,C), schedule(A,N',D',P,C')$$

For each $n \in \mathbb{N}$, the following query Q_n is a contained CQ rewriting of Q:

$$\begin{aligned} Q_n(P) : -\ & V(D_1,mike,C_1), V(D_2,P_2,C_1), \\ & V(D_3,P_2,C_2), V(D_4,P_3,C_2), \\ & V(D_5,P_3,C_3), V(D_6,P_4,C_3), \ldots, \\ & V(D_{2n-2},P_n,C_{n-1}), \\ & V(D_{2n-1},P_n,C_n), V(D_{2n},P,C_n) \end{aligned}$$

However, Q_n is not an equivalent rewriting for all $n \in \mathbb{N}$. To see this, consider the following database instance D: $D = \{schedule(a_1,n_1,d_1,mike,c_1), schedule(a_2,n_2,d_2,\ p_2,\ c_1),\ schedule(a_2,n_2,d_3,p_2,c_2), schedule(a_3,n_3,\ d_4,p_3,c_2), \ldots, schedule(a_{2n-2},n_{2n-2},d_{2n-2},p_n,\ c_{n-1}),\ schedule(a_{2n-1},n_{2n-1}, d_{2n-1},p_n,c_n),\ schedule(a_{2n},n_{2n},d_{2n},p,c_n),\ schedule(a_{2n+1}, n_{2n+1},d_{2n+1},p,\ c_{n+1}), schedule(a_{2n+2},n_{2n+2},\ d_{2n+2},p_{n+1},c_{n+1})\}$

In order for D to satisfy the dependencies $\mathcal{C}$, it must hold that $a_1 = a_2 = \cdots = a_{2n+2}$, which means that all the pilots $\{mike, p_2, \ldots, p, p_{n+1}\}$ work for the same airline, and thus $Q(D) = \{mike, p_2, \ldots, p_{n+1}, p\}$. We also have that $V(D) = \{v(d_1,mike,c_1), v(d_2,p_2,c_1), \ldots, v(d_{2n+2},\ p_{n+1},c_{n+1})\}$.

Since the number of subgoals of Q_n is n and the first subgoal of Q_n must always be assigned to the fact $schedule(a_1,n_1,d_1,mike,c_1)$, we have that $p_{n+1} \notin Q_n(D)$ (and to be precise, $Q_n(D) = \{mike, p_2, \ldots, p_n, p\}$). Thus, $Q_n(D) \subsetneq Q(D)$. In this example, there is a Datalog MCR of Q, which is the following:

$$\begin{aligned} & relevantPilot(mike) \\ & relevantAirCraft(C) : -V(D,mike,C) \\ & relevantAirCraft(C) : -V(D,P,C),\quad relevantPilot(P) \\ & relevantPilot(P) : -relevantPilot(P_1), \\ & relevantAirCraft(C), v(D_1,P_1,C),\quad v(D_2,P,C) \end{aligned}$$

In Section 5.2, we will show that there is no maximally contained rewriting of Q with respect to the language of unions of conjunctive queries.

Our contributions are the following:

- We propose an efficient algorithm that finds equivalent rewritings with the minimum number of subgoals in the presence of a class of dependencies known as weakly acyclic LAV tgds (Section 4). Then, we extend this algorithm to apply on a larger class of dependencies. We do that by identifying an important property used for optimization in a number of query rewriting algorithms in the literature.
- We propose an efficient algorithm that finds maximally contained rewritings of a UCQ query Q with respect to UCQ in the presence of weakly acyclic LAV tgds (Section 5.1).
- We show that in the absence of dependencies, a maximally contained rewriting of a UCQ query Q with respect to UCQ always exists and computes the certain answers of the query over a view instance. In the presence of a set of dependencies $\mathcal{C}$ such that the chase of Q with $\mathcal{C}$ always terminates, we prove that a maximally contained rewriting of a UCQ query Q with respect to UCQ computes the certain answers of the query over a view instance, if such and MCR exists (Section 5.2).

2. Related work

The problem of finding equivalent rewritings in the presence of dependencies was studied in [25,15], where an algorithm called Chase & BackChase is presented. This algorithm works in two phases: in the first phase (chase phase) the original query Q is chased using all the applicable dependencies (until no more chase steps are possible) into a new query U that is called a universal plan. The universal plan essentially incorporates all possible alternative ways to answer Q in the presence of the

dependencies, i.e. all rewritings of the Q will be subqueries of the universal plan. In the second phase (backchase step) all the available subqueries of U are checked for equivalence with Q by chasing them with the dependencies, and those that are found to be equivalent and have the minimum number of subgoals are returned as the answer. This step is called "backchase", because the chase sequences of the subqueries of U go "backwards" toward the query U. This algorithm considers all combinations of views for candidate rewritings. In [7,8], the number of such combinations is significantly reduced by sophisticated techniques and in particular the notion of the tuple-core is introduced. In this paper, we use these techniques to obtain an efficient algorithm for finding equivalent rewritings in the presence of weakly acyclic local as view tuple generating dependencies.

In [18], the notion of tuple-core has been extended to tuple-coverage, where the authors search for a minimal cover (instead of maximal) of the query's subgoals when using view tuples to find equivalent rewritings of a conjunctive query using CQ views. In addition, in this paper tuple-coverage is also extended for finding equivalent rewritings for conjunctive queries with aggregates using views that are also conjunctive queries with aggregates.

Algorithms that find maximally contained rewritings for various cases of queries and views which are conjunctive queries with or without arithmetic comparisons, are presented in [26,23,6]. The problem of finding maximally contained rewritings in the presence of functional dependencies or full dependencies is studied in [16], and the inverse rules algorithm is modified to produce rewritings. The algorithm starts by creating the datalog program as in the case where no dependencies are present, and then some additional rules are added which simulate chasing the query with a dependency. In the paper it is noted that this extension to the inverse rules algorithm will not work with inclusion dependencies that are not full, because the introduced existential variables may create new Skolem terms in a recursive way, and so semi-naive evaluation on the resulting datalog program may not terminate.

In [21], an algorithm is presented that finds maximally contained rewritings under a special class of dependencies called conjunctive inclusion dependencies, which are essentially GLAV mappings that are viewed as dependencies. This algorithm is sound but not complete and can be seen as an extension of the inverse rules algorithm in [16].

In [12], the authors deal with the problem of creating maximally contained rewritings in global-as-view data integration systems in the presence of inclusion dependencies and key constraints. They first show that the problem is undecidable in the general case and in the case where inclusion dependencies are present together with key constraints. Then, they present a sound and complete algorithm for the case of inclusion dependencies, and the case of key constraints together with a subclass of inclusion dependencies which they call "non-key conflicting inclusion dependencies". The main idea of the algorithm was then extended in [11] to GLAV data integration systems in the presence of tuple generating dependencies.

Additional work on related problems includes the following: In [9], the problem of finding contained rewritings in the presence of inclusion dependencies was studied, where the MiniCon algorithm is modified. The resulting algorithm first chases the query and the views with the dependencies and then applies MiniCon properly modified to utilize the new view and query subgoals. Another algorithm that finds equivalent rewritings under a set of inclusion dependencies is found in [19]. This algorithm first undoes all the chase steps that may have been applied in the query, and then for each atom in the query it tries to find equivalent replacement atoms by using the set of inclusion dependencies. An extension to this algorithm is also presented that finds all contained rewritings.

3. Preliminaries

A conjunctive query or CQ [2,28] over a schema R is an expression of the form $Q(\vec{x}) : -\phi(\vec{x},\vec{y})$, where $\phi(\vec{x},\vec{y})$ is a conjunction of atomic formulas that are also called subgoals of the query. The dependencies we will deal within this paper are called tuple generating dependencies and equality generating dependencies and are defined below:

Definition 1. A tuple generating dependency (tgd) is a logic formula of the form $\forall\vec{x}(\phi(\vec{x})\rightarrow\exists\vec{y}\psi(\vec{x},\vec{y}))$, and an equality generating dependency (egd) is a logic formula of the form $\forall\vec{x}(\phi(\vec{x})\rightarrow x_1 = x_2)$. In both tgds and egds, $\phi(\vec{x})$ and $\psi(\vec{x},\vec{y})$ are conjunctions of atomic formulas over R, all of the variables in $\vec{x}$ must appear in $\phi(\vec{x})$ and x_1, x_2 must appear in $\vec{x}$.

We now give the definition of a weakly acyclic set of tgds that will be used in the paper:

Definition 2 (*Fagin [17]*). Let $\mathcal{C}$ be a set of tgds over a fixed schema. Construct a directed graph, called the dependency graph, as follows:

1. There is a node for every pair (R,A) with R a relation symbol of the schema and A an attribute of R and call such pair (R,A) a position.
2. Add edges as follows: for every tgd $\phi(\vec{x})\rightarrow\exists\vec{y}\psi(\vec{x},\vec{y})$ in $\mathcal{C}$ and for every x in $\vec{x}$ that occurs in $\vec{y}$:
   - For every occurrence of x in ϕ in position (R,A_i):
     (a) for every occurrence of x in ψ in position (S,B_j), add an edge $(R,A_i)\rightarrow(S,B_j)$ (if it does not already exist);
     (b) in addition, for every existentially quantified variable y and for every occurrence of y in ψ in position (T,C_k), add a special edge $(R,A_i)\rightarrow(T,C_k)$ (if it does not already exist).

Note that there may be two edges in the same direction between two nodes, if exactly one of the two edges is special. Then $\mathcal{C}$ is weakly acyclic if the dependency graph has no cycle going through a special edge.

We also give the definition of full dependencies that are used in the inverse rules algorithm [16]:

Definition 3. A full dependency is a first order formula of the form $\forall\vec{x}(\phi(\vec{x})\rightarrow\psi(\vec{y}))$, where $\phi(\vec{x})$ is a conjunction of relations and equality atoms with variables $\vec{x}$, $\psi(\vec{y})$ is either a single relation of an equality atom with variables $\vec{y}$, and $\vec{y}\subseteq\vec{x}$, i.e. all variables that appear in $\vec{y}$ must also appear in $\vec{x}$.

Let $\mathcal{C}$ be a set of constraints (tdgs and egds), and Q_1, Q_2 be two conjunctive queries. We say that Q_1 is contained in Q_2 under the constraints $\mathcal{C}$ and we write $Q_1\sqsubseteq_{\mathcal{C}}Q_2$, if for all databases D that satisfy $\mathcal{C}$ we have that $Q_1(D)\subseteq Q_2(D)$. If $\mathcal{V}$ is a set of CQ views, an equivalent CQ rewriting of Q using $\mathcal{V}$ in the presence of C under the closed world assumption (CWA) is a CQ query R that has the same head variables with Q, its body consists only of views from $\mathcal{V}$ and for all databases D that satisfy $\mathcal{C}$: $R(\mathcal{V}(D))=Q(D)$.

Suppose that Q is a query in the language of unions of conjunctive queries (UCQ) and $\mathcal{V}$ is a set of CQ views. A UCQ query R that uses only views from $\mathcal{V}$ is a contained rewriting of Q using $\mathcal{V}$ if for all view instances $\mathcal{I}$ and for all databases D such that $\mathcal{I}\subseteq\mathcal{V}(D)$ we have that $P(\mathcal{I})\subseteq Q(D)$.

If $\mathcal{L}$ is a query language, a maximally contained rewriting (MCR) $\mathcal{P}$ of Q with respect to $\mathcal{L}$ using $\mathcal{V}$ in the presence of C under the open world assumption (OWA) is an $\mathcal{L}$-query that uses only views from $\mathcal{V}$ and for all databases D and for any view instance $\mathcal{I}$ such that $\mathcal{I}\subseteq\mathcal{V}(D)$ and D satisfies the constraints $\mathcal{C}$ we have that $\mathcal{P}$ is a contained rewriting of Q, and if there is another $\mathcal{L}$-query $\mathcal{P}'$ such that $\mathcal{P}'$ is a contained rewriting of Q, then $\mathcal{P}'\sqsubseteq\mathcal{P}$.

Let R be a CQ rewriting of a CQ query Q using CQ views from $\mathcal{V}$. The expansion of R is denoted by R^{exp} and is obtained from R by replacing all the views in the body of R by their corresponding base relations in their view definitions, such that the existentially quantified variables in each view are replaced by fresh variables in R^{exp}.

We define the certain answers of $(Q,\mathcal{I})$ with respect to $\mathcal{V}$ as follows: under CWA, $\text{certain}(Q,\mathcal{I})=\bigcap\{Q(D): D \text{ such that } \mathcal{I}=\mathcal{V}(D)\}$, and under OWA, $\text{certain}(Q,\mathcal{I})=\bigcap\{Q(D): D \text{ such that } \mathcal{I}\subseteq\mathcal{V}(D)\}$ (in the presence of constraints $\mathcal{C}$, we also require that all databases D used for certain $(Q,\mathcal{I})$ satisfy $\mathcal{C}$).

We next give the definition of evaluating a query on a database instance.

Definition 4. Let D be an instance and Q a CQ. We say that $t\in Q(D)$ (i.e. tuple t belongs to the answers of Q when applied to D) when there exists a homomorphism $h: Var(Q)\rightarrow Const(D)$ such that the following three hold:

1. h maps all the variables of Q to constants in D and all constants of Q to the same constants in D.
2. If $R_i(y_1,\ldots,y_n)\in Body(Q)$ then $R_i(h(y_1),\ldots,h(y_n))$ is a fact of D, i.e. $R_i(h(y_1),\ldots,h(y_n))\in D$.
3. If $Head(Q)=(x_1,\ldots,x_k)$, then $t=(h(x_1),\ldots,h(x_k))$.

The chase procedure is used in this paper and is defined below:

Definition 5 (*Chase step*). Let Q be a conjunctive query and B be the set of all predicate atoms in the body of Q.

- Suppose that d is a tgd $\phi(\vec{x})\rightarrow\exists\vec{y}\psi(\vec{x},\vec{y})$. Let h be a homomorphism from $\phi(\vec{x})$ to B such that there is no extension of h to a homomorphism h' from $\phi(\vec{x})\wedge\psi(\vec{x},\vec{y})$ to B. We say then that d can be applied to Q with homomorphism h, and we construct B' as follows: define B' to be the union of B with a set of facts obtained by taking the atoms of ψ and substituting all variables x with $h(x)$ and every existentially quantified variable y (not defined in h) with a fresh variable. Thus, if we denote by $\vec{z}$ the vector of fresh variables that we use for $\vec{y}$, we can write $B'=B\cup\{\psi(h(\vec{x}),\vec{z})\}$. We then form a conjunctive query Q' that has the same head variables as Q and the predicates in the body of Q' are exactly the elements of the setB'. We say that the result of applying d to Q with h is Q', and we write $B\xrightarrow{d,h}B'$ and $Q\xrightarrow{d,h}Q'$.
- Suppose that d be an egd $\phi(\vec{x})\rightarrow(x_1=x_2)$. Let h be a homomorphism from $\phi(\vec{x})$ to B such that $h(x_1)\neq h(x_2)$. We say that d can be applied to Q with homomorphism h, and we construct B' as follows: define B' to be the set that is produced from the elements of B in the following manner: take each subgoal of B, replace each occurrence of the variable $h(x_2)$ with $h(x_1)$, and add the resulting subgoal to the set B' (i.e. $B'=\theta(B)$, where θ is defined as the identity function on all variables except x_2, and $\theta(x_2)=x_1$). We then form a conjunctive query Q', where the head of Q' is defined as $Head(Q')=\theta(Head(Q))$, and the predicates in the body of Q' are exactly the elements of the set B'. We say that the result of applying d to Q with h is Q', and we write $B\xrightarrow{d,h}B'$ and $Q\xrightarrow{d,h}Q'$.

Definition 6 (*Chase*). Let $\mathcal{C}$ be a set of tgds and egds and let Q be a conjunctive query.

- A *chase sequence of Q with $\mathcal{C}$* is a finite or infinite sequence of chase steps $Q_i\overset{d_i,h_i}{\rightarrow}Q_{i+1}$, with $i=0,1,\ldots$, $Q_0=Q$ and d_i a dependency in $\mathcal{C}$.
- A *finite chase of Q with $\mathcal{C}$* is a finite chase sequence $Q_i\overset{d_i,h_i}{\rightarrow}Q_{i+1}$, $0\leq i\leq n$, such that there is no dependency $d_j\in\mathcal{C}$ and there is no homomorphism h_j such that d_j can be applied to Q_n. We say that Q_n is the result of the finite chase and we write $Q\overset{\mathcal{C}}{\rightarrow}Q_n$.

The following proposition that uses properties of the chase step first appeared in [10] and will be used in some of the proofs:

Proposition 1. *If the chase of a conjunctive query Q with a set of constraints $\mathcal{C}$ terminates and $Q_{\mathcal{C}}$ is the chased query, then for all databases D that satisfy the constraints $\mathcal{C}$, $Q(D)\subseteq Q_{\mathcal{C}}(D)$, i.e. $Q\sqsubseteq_{\mathcal{C}}Q_{\mathcal{C}}$.*

Proof. Let D be a database that satisfies the constraints $\mathcal{C}$. Suppose that the chase of Q with $\mathcal{C}$ terminates after n steps and that the chase sequence is $Q=Q_1,Q_2,\ldots,Q_n=Q_{\mathcal{C}}$, where $Q_i\xrightarrow{d_i,g_i}Q_{i+1}$ for $d_i\in\mathcal{C}$. We will show that if there exists a homomorphism $h_i: Var(Q_i)\rightarrow Const(D)$ that computes an answer t of query Q in database D, then there exists a

homomorphism $h_{i+1} : Var(Q_{i+1}) \to Const(D)$ that computes the same answer t of Q_{i+1} in D.

Let $t \in Q_i(D)$. By Definition 4 there exists a homomorphism $h_i^t : Var(Q_i) \to Const(D)$ such that:

1. h_i^t maps all the variables of Q_i to constants in D.
2. If $R_j(\vec{y_j}) \in Body(Q_i)$ then $R_i(h_i^t(\vec{y_j}))$ is a fact of D.
3. If $Head(Q_i) = (x_1, \ldots, x_k)$, then $t = (h_i^t(x_1), \ldots, h_i^t(x_k))$.

Suppose that $Q_i \xrightarrow{d,g_i} Q_{i+1}$ with $d \in \mathcal{C}$. We consider the following two cases:

1. d is a tgd $\phi(\vec{x}) \to \exists \vec{y} \psi(\vec{x}, \vec{y})$. We denote the set of subgoals in the body of Q_i by B_{Q_i}. By Definition 5 of the chase step, there is a homomorphism $g_i : \phi(\vec{x}) \to B_{Q_i}$. Thus, $h_i^t \circ g_i : \phi(\vec{x}) \to D$ is also a homomorphism, and because D satisfies d, $h_i^t \circ g_i$ can be extended to a homomorphism $h' : \phi(\vec{x}) \wedge \psi(\vec{x}, \vec{y}) \to D$, such that $h'(\vec{x}) = h_i^t(h(\vec{x}))$. For each variable y in $\vec{y}$,we denote by z_y the fresh variable that replaces y in the chase step $Q_i \xrightarrow{d,g_i} Q_{i+1}$.
   We define $h_{i+1}^t : Q_{i+1} \to D$, where $h_{i+1}^t(x) = h_i^t(x)$ if x is among the variables of B_{Q_i}, and $h_{i+1}^t(z_y) = h'(y)$ for y in $\vec{y}$. We will show that h_{i+1}^t is a homomorphism.
   (a) Let x be a variable of Q_{i+1}. If x is among the variables of Q_i, then x is among the variables of B_{Q_i} and $h_{i+1}^t(x) = h_i^t(x)$. But $h_i^t(x) \in Const(D)$, so $h_{i+1}^t(x) \in Const(D)$. If x does not appear in Q_i, then x appears in $\vec{z_y}$ and we have that $h_{i+1}^t(x) = h'(x')$ for an appropriate x'. From the definition of h', we have that $h'(x')$ is a constant in D, so $h_{i+1}^t(x)$ is also a constant in D. Thus, h_{i+1}^t maps all variables of Q_{i+1} to constants in D.
   (b) Let $R_j'(\vec{y_j}) \in Body(Q_{i+1})$. Then either $R_j'(\vec{y_j}) \in Body(Q_i)$ or $R_j'(\vec{y_j})$ is an atom in the conjunction ψ. If $R_j'(\vec{y_j}) \in Body(Q_i)$, then $\vec{y_j}$ are among the variables of Q_i, so $R_j'(h_{i+1}^t(\vec{y_j})) = R_j'(h_i^t(\vec{y_j}))$. But h_i^t is a homomorphism, so $R_j'(h_i^t(\vec{y_j}))$ is a fact of D, hence $R_j'(h_{i+1}^t(\vec{y_j}))$ is also a fact of D. If $R_j'(\vec{y_j})$ is an atom in the conjunction ψ, then it will have the form $T(\vec{x_0}, \vec{y_0})$, where $\vec{x_0}$ are among the variables of Q_i and $h_{i+1}^t(\vec{x_0}) = h_i^t(\vec{x_0})$, and $\vec{y_0}$ appears in $\vec{z_y}$, so $h_{i+1}^t(\vec{y_0}) = h'(\vec{y_1})$ for an appropriate $\vec{y_1}$. Using the previously defined homomorphism h, we can write $\vec{x_0} = h(\vec{x})$. We have that $T(h_{i+1}^t(\vec{x_0}), h_{i+1}^t(\vec{y_0})) = T(h_i^t(\vec{x_0}), h'(\vec{y_1})) = T(h_i^t(h(\vec{x})), h'(\vec{y_1}))$. But x appears in $\phi(x)$, so $h_i^t(h(\vec{x})) = h'(\vec{x})$ and we have that $T(h_i^t(h(\vec{x})), h'(\vec{y_1})) = T(h'(\vec{x}), h'(\vec{y_1}))$. Since $h' : \phi(\vec{x}) \wedge \psi(\vec{x}, \vec{y}) \to D$ is a homomorphism, $T(h'(\vec{x}), h'(\vec{y_1}))$ is a fact of D, hence $T(h_{i+1}^t(\vec{x_0}), h_{i+1}^t(\vec{y_0}))$ is also a fact of D. Thus, h_{i+1}^t makes each subgoal of Q_{i+1} a fact of D.

   By the construction of Q_{i+1}, Q_i and Q_{i+1} have the same head variables, so the following hold for the answer of $t' \in Q_{i+1}(D)$ computed by h_{i+1}^t: $t' = (h_{i+1}^t(x_1), \ldots, h_{i+1}^t(x_k)) = (h_i^t(x_1), \ldots, h_i^t(x_k)) = t$. Hence, $t \in Q_{i+1}(D)$.
2. d is an egd $\phi(\vec{x}) \to (x_1 = x_2)$. As in the previous case, by Definition 5 of the chase step, there is a homomorphism $g_i : \phi(\vec{x}) \to B_{Q_i}$ and we have that $h_i^t \circ g_i : \phi(\vec{x}) \to D$ is a homomorphism. Because the database D satisfies the egd d, we have that $h_i^t(x_1) = h_i^t(x_2)$. We define $h_{i+1}^t : Q_{i+1} \to D$ where $h_{i+1}^t(x) = h_i^t(x)$. Let $R_j(\vec{y_j}) \in Body(Q_{i+1})$ and $R_j(\vec{y_j'}) \in Body(Q_i)$ the corresponding subgoal of Q_i that produced it during the chase step. If x_2 does not appear in the variables of $R_j(\vec{y_j'})$, then $R_j(h_{i+1}^t \vec{y_j}) = R_j(h_i^t(\vec{y_j'}))$ and thus $R_j(h_{i+1}^t \vec{y_j})$ is a fact of D. If x_2 appears in the variables of $R_j(\vec{y_j'})$, then, since $h_i^t(x_1) = h_i^t(x_2)$, we have that $R_j(h_{i+1}^t \vec{y_j}) = R_j(h_i^t(\vec{y_j'}))$ and thus $R_j(h_{i+1}^t \vec{y_j})$ is a fact of D. Hence h_{i+1}^t is a homomorphism and $t \in Q_{i+1}(D)$.

Thus, $Q_i(D) \subseteq Q_{i+1}(D)$ for $i = 1, \ldots, n-1$, so for all databases D that satisfy the constraints $\mathcal{C}$: $Q(D) \subseteq Q_{\mathcal{C}}(D)$. □

Corollary 1. *If the chase of a conjunctive query Q with a set of constraints $\mathcal{C}$ terminates and $Q_{\mathcal{C}}$ is the chased query, then $Q \equiv_{\mathcal{C}} Q_{\mathcal{C}}$, i.e. for all databases D that satisfy $\mathcal{C}$: $Q(D) = Q_{\mathcal{C}}(D)$.*

Proof. From Proposition 1 we have that for all databases D that satisfy $\mathcal{C}$: $Q(D) \subseteq Q_{\mathcal{C}}(D)$, and also trivially $Q_{\mathcal{C}}(D) \subseteq Q(D)$. Hence, $Q \equiv_{\mathcal{C}} Q_{\mathcal{C}}$. □

In the rest of the paper we assume that UCQ is the language of finite union of conjunctive queries, $\mathcal{V}$ is a set of views in the language of CQs and $\mathcal{I}$ is a view instance for $\mathcal{V}$. Let Q be a CQ, $\mathcal{V}$ a set of views and $\mathcal{C}$ a set of tgds and egds such that the chase sequence of Q with $\mathcal{C}$ always terminates. If $Q \xrightarrow{\mathcal{C}} Q_{\mathcal{C}}$, then the canonical database D_{Q}' of $Q_{\mathcal{C}}$ is obtained by turning each subgoal into a fact by replacing each variable in the body by a distinct constant, and treating the resulting subgoals as the only tuples in D_{Q}'. The set of chased view tuples $\mathcal{T}(Q, \mathcal{V}, \mathcal{C})$ is obtained by taking all the resulting tuples from $\mathcal{V}(D_{Q}')$ (i.e. the application of the view definitions $\mathcal{V}$ on D_{Q}') and restoring each introduced constant back to the original variable of $Q_{\mathcal{C}}$.

4. Certain answers in the presence of dependencies under the CWA

In order to compute the certain answers (both in the presence and absence of dependencies) under the closed world assumption, all we need is an equivalent rewriting of the query as the following proposition shows:

Proposition 2. *For a UCQ query Q and views $\mathcal{V}$, an equivalent rewriting of Q using $\mathcal{V}$ computes all the certain answers under the Closed World Assumption, both in the presence and absence of dependencies.*

Proof. Let Q be a conjunctive query, $\mathcal{V}$ a set of views, $\mathcal{C}$ a set of dependencies and $\mathcal{I}$ a view instance. Suppose now that R is an equivalent rewriting of Q using $\mathcal{V}$ in the presence of $\mathcal{C}$. Then, for all databases D that satisfy $\mathcal{C}$ we have that $R(\mathcal{V}(D)) = Q(D)$. The following hold: $t \in \text{certain}(Q, \mathcal{I}) \Leftrightarrow \forall D$ such that $\mathcal{I} = \mathcal{V}(D)$ and $D \models \mathcal{C} : t \in Q(D) \Leftrightarrow \forall D$ such that $\mathcal{I} =$

$\mathcal{V}(D)$ and $D \models \mathcal{C} : t \in R(\mathcal{V}(D)) \Leftrightarrow \forall D$ such that $\mathcal{I} = \mathcal{V}(D)$ and $D \models \mathcal{C} : t \in R(\mathcal{I})$. Thus, $R(\mathcal{I}) = \text{certain}(Q, \mathcal{I})$. □

In this section, we deal with the problem of finding equivalent CQ rewritings in the presence of dependencies. We focus on finding CQ rewritings such that the number of subgoals they have is minimal, since these rewritings will have fewer number of joins and thus if we use them to compute the certain answers we will be able to do it efficiently. The only known efficient algorithm for finding equivalent rewritings in the presence of dependencies with the minimal number of subgoals is [15]. It has been proved in the literature that in many cases, the shared variable property (Definition 9) speeds up significantly the process of creating such a minimal rewriting, by reducing the search space (for example in Minicon [26], CoreCover [7] and the Shared Variable Bucket algorithm [23]). In this section, we propose a novel algorithm CoreCover$\mathcal{C}$ that uses the shared variable property to find equivalent CQ rewritings in the presence of dependencies. Before presenting CoreCover$\mathcal{C}$, it is good to have in mind a naive algorithm so that some of the techniques we use will be more clearly described in the simple case, i.e. in the case without optimization concerns. The naive algorithm finds an equivalent rewriting of Q under $\mathcal{C}$, such that R is a conjunctive query. As it is shown in [5,24], such a rewriting may not always exist in the language of CQs, but may exist in a more expressive query language. We now give such an example:

Example 3. Suppose that $\mathcal{C} = \emptyset$, and that we have the following query:

$$Q(X,Y) : -a(X,Z_1), a(Z_1,Z_2), a(Z_2,Z_3), a(Z_3,Z_4), a(Z_4,Y)$$

and the following two views:

$$V_3(X,Y) : -a(X,Z_1), a(Z_1,Z_2), a(Z_2,Y)$$
$$V_4(X,Y) : -a(X,Z_1), a(Z_1,Z_2), a(Z_2,Z_3), a(Z_3,Y)$$

Then, the naive algorithm will first create the CQ rewriting

$$R(X,Y) : -V_3(X,Z_3), V_3(Z_1,Z_4), V_3(Z_2,Y), V_4(X,Z_4), V_4(Z_1,Y)$$

If we compute the expansion of R we will get

$$\begin{aligned} R^{exp}(X,Y) : -a(X,Z_1'), a(Z_1',Z_2'), a(Z_2',Z_3), \\ a(Z_1,Z_2''), a(Z_2'',Z_3'), a(Z_3',Z_4), \\ a(Z_2,Z_3''), a(Z_3'',Z_4'), a(Z_4',Y), \\ a(X,Z_1''), a(Z_1'',Z_2'''), a(Z_2''',Z_3'''), \\ a(Z_3''',Z_4), a(Z_1,Z_2''''), a(Z_2'''',Z_3''''), \\ a(Z_3'''',Z_4''), a(Z_4'',Y) \end{aligned}$$

and it is easy to see that there is no containment mapping $\mu : Q \to R^{exp}$, i.e. $R_{\mathcal{C}}^{exp} \not\equiv Q$. Thus, there is no equivalent rewriting that is a conjunctive query. However, the following first-order query is an equivalent rewriting of Q using V_3, V_4: $\phi(X,Y) : -\exists Z[P_4(X,Z) \wedge \forall W(P_3(W,Z) \to P_4(W,Y))]$.

4.1. Naive algorithm

We present a naive algorithm (which is a slight variant of the algorithm in [15]) for finding equivalent rewritings in the presence of a set of weakly acyclic dependencies $\mathcal{C}$ under the closed world assumption. The Naive algorithm works as follows:

1. Chase query Q with $\mathcal{C}$ and create $Q_{\mathcal{C}}$. Compute the canonical database D'_Q of $Q_{\mathcal{C}}$ and the set $\mathcal{T}(Q,\mathcal{V},\mathcal{C})$ of chased view tuples.
2. Create rewriting R by joining all view tuples in $\mathcal{T}(Q,\mathcal{V},\mathcal{C})$.
3. Compute the expansion R^{exp} of R and chase R^{exp} with $\mathcal{C}$ to create $R_{\mathcal{C}}^{exp}$. Check whether $R_{\mathcal{C}}^{exp} \sqsubseteq Q$. If yes, then R is an equivalent rewriting, or else there is no conjunctive query that is an equivalent rewriting under $\mathcal{C}$.

Example 4. To demonstrate the algorithm we give the following example: suppose the database schema R consists of the binary relations a, b, c, d, e, the set $\mathcal{C}$ consists of the tgd $d : b(x,y) \wedge c(y,z) \to e(x,z)$, and the set $\mathcal{V}$ consists of two views $\{V_1, V_2\}$, where $V_1(z,y) : -a(z,x), b(x,y)$ and $V_2(y,w) : -c(y,z), d(z,w)$. We are given the query $Q(y) : -a(z,x), b(x,y), c(y,z), d(z,w), e(x,z)$. In this case, $Q_{\mathcal{C}}$ is the same as Q, the canonical database is $D'_Q = \{a(c_z,c_x), b(c_x,c_y), c(c_y,c_z), d(c_z,c_w), e(c_x,c_z)\}$ and the set of view tuples is $\mathcal{T}(Q,\mathcal{V},\mathcal{C}) = \{v_1(c_z,c_y),\ v_2(c_y,c_w)\}$. Thus, the algorithm will create the rewriting $R(y) : -V_1(z,y), V_2(y,w)$.

Note that $R^{exp}(D)$ will not always be contained in $Q(D)$ for databases that do not satisfy the constraints $\mathcal{C}$. To see this, consider the database $D = \{a(1,2),\ b(2,3),\ c(3,4),\ d(4,5)\}$. Then $D \not\models \mathcal{C}$ since $e(2,4) \notin D$. Thus, $R^{exp}(D) = \{3\}$ and $Q(D) = \emptyset$, so clearly $R^{exp}(D) \not\subseteq Q(D)$.

We will now prove the correctness of the naive algorithm, but first we need a slightly different definition of the chased tuple core:

Definition 7. Let t_v be a view tuple of a view V for a query Q. A tuple-core of t_v is a maximal collection G of subgoals in the query Q such that there is a containment mapping μ *from* $(t_v^{exp})'$ *of* t_v *to* G, and μ has the same properties as in Definition 10.

Theorem 1. *Let Q be a conjunctive query, $\mathcal{C}$ a set of weakly acyclic dependencies and $\mathcal{V}$ a set of CQ views. The rewriting R that is computed from the* Naive *algorithm is a CQ equivalent rewriting of Q using $\mathcal{V}$ under constraints $\mathcal{C}$.*

Proof. We need to show that for all databases D that satisfy the constraints $\mathcal{C}$ the following hold: $R(\mathcal{V}(D)) \subseteq Q(D)$ and $Q(D) \subseteq R(\mathcal{V}(D))$

- $R(\mathcal{V}(D)) \subseteq Q(D)$.
  We will prove the above in three steps:
  1. We will first show that $R(\mathcal{V}(D)) \subseteq R^{exp}(D)$.
  2. Then, from Proposition 1 we have that $R^{exp}(D) \subseteq_{\mathcal{C}} R_{\mathcal{C}}^{exp}(D)$.
  3. Finally, from our hypothesis we have that $R_{\mathcal{C}}^{exp} \sqsubseteq Q$.

  We now prove that $R(\mathcal{V}(D)) \subseteq R^{exp}(D)$. Note that for this to hold D does not need to satisfy the constraints. Suppose that $\mathcal{V} = \{V_1, V_2, \ldots, V_n\}$ and let $t_0 \in R(\mathcal{V}(D))$. Then there exists a homomorphism $h_1 : Var(R) \to Const(\mathcal{V}(D))$ that satisfies the following conditions:

- h_1 maps all the variables of R to constants in $\mathcal{V}(D)$.
- If $V_j(z'_1,\ldots,z'_m) \in Body(R)$, then $V_j(h_1(z'_1),\ldots,h_1(z'_m)) \in \mathcal{V}(D)$.
- If $Head(R)=(x'_1,\ldots,x'_k)$, then $t_0=(h_1(x'_1),\ldots,h_1(x'_k))$.

If a tuple $t=V_j(h_1(z'_1),\ldots,h_1(z'_m)) \in \mathcal{V}(D)$, then there is a homomorphism $h^t_j : Var(V_j)\to Const(D)$ with the following properties:

- h^t_i maps all the variables of V_i to constants in D.
- If $P_j(z_1,\ldots,z_m) \in Body(V_i)$, then $P_j(h^t_i(z_1),\ldots,h^t_i(z_m)) \in D$.
- If $Head(V_j)=(x_1,\ldots,x_k)$, then $t=(h^t_i(x_1),\ldots,h^t_i(x_k))$.

Note that if variable x appears in the head of V_j, then $h^t_j(x)=h_1(x)$, because if $Head(V_j)=(z'_1,\ldots,z'_m)$, then by the definitions of h_1 and h^t_j we have that $V_j(h_1(z'_1),\ldots,h_1(z'_m))=V_j(h^t_j(z'_1),\ldots,h^t_j(z'_m))$.
We extend all homomorphisms $h^t_1,\ldots,h^t_n$ to a mapping $h : Var(R^{exp})\to Const(D)$, such that the restriction of h to the domain of the subgoals of V_j is h^t_j. We now show that h is well-defined, which means that there is no variable x of R^{exp} such that for some $1\le i<j\le n : h^t_i(x)\neq h^t_j(x)$. We consider the following two cases:

1. x is a non-distinguished variable: In this case, $h^t_i(x)$ can only be defined in one of the homomorphisms h^t_i, because when we produce the expansion R^{exp} we use fresh variables for the non-distinguished variables in the definitions of the views.
2. x is a distinguished variable: Let $h^t_i(x)$ be defined in the homomorphisms $h^t_{i_1},\ldots,h^t_{i_k}$. Because x is distinguished, it must appear in all the heads of the views $V_{i_1},\ldots,V_{i_k}$. But we have already noted that if variable x appears in the head of V_j, then $h^t_j(x)=h_1(x)$, so $h^t_{i_1}(x)=\cdots=h^t_{i_k}(x)=h_1(x)$.

Hence, because each $h^t_j : Var(V_j)\to Const(D)$ is a homomorphism, $h : Var(R^{exp})\to Const(D)$ is also a homomorphism. Since R and R^{exp} have the same head variables $(x'_1,\ldots,x'_k)$ and h is a homomorphism, we have that $(h(x'_1),\ldots,h(x'_k)) \in R^{exp}(D)$. But every x'_i is a distinguished variable and h is an extension of the homomorphisms $h^t_1,\ldots,h^t_n$, so $h(x)=h_1(x)$. Thus, $(h(x'_1),\ldots,h(x'_k))=(h_1(x'_1),\ldots,h_1(x'_k))=t_0$ and $t_0 \in R^{exp}(D)$. Hence, $R(\mathcal{V}(D))\subseteq R^{exp}(D)$.

- $Q(D)\subseteq R(\mathcal{V}(D))$.
  We will prove the above in three steps, and we give an outline of the proof:
  1. Using Proposition 1 we have that for all databases D that satisfy the constraints $\mathcal{C}$: $Q(D)\subseteq_{\mathcal{C}} Q_{\mathcal{C}}(D)$.
  2. Then we prove that for all databases D : $Q_{\mathcal{C}}(D)\subseteq R^{exp}(D)$. To do this, we will use the mappings from the expansion of each view tuple t_v to the $Body(Q_{\mathcal{C}})$. We have these mappings from the definition of the view tuples and they are one-to-one, so we compose all of them into one mapping that is from the body of R^{exp} to $Q_{\mathcal{C}}$, which gives us that for all databases D, $Q_{\mathcal{C}}(D)\subseteq R^{exp}(D)$.
  3. Finally, we prove for all databases D : $R^{exp}(D)\subseteq R(\mathcal{V}(D))$. The idea is to take the homomorphism $h : Var(R^{exp})\to Const(D)$ that computes an answer $t \in R^{exp}(D)$ and restrict it to the variables of R, in order to create a homomorphism $h' : Var(R)\to Const(\mathcal{V}(D)$ that will compute the same answer $t \in R(\mathcal{V}(D))$, and thus $R^{exp}(D)\subseteq R(\mathcal{V}(D))$. This needs one extra step, which is to unify the subgoals of the body of R^{exp} that appear in the body of a view definition V_i with the subgoals in the body of the view definition V_i, since the variables in the two definitions need not be the same.

We will now show that for all databases D : $Q_{\mathcal{C}}(D)\subseteq R^{exp}(D)$. For each view tuple $t_v \in \mathcal{T}(Q,\mathcal{V},\mathcal{C})$ there exists a maximal collection G_v of subgoals in the query $Q_{\mathcal{C}}$, such that there is a containment mapping μ_{t_v} from the expansion t^{exp}_v to G_v and μ_{t_v} has the properties of Definition 7. We extend all those mappings μ_{t_v} to a mapping $h : R^{exp}\to Q_{\mathcal{C}}$. We now show that h is well-defined, i.e. there is no variable X of R^{exp} such that for some $t_v, t'_v \in \mathcal{T}(Q,\mathcal{V},\mathcal{C}), t_v\neq t'_v : \mu_{t_v}(X)\neq\mu_{t'_v}(X)$. We consider the following two cases:

- X is a distinguished variable: from property (2) of Definition 7, we have that $\forall t_v \in \mathcal{T}(Q,\mathcal{V},\mathcal{C}) : \mu_{t_v}(X)=X$, thus there cannot exist $t_v, t'_v \in \mathcal{T}(Q,\mathcal{V},\mathcal{C}), t_v\neq t'_v$ such that $\mu_{t_v}(X)\neq\mu_{t'_v}(X)$.
- X is a non-distinguished variable: from property (3) of Definition 7,we have that X will only be defined at one of the containment mappings μ_{t_v} where $t_v \in \mathcal{T}(Q,\mathcal{V},\mathcal{C})$, thus there cannot exist $t_v, t'_v \in \mathcal{T}(Q,\mathcal{V},\mathcal{C}), t_v\neq t'_v$ such that $\mu_{t_v}(X)\neq\mu_{t'_v}(X)$.

Hence, because each $\mu_{t_v} : t^{exp}_v\to G_v$ is a containment mapping, R^{exp} and $Q_{\mathcal{C}}$ have the same head variables and the body of R^{exp} contains exactly the expansion of every view tuple in $\mathcal{T}(Q,\mathcal{V},\mathcal{C})$, we have that $h : R^{exp}\to Q_{\mathcal{C}}$ is a containment mapping. Thus, for all databases D : $Q_{\mathcal{C}}(D)\subseteq R^{exp}(D)$.
To complete the proof, we will now prove that $R^{exp}(D)\subseteq R(\mathcal{V}(D))$. Suppose that $t \in R^{exp}(D)$. Then, there is a homomorphism $h : Var(R^{exp})\to Const(D)$, such that

- h maps all the variables of R^{exp} to constants in D.
- If $P_j(z_1,\ldots,z_m) \in Body(R^{exp})$, then $P_j(h(z_1),\ldots,h(z_m))$ is a fact of D, i.e. $P_j(h(z_1),\ldots,h(z_m)) \in D$.
- If $Head(R^{exp})=(x_1,\ldots,x_k)$, then $t=(h(x_1),\ldots,h(x_k)) \in R^{exp}(D)$.

Since R^{exp} is the expansion of R, there is a substitution s with the property that if V_i appears in the body of R and $R_{ij}(x_1,\ldots,x_k)$ is a subgoal of V_i, and if $R_{ij}(x'_1,\ldots,x'_l)$ is the subgoal in the body of R^{exp} with the same predicate name, then s unifies $R_{ij}(x_1,\ldots,x_k)$ with $R_{ij}(x'_1,\ldots,x'_k)$, i.e. $R_{ij}(x_1,\ldots,x_k)=s(R_{ij}(x'_1,\ldots,x'_k))$. It is clear that since R and R^{exp} have the same distinguished variables, if x is a distinguished variable then $s(x)=x$.
For each $V_i \in Body(R)$, the mapping $h' : Var(V_i)\to Const(\mathcal{V}(D))$, where $h'=h\circ s$, is a homomorphism, because from the definition of h it maps all variables

of V_i to constants in $\mathcal{V}(D)$, and if $R_{ij}(x_1,\ldots,x_m) \in Body(V_i)$ then $R_{ij}(s(x_1),\ldots,s(x_k)) \in Body(R^{exp})$ and from the definition of h we have that $R_{ij}(h(s(x_1)),\ldots,\ h(s(x_k))) \in D$. Thus, for each $V_i(y_1,\ldots,y_m) \in Body(R)$, $V_i(h(s(y_1)),\ldots,$ $h(s(y_m))) \in \mathcal{V}(D)$.
We consider the mapping $h'' : Var(R) \to Const(\mathcal{V}(D))$ which is the restriction of h' on the variable of R, i.e. $\forall x \in Var(R)\ h''(x) = h(x)$. The mapping h'' is well defined because all of the variables of R appear in R^{exp}, and h'' maps all the variables of R to constants in $\mathcal{V}(D)$. If $V_i \in Body(R)$, then we have that $V_i(h''(y_1),\ldots,$ $h''(y_m)) \in \mathcal{V}(D)$. Thus, h'' is a homomorphism, and since $Head(R) = Head(R^{exp}) = (x_1,\ldots,x_k)$, we have that $(h''(x_1),\ldots,h''(x_k)) \in R(\mathcal{V}(D))$. But $x_1,\ldots,x_k$ are distinguished variables, thus $(h''(x_1),\ \ldots,h''(x_k)) = (h(s(x_1)),$ $\ldots,h(s(x_k))) = (h(x_1),\ldots,\quad h(x_k)) = t$, i.e. $t \in R(\mathcal{V}(D))$. Hence $R^{exp} \subseteq R(\mathcal{V}(D))$.

Thus, for all databases that satisfy $\mathcal{C}$, we have that $R(\mathcal{V}(D)) = Q(D)$. □

Theorem 2. *Let Q be a conjunctive query, $\mathcal{C}$ a set of weakly acyclic dependencies and $\mathcal{V}$ a set of CQ views. The* Naive *algorithm is complete in the sense that if there is a conjunctive query R that is an equivalent rewriting of Q using $\mathcal{V}$ under constraints $\mathcal{C}$, then it will find a CQ equivalent rewriting R' of Q using $\mathcal{V}$ under constraints $\mathcal{C}$.*

Proof. Suppose that R is a conjunctive query that is an equivalent rewriting of Q under constraints $\mathcal{C}$. We will create an equivalent rewriting R_0 of Q under $\mathcal{C}$ from R. R_0 will have the same head as R and will consist only of view tuples.

Since R^{exp} is the expansion of R, there is a substitution s with the property that if V_i appears in the body of R and $P_{ij}(x_1,\ldots,x_k)$ is a subgoal of V_i, and if $P_{ij}(x'_1,\ldots,x'_l)$ is the sugboal in the body of R^{exp} with the same predicate name, then s unifies $P_{ij}(x_1,\ldots,x_k)$ with $P_{ij}(x'_1,\ldots,x'_k)$, i.e. $P_{ij}(x_1,\ldots,x_k) = s(P_{ij}(x'_1,\ldots,x'_k))$. It is clear that since R and R^{exp} have the same distinguished variables, if x is a distinguished variable then $s(x) = x$.

We chase R^{exp} with $\mathcal{C}$ and produce $R^{exp}_{\mathcal{C}}$. Since R is an equivalent rewriting of Q under constraints $\mathcal{C}$, there is a containment mapping $\mu : R^{exp}_{\mathcal{C}} \to Q_{\mathcal{C}}$. Let V_i be a view that appears in the rewriting R, and $P_{ij}(y_1,\ldots,y_k) \in Body(V_i)$. Then we have that $P_{ij}(s(y_1),\ldots,s(y_k)) \in Body(R^{exp})$ and moreover $P_{ij}(s(y_1),\ldots,s(y_k)) \in Body(R^{exp}_{\mathcal{C}})$. From the definition of μ we have that $P_{ij}(\mu(s(y_1)),\ldots,\mu(s(y_k))) \in Body(Q_{\mathcal{C}})$. Thus, we have that $P_{ij}(\mu(s(y_1)),\ldots,\mu(s(y_k))) \in D_{Q'}$. Since this holds for every subgoal P_{ij} in the body of V_i we have that $V_i(\mu(s(x_1)),\ldots,\mu(s(x_n))) \in \mathcal{T}(Q,\mathcal{V},\mathcal{C})$.

To create the body of R_0 we do the following: if $V_i(\vec{x})$ is a subgoal in the body of R, then we add $V_i(\mu(s(\vec{x})))$ to the body of R_0, and we repeat this procedure for all subgoals in the body of R. We now show that R_0 is an equivalent rewriting of Q under $\mathcal{C}$.

Since the body of R_0 consists only of view tuples, there will be a substitution s_2 of variables that unifies each subgoal in the body of R^{exp}_0 with each subgoal in the body of $Q_{\mathcal{C}}$ with the same predicate name. We now define the mapping $h : R^{exp}_{\mathcal{C}} \to R^{exp}_0$, where $h(x) = (s_2 \circ \mu \circ s)(x)$. We have that $\mu \circ s$ maps each subgoal in the body of $R^{exp}_{\mathcal{C}}$ to a subgoal in the body of $Q_{\mathcal{C}}$ and s_2 maps each subgoal in the body of $Q_{\mathcal{C}}$ to a subgoal in the body of R^{exp}_0. Thus, since $R^{exp}_{\mathcal{C}}$ and R^{exp}_0 have the same head variables, h is a containment mapping from $R^{exp}_{\mathcal{C}}$ to R^{exp}_0, so $R^{exp}_0 \sqsubseteq R^{exp}_{\mathcal{C}}$. From our hypothesis we have that R is an equivalent rewriting of Q, so $R^{exp}_{\mathcal{C}} \sqsubseteq Q$, thus $R^{exp}_0 \sqsubseteq Q$.

The other direction is trivial, because $s_2 : R^{exp}_0 \to Q_{\mathcal{C}}$ is a containment mapping (since it maps each subgoal in the body of R^{exp}_0 with each subgoal in the body of $Q_{\mathcal{C}}$ with the same predicate name). Hence, $Q_{\mathcal{C}} \sqsubseteq R^{exp}_0$ and R_0 is an equivalent rewriting of Q under constraints C. □

4.2. Algorithm CoreCover$\mathcal{C}$ for weakly acyclic LAV constraints

In this section we will present an algorithm that finds equivalent rewritings in the presence of weakly acyclic LAV dependencies, such that the rewritings have the minimum number of subgoals. The main motivation for this is to minimize the number of join operations, which tend to be expensive in practice, when a rewriting is evaluated. The Naive algorithm does not always compute the optimal equivalent rewriting (optimal with respect to the number of joins) as it is shown by the following example:

Example 5. Suppose that $q(x) : -a(x), b(x)$, $\mathcal{C} = \{d_1,\ d_2,\ d_3\}$ where $d_1 : a(x), b(x) \to c(x)$, $d_2 : c(x) \to d(x)$, $d_3 : d(x) \to e(x)$ and $\mathcal{V} = \{V_1, V_2, V_3\}$, where $V_1(x) : -a(x), b(x), d(x)$, $V_2(x) : -c(x)$, $V_3(x) : -e(x)$. Then, since $q_{\mathcal{C}}(x) : -a(x), b(x), c(x), d(x), e(x)$, the previous algorithm will create the rewriting $R(x) : -V_1(x), V_2(x), V_3(x)$, whereas the optimal rewriting in this case is $R'(x) : -V_1(x)$ (note that R' is a rewriting of Q only in the presence of dependencies $\mathcal{C}$).

We now define the class of weakly acyclic LAV tuple generating dependencies:

Definition 8. We define $\mathcal{C}^w_{\mathsf{LAV}}$ to be the set of LAV (local-as-view) tuple generating dependencies that consists only of weakly acyclic tuple generating dependencies [17] of the form $\forall\vec{x}(A(\vec{x}) \to \exists\vec{y}\psi(\vec{x},\vec{y}))$, i.e. where the left hand side consists of a single atom $A(\vec{x})$.

It is clear that $\mathcal{C}^w_{\mathsf{LAV}}$ contains all weakly acyclic inclusion dependencies. Algorithm CoreCover$\mathcal{C}$ finds equivalent rewritings with the minimal number of subgoals under a set of dependencies $\mathcal{C}$ that is a subset of $\mathcal{C}^w_{\mathsf{LAV}}$. In fact, CoreCover$\mathcal{C}$ works for a more general class of dependencies $\mathcal{C}'$, which consists of all dependencies such that the shared variable property holds on every rewriting that is considered under $\mathcal{C}'$, no matter what the query Q is.

4.2.1. Shared variable property

The efficient techniques that we use in CoreCover$\mathcal{C}$ are based on the following property:

Definition 9 (*Shared variable property*). Let Q be a conjunctive query with n subgoals, $\mathcal{V}$ be a set of CQ views and $\mathcal{C}$

a set of dependencies such that the chase of Q with $\mathcal{C}$ terminates. Suppose that R is a CQ contained rewriting of Q using $\mathcal{V}$ in the presence of $\mathcal{C}$.

We say that R has the shared variable property with respect to $Q, \mathcal{C}$ if for every homomorphism $h : Q_{\mathcal{C}} \to R_{\mathcal{C}}^{exp}$, there is a partition $\{Q_1, \ldots, Q_n\}$ of $Q_{\mathcal{C}}$'s subgoals and h can be *decomposed* into n homomorphisms $h_1, \ldots, h_n$, in the following sense:

- $h_i : Q_i \to (g_i^{exp})'$, where g_i is the i-th subgoal of R and $(g_i^{exp})'$ is the set of atoms we obtain after chasing the expansion of g_i with $\mathcal{C}$.
- For every variable x defined in h_i, then $h_i(x) = h(x)$.

We say that a class of dependencies $\mathcal{C}$ has the shared variable property if for any CQ query Q and set of CQ views $\mathcal{V}$ the following holds: if R is a CQ contained rewriting of Q using $\mathcal{V}$ in the presence of $\mathcal{C}$, then R has the shared variable property.

The following example shows how we can decompose a homomorphism if the shared variable property holds:

Example 6. Let $q(x) : -a(x, y_1), b(y_2, y)$. Suppose the set $\mathcal{V}$ contains the views $V_1(x) : -a(x, y_1), a(y_1, x)$ and $V_2(x) : -b(x, y_2), b(y_2, x)$, and that $\mathcal{C}$ contains $d_1 : a(x, y_1) \to a(y_1, x)$ and $d_2 : b(x, y_2) \to b(y_2, x)$. We have that $q_{\mathcal{C}}(x) : -a(x, y_1), a(y_1, x), b(x, y_2), b(y_2, x)$ and thus $r(x) : -V_1(x), V_2(x)$ is an equivalent rewriting under $\mathcal{C}$. The only possible homomorphism $h : Q_{\mathcal{C}} \to R_{\mathcal{C}}^{exp}$, where $R_{\mathcal{C}}^{exp}(x) : -a(x, z_1), a(z_1, x), b(x, z_2), b(z_2, x)$, is $h(x) = x$, $h(y_1) = z_1$ and $h(y_2) = z_2$. It is clear that h can be decomposed into $h_1 : \{a(x, y_1), a(y_1, x)\} \to \{a(x, z_1), a(z_1, x)\}$ and $h_2 : \{b(x, y_2)\ b(y_2, x))\} \to \{b(x, z_2), b(z_2, x)\}$, where $h_1(x) = h_2(x) = x$, $h_1(y_1) = z_1$ and $h_2(y_2) = z_2$.

Unfortunately, as the next theorem shows, there are cases when $\mathcal{C} \neq \emptyset$ and the shared variable property does not hold for any of the candidate rewritings.

Theorem 3. *There exists a query Q, a view set $\mathcal{V}$ and a set of functional dependencies $\mathcal{C}$ such that for every equivalent rewriting of Q using $\mathcal{V}$ under $\mathcal{C}$ the shared variable property does not hold.*

Proof. To prove Theorem 3, we give an example of such a query Q, set of views $\mathcal{V}$ and set of functional dependencies $\mathcal{C}$:

Example 7. We revisit Example 1, where $R^{exp}(N, T) : -\mathsf{Dept_A}(N), \mathsf{Dept_B}(N), \mathsf{Staff}(N, T, B), \mathsf{Staff}(N, T', T)$. We chase R^{exp} with the functional dependencies $\mathcal{C}$ and we have that $R_{\mathcal{C}}^{exp}(N, T) : -\mathsf{Dept_A}(N), \mathsf{Dept_B}(N), \mathsf{Staff}(N, T, T)$. Since $Q_{\mathcal{C}}(N, T) : -\mathsf{Dept_A}(N), \mathsf{Dept_B}(N), \mathsf{Staff}(N, T, T)$, there is only one possible homomorphism $h : Q_{\mathcal{C}} \to R_{\mathcal{C}}^{exp}$, where $h(N) = N$ and $h(T) = T$. However, it is clear that we cannot decompose h into two homomorphisms h_1, h_2 as in Definition 9, since then we would have that either $h_1(T) = T$ and $h_1(T) = B$ or $h_2(T) = T'$ and $h_2(T) = T$, which both lead to contradiction.

We next prove that the shared variable property will hold on all equivalent rewritings under a set $\mathcal{C} \subseteq \mathcal{C}_{\mathsf{LAV}}^{w}$:

Theorem 4. *Let Q be a CQ and $\mathcal{V}$ a set of CQ views. If R is a contained rewriting of Q using $\mathcal{V}$ under a set of tgds $\mathcal{C} \subseteq \mathcal{C}_{\mathsf{LAV}}^{w}$, then R has the shared variable property.*

Proof. Let R be a contained rewriting of Q using $\mathcal{V}$ under a set of tgds $\mathcal{C} \subseteq \mathcal{C}_{\mathsf{LAV}}^{w}$. We chase the expansion of R with $\mathcal{C}$ and we create $R_{\mathcal{C}}^{exp}$. Suppose that h is a homomorphism $h : Var(Q_{\mathcal{C}}) \to Var(R_{\mathcal{C}}^{exp})$, and $a(\vec{x}) \in Body(Q_{\mathcal{C}})$. By the definition of the homomorphism h, we have that $a(h(\vec{x})) \in Body(R_{\mathcal{C}}^{exp})$. We will show that there exists a view subgoal g in the body of R such that $a(h(\vec{x})) \in Body(g_{\mathcal{C}}^{exp})$, where $g_{\mathcal{C}}^{exp}$ is the chase with $\mathcal{C}$ of the expansion of g. We consider the following two cases:

1. $a(h(\vec{x}))$ is not introduced by a chase step. Then, there exists a subgoal $a(\vec{y}) \in Body(R^{exp})$ such that $a(h(\vec{x})) = a(\vec{y})$, and this subgoal must belong to at least one of the expansions of the view subgoals of R. If g is such a view subgoal, then it is clear that $a(h(\vec{x})) \in Body(g_{\mathcal{C}}^{exp})$.
2. $a(h(\vec{x}))$ is introduced by a chase step. Thus, there exists a tgd $d : b(\vec{x}) \to \exists \vec{y} \psi(\vec{x}, \vec{y})$ such that a appears in the conjunction $\psi(\vec{x}, \vec{y})$ and a homomorphism $h_1 : b(\vec{x}) \to R^{exp}$ that cannot be extended to a homomorphism $h_1' : b(\vec{x}) \cup \psi(\vec{x}, \vec{y}) \to R^{exp}$. Let g be the view subgoal of R such that $h_1(b(\vec{x})) \in g^{exp}$. It is clear that h_1 will be a homomorphism $h_1 : b(\vec{x}) \to g^{exp}$ that cannot be extended to a homomorphism $h_1' : b(\vec{x}) \cup \psi(\vec{x}, \vec{y}) \to g^{exp}$,so, if we chase g with d we will have that $a(h(\vec{x}))$ will be introduced in $g_{\mathcal{C}}^{exp}$.

In any case, if $a(\vec{x}) \in Body(Q_{\mathcal{C}})$ then there exists a view subgoal g in the body of R such that $a(h(\vec{x})) \in Body(g_{\mathcal{C}}^{exp})$, thus h can be decomposed into the mappings $h_1, \ldots, h_n$ with the properties of Definition 9, which means that R has the shared variable property. □

4.2.2. CoreCover$\mathcal{C}$ *algorithm*

We propose a novel algorithm CoreCover$\mathcal{C}$ for finding equivalent rewritings in the presence of a set of tgds that has the shared variable property. First we give the definition of the tuple-core from [7], the only difference is that here we search for a mapping from the *chased query* to the *chased expansion* of a view tuple:

Definition 10. Let $t_v \in \mathcal{T}(Q, \mathcal{V}, \mathcal{C})$. A chased tuple-core of t_v is a maximal collection G of subgoals in the query $Q_{\mathcal{C}}$ such that there is a containment mapping μ from G to $(t_v^{exp})'$, where $(t_v^{exp})'$ is the chased expansion of t_v with $\mathcal{C}$, and μ has the following properties:

1. μ is a one-to-one mapping, and it maps the arguments in G that appear in t_v as the identity mapping on arguments.
2. Each distinguished variable X in G_v is mapped to a distinguished variable in $(t_v^{exp})'$ (furthermore, by property (1), $\mu(X) = X$).
3. If a non-distinguished variable X in G is mapped under μ to an existential variable in the chased expansion of t_v, then G includes all subgoals in $Q_{\mathcal{C}}$ that use this variable X.

The following proof appears in [8], but we repeat it here for self-containment.

Lemma 1. *A view tuple for a minimal query has a unique chased tuple-core.*

Proof. Suppose a view tuple t_v for a minimal query $Q_{\mathcal{C}}$ has two distinct chased tuple-cores G_1 and G_2, with the corresponding mappings μ_1 and μ_2 in Definition 10. Let $H_1 = \mu_1(G_1)$ and $H_2 = \mu_2(G_2)$ be the targets (sets of subgoals in $(t_v^{exp})'$), respectively. Either $G_1 - G_2$ or $G_2 - G_1$ is not empty (otherwise G_1 and G_2 are identical).

Suppose $G_1 - G_2$ is not empty. Then, each variable X used in $G_1 - G_2$ can be in two cases:

1. $\mu_1(X) = X$. We will show that either X is not used in G_2 or that $\mu_2(X) = X$.
2. $\mu_1(X) \neq X$. We will show that X cannot be used in G_2.

In summary, the two mappings μ_1 and μ_2 do not conflict with each other on their source variables in $Q_{\mathcal{C}}$. Therefore, we can define a mapping μ_2' from $G_1 \cup G_2 = G_2 \cup (G_1 - G_2)$ onto $H_2 = \mu_1(G_1 - G_2) \cup \mu_2(G_2)$ as follows: if X is used in G_2, $\mu_2'(X) = \mu_2(X)$; if X is used in $G_1 - G_2$, $\mu_2'(X) = \mu_1(X)$. We will also show that: (3) The mapping μ_2' is one-to-one, thus we have a larger set $G_1 \cup G_2$ of query subgoals that satisfies the conditions in Definition 10, contradicting to the fact that G_2 is maximal.

We first prove case (1). Suppose X appears in G_2, and $\mu_2(X) \neq X$. Then X must be a non-distinguished variable in $Q_{\mathcal{C}}$, because if we assume that X is a distinguished variable in the query, then $\mu_2(X) = X$ by property 2 of Definition 10, which is a contradiction. In addition, $\mu_2(X)$ must also be a non-distinguished variable in t_v^{exp}, because if we assume that $\mu(X)$ is a distinguished variable in t_v^{exp}, i.e. $\mu(X)$ appears in t_v, then by the construction of the view tuple t_v, $\mu(X)$ will also be a variable of the query $Q_{\mathcal{C}}$. Thus, by property 1 of Definition 10 we would have that $\mu(X) = X$ which is a contradiction. By G_2's definition, G_2 includes all query subgoals that use X, contradicting to the fact that X appears in $G_1 - G_2$.

Now we prove case (2). Suppose X is in G_2. By G_1's definition, X cannot be a distinguished variable in $Q_{\mathcal{C}}$, since then from property 2 of Definition 10 we would have $\mu_1(X) = X$. There are two cases:

- If $\mu_2(X)$ is a non-distinguished variable in t_v^{exp}, then by G_2's definition, G_2 should include all the query subgoals that use X, contradicting to the fact that X appears in $G_1 - G_2$.
- If $\mu_2(X)$ is a distinguished variable in t_v^{exp}, i.e. $\mu_2(X)$ appears in t_v, then by the construction of the view tuple t_v, $\mu_2(X)$ will also be a variable of the query $Q_{\mathcal{C}}$. Thus, by property 1 we would have that $\mu_2(X) = X$, i.e. X appears in t_v. Since X appears in G_1, by property 2, we have that $\mu_1(X) = X$, which is a contradiction.

In the rest of the proof, we prove claim (3): Since t_v is a view tuple, there is a mapping λ from $(t_v^{exp})'$ to $Q_{\mathcal{C}}$. Suppose mapping μ_2' is not one-to-one. Then, mapping $\mu_2' \circ \lambda$ is a mapping from $G_1 \cup G_2$ to $Q_{\mathcal{C}}$ which is not one-to-one either. We show that we can extend $\mu_2' \circ \lambda$ to a mapping from $Q_{\mathcal{C}}$ to $Q_{\mathcal{C}}$ which is not one-to-one, contradicting the fact that $Q_{\mathcal{C}}$ is minimal. For this extension to be feasible, we need to show: (i) no distinguished variable of $Q_{\mathcal{C}}$ is mapped on another distinguished variable of $Q_{\mathcal{C}}$ under $\mu_2' \circ \lambda$ and (ii) if $G_1 \cup G_2$ shares a variable X with a subgoal not in $G_1 \cup G_2$, then $\mu_2'(X) = X$. If (i) and (ii) hold, then we easily extend $\mu_2' \circ \lambda$ by having the variables not in $G_1 \cup G_2$ mapped each on itself.

We prove (i): if mapping μ_2' is not one-to-one, then, there exist variables X used in $G_1 - G_2$ and not used in G_2 and, X' used in G_2 such that $\mu_1(X) = \mu_2(X')$. Then, either X or X' is a non-distinguished variable of $Q_{\mathcal{C}}$.

We prove (ii): if $G_1 \cup G_2$ shares a variable X with a subgoal which is not in $G_1 \cup G_2$, then X is a variable in the view tuple t_v. Hence $\mu_2'(X) = X$. □

The unique chased tuple-core of a view tuple t_v is denoted by $\mathcal{C}(t_v)$. We also need to define the notion of a sub-core of the chased tuple-core of a view tuple:

Definition 11 (*Partition in sub-cores*). Let C be a tuple-core. Let $C_1, C_2, \ldots$ be a partition of C such that each C_i has all the properties of the chased tuple-core (see Definition 10) except that it is minimal (instead of maximal), i.e. any proper subset of C_i does not have properties 1–3 of Definition 10. Then, each C_i is called a sub-core of C.

We now describe CoreCover$\mathcal{C}$:

1. Chase query Q with $\mathcal{C}$ and create $Q_{\mathcal{C}}$.
2. Minimize $Q_{\mathcal{C}}$ by removing redundant subgoals. Let Q_m be the minimal equivalent of $Q_{\mathcal{C}}$.
3. Construct the canonical database D'_Q of Q_m and compute the view tuples $\mathcal{T}(Q, \mathcal{V}, \mathcal{C})$ by applying the view definitions $\mathcal{V}$ on the database D'_Q.
4. For each view tuple $t_v \in \mathcal{T}(Q, \mathcal{V}, \mathcal{C})$, compute its chased tuple-core $\mathcal{C}(t_v)$.
5. Use the non-empty tuple-cores to p-cover the query subgoals in Q_m with the minimum number of tuple-cores. For each cover, construct a rewriting by combining the corresponding view tuples.

This algorithm works like the naive algorithm, but it uses the chased tuple-cores to find a minimum number of view tuples that p-cover the query subgoals (this can be done using a classic algorithm for the set-covering problem [14]). We say that the chased tuple-cores p-cover the query subgoals when the union of the chased tuple-cores is equal to all of the query subgoals and there is a collection of sub-cores of the corresponding view tuples that comprises a partition of all the query subgoals in Q_m. Notice that by the construction of the chased tuple-cores, we do not need to chase or to perform an additional containment mapping check as before.

The relation of CoreCover$\mathcal{C}$ and CoreCover [7] is the following: CoreCover$\mathcal{C}$ is a slight modification of CoreCover in that CoreCover$\mathcal{C}$ uses the minimal chased query

and the chased view tuples and chased tuple-cores to create equivalent rewritings. The proof techniques for the correctness of CoreCover$\mathcal{C}$ are largely along the lines of the proof of the correctness of CoreCover, with a few subtle points which, however, are crucial in identifying the correct class of dependencies for which the algorithm works.

Remark. Note that we need to introduce the notion of sub-core[1] in order for the algorithm to be sound. This means that CoreCover in [8] needs to be rectified accordingly in order to be sound, i.e. for Theorem 4.1 in [8] (that corresponds to Theorem 5 in the present paper) to be correct.[2]

We now give an example to demonstrate how CoreCover$\mathcal{C}$ works:

Example 8. Suppose that $\mathcal{V}$ contains the following four views:

$V_1(X_1,X_2) : -a(X_1,Z_4), b(Z_4,X_2)$
$V_2(X_1,X_2) : -b(X_1,Z_5), c(Z_5,X_2)$
$V_3(X_1,X_2,X_3) : -b(X_1,X_2), c(X_2,X_3)$
$V_4(X_1) : -a(X_1,X_1), d(X_1,X_1)$

Suppose we have the query $Q(X,Y) : -a(X,Z_2), b(Z_2,Z_3), c(Z_3,Y), a(Y,Z_1), a(Z_1,Y), d(Y,Y)$ and that we also have the following tgd $d : d(X,Y) \rightarrow a(X,Y)$.

The chased query is:

$$Q_{\mathcal{C}}(X,Y) : -a(X,Z_2), b(Z_2,Z_3), c(Z_3,Y), a(Y,Z_1), a(Z_1,Y), a(Y,Y), d(Y,Y)$$

If we minimize $Q_{\mathcal{C}}$, we will get the query Q_m:

$$Q_m(X,Y) : -a(X,Z_2), b(Z_2,Z_3), c(Z_3,Y), a(Y,Y), d(Y,Y)$$

The canonical database is $D'_Q = \{a(x,z_2), b(z_2,z_3), c(z_3,y), a(y,y), d(y,y)\}$, and if we apply D'_Q to the view definitions we get the set of view tuples $\mathcal{T}(Q,\mathcal{V},\mathcal{C}) = \{V_1(X_1,Z_3), V_2(Z_2,Y), V_3(Z_2,Z_3,Y), V_4(Y)\}$. We next compute the chased tuple cores and sub-cores of each $t_v \in \mathcal{T}(Q,\mathcal{V},\mathcal{C})$.

- $V_1(X_1,Z_3)$: We have that $(V_1^{exp})'(X_1,Z_3) : -a(X_1,Z'_4), b(Z'_4,Z_3)$, thus the chased tuple-core is the set $\{a(X,Z_2), b(Z_2,Z_3)\}$. It is easy to see that $V_1(X_1,Z_3)$ has only one sub-core, which is $\{a(X,Z_2), b(Z_2,Z_3)\}$.
- $V_2(Z_2,Y)$: We have that $(V_2^{exp})'(Z_2,Y) : -b(Z_2,Z'_5), c(Z'_5,Y)$, thus the chased tuple-core is the set $\{b(Z_2,Z_3), c(Z_3,Y)\}$. Like the first view tuple, $V_2(Z_2,Y)$ has only one sub-core, which is $\{b(Z_2,Z_3), c(Z_3,Y)\}$.
- $V_3(Z_2,Z_3,Y)$: We have that $(V_3^{exp})'(Z_2,Z_3,Y) : -b(Z_2,Z_3), c(Z_3,Y)$, thus the chased tuple-core is the set $\{b(Z_2,Z_3), c(Z_3,Y)\}$. This time however, notice that $V_3(Z_2,Z_3,Y)$ has two sub-cores: $\{b(Z_2,Z_3)\}$ and $\{c(Z_3,Y)\}$.
- $V_4(Y)$: We have that $(V_4^{exp})'(Y) : -a(Y,Y), d(Y,Y)$, thus the chased tuple-core is the set $\{a(Y,Y), d(Y,Y)\}$. Similar to the first two view tuples, $V_4(Y)$ has only one sub-core, which is $\{a(Y,Y), d(Y,Y)\}$.

In the last step, CoreCover$\mathcal{C}$ finds a minimum number of view tuples, to p-cover the $Q_{\mathcal{C},m}$'s subgoals and constructs a rewriting by combining the corresponding view tuples. In our example, CoreCover$\mathcal{C}$ will find only one equivalent rewriting, which is $R(X,Y) : -V_1(X_1,Z_3), V_3(Z_2,Z_3,Y), V_4(Y)$. We can verify that R is an equivalent rewriting of Q using containment mappings:

$$Q_m(X,Y) : -a(X,Z_2), b(Z_2,Z_3), c(Z_3,Y), a(Y,Y), d(Y,Y)$$

$$R_{\mathcal{C}}^{exp}(X,Y) : -a(X_1,Z'_4), b(Z'_4,Z_3), b(Z_2,Z_3), c(Z_3,Y), a(Y,Y), d(Y,Y)$$

For example, if $h(X) = X$, $h(Z_2) = Z'_4$, $h(Z_3) = Z_3$, and $h(Y) = Y$, then h is a containment mapping $h : Q_{\mathcal{C},m} \rightarrow R_{\mathcal{C}}^{exp}$, which means that $Q_{\mathcal{C},m} \sqsubseteq P_{\mathcal{C}}^{exp}$ (the other direction is similar).

Notice that if we had not performed the minimization step in $Q_{\mathcal{C}}$, i.e. if we had considered $Q_{\mathcal{C}}$ instead of $Q_{\mathcal{C},m}$, then the canonical database D'_Q, the set of chased view tuples $\mathcal{T}(Q,\mathcal{V},\mathcal{C})$ and the chased tuple-cores would be the same, since the two subgoals $a(Y,Z_1), a(Z_1,Y)$ could not be part of any chased tuple-core: this is easy to see because the only view-tuple that this could happen is $V_4(Y)$, and if we include them in $\mathcal{C}(V_4(X_1))$ it is easy to see that the mapping $\mu : \{a(Y,Z_1), a(Z_1,Y), a(Y,Y), d(Y,Y)\} \rightarrow \{a(Y,Y), d(Y,Y)\}$ is not one-to-one, as it is required by the definition of the tuple-core. This means $a(Y,Z_1), a(Z_1,Y)$ would not be covered by any view tuple, and thus CoreCover$\mathcal{C}$ would not find any equivalent rewriting.

Notice also that the query $P(X,Y) : -V_1(X_1,Z_3), V_2(Z_2,Y), V_4(Y)$ is not an equivalent rewriting of Q, because the sub-cores of the view tuples in P do not comprise a partition of the Q's subgoals: the sub-cores of $V_1(X_1,Z_3)$ and $V_2(Z_2,Y)$ both contain $b(Z_2,Z_3)$. We can also verify that P is not an equivalent rewriting of Q using containment mappings:

$$Q_m(X,Y) : -a(X,Z_2), b(Z_2,Z_3), c(Z_3,Y), a(Y,Y), d(Y,Y)$$

$$P_{\mathcal{C}}^{exp}(X,Y) : -a(X_1,Z'_4), b(Z'_4,Z_3), b(Z_2,Z'_5), c(Z'_5,Y), a(Y,Y), d(Y,Y)$$

If we try to find a containment mapping $h : Q_m \rightarrow P_{\mathcal{C}}^{exp}$, we will have that $h(Z_3) = Z_3$ and $h(Z_3) = Z'_5$, which is a contradiction.

4.2.3. *Proof of correctness of* CoreCover$\mathcal{C}$

We will next prove that CoreCover$\mathcal{C}$ is sound and complete. The proof uses the following lemma:

Lemma 2. *Suppose Q is a conjunctive query and $\mathcal{V}$ is a set of CQ views, and let R be an equivalent rewriting of Q that uses only view tuples in $\mathcal{T}(Q,\mathcal{V},\mathcal{C})$. Let $Q_{\mathcal{C},m}$ be a minimal equivalent of $Q_{\mathcal{C}}$. There is a containment mapping μ from $Q_{\mathcal{C},m}$ to $R_{\mathcal{C}}^{exp}$, such that (1) μ is a one-to-one mapping, i.e., different arguments in $Q_{\mathcal{C},m}$ are mapped to different arguments in $R_{\mathcal{C}}^{exp}$; (2) for all arguments in $Q_{\mathcal{C},m}$ that appear in R, they are mapped by μ as is the identity mapping on arguments, i.e., $\mu(X) = X$ for all $X \in Var(R)$.*

[1] This is similar to the notion of tuple-coverage from [18].
[2] We thank Rada Chirkova and Michael Compton for noticing this.

(*Notice that the definition of rewriting R guarantees a containment mapping from $Q_{\mathcal{C},m}$ to $R_{\mathcal{C}}^{exp}$, but this containment mapping might not have the two properties*).

Proof. Consider a minimal equivalent query R_m^{exp} of $R_{\mathcal{C}}^{exp}$. Suppose without loss of generality that $Q_{\mathcal{C}}$ is minimal.

Notice that both $Q_{\mathcal{C}}$ and R_m^{exp} are minimal equivalents of the expansion $R_{\mathcal{C}}^{exp}$. Thus their only difference is variable renamings. By the construction of the view tuples, there is a containment mapping from $R_{\mathcal{C}}^{exp}$ to $Q_{\mathcal{C}}$, such that it maps each argument in $R_{\mathcal{C}}^{exp}$ that appears in $Q_{\mathcal{C}}$ under identity. Let v be the corresponding containment mapping from R_m^{exp} to $Q_{\mathcal{C}}$.

Since $Q_{\mathcal{C}}$ and R_m^{exp} are equivalent, there is a containment mapping τ from $Q_{\mathcal{C}}$ to R_m^{exp}. The composition of this mapping and v is a containment mapping from $Q_{\mathcal{C}}$ to $Q_{\mathcal{C}}$. Since $Q_{\mathcal{C}}$ is minimal, the composed containment mapping $\tau \circ v$ should be one-to-one and onto. Thus v should also be one-to-one and onto. Then we can reverse the mapping v, and obtain a containment mapping $\mu = v^{-1}$ from $Q_{\mathcal{C}}$ to R_m^{exp}, such that μ is one-to-one, and maps the arguments in $Q_{\mathcal{C}}$ that appear in R under identity. □

We will now prove that CoreCover$\mathcal{C}$ is sound and complete for a class of dependencies $\mathcal{C}$ that has the shared variable property:

Theorem 5. *Suppose $\mathcal{C}$ is a class of tgds such that all equivalent rewritings have the shared variable property. If R_0 is an equivalent rewriting of Q under $\mathcal{C}$ and R_0 has m subgoals, then* CoreCover$\mathcal{C}$ *will find an equivalent rewriting R of Q under $\mathcal{C}$ with n subgoals, such that $n \leq m$.*

Proof. We will first show that if $Q_{\mathcal{C},m}$ is a minimal equivalent of $Q_{\mathcal{C}}$ and R is a query that has the head of Q and uses a minimal number of view tuples in $\mathcal{T}(Q,\mathcal{V},\mathcal{C})$ in its body, then R is an equivalent rewriting of Q if and only if there is a collection $\mathcal{K}$ of sub-cores of the view tuples in R such that it comprises a partition of all the query subgoals in $Q_{\mathcal{C},m}$.

"If" direction: we construct such a rewriting and we will prove that there is a one-to-one containment mapping which maps all query subgoals to the chased expansion of the rewriting. This implies that there is a containment mapping from the query to the chased expansion too.

The proof is by induction on the number of sub-cores in $\mathcal{K}$. Inductive hypothesis: Suppose the number of elements in $\mathcal{K}$ is n and suppose that the sub-cores in $\mathcal{K}$ contain a set S_K of query subgoals. Then there is a one-to-one containment mapping from S_K to $\mathcal{K}$ such that the variables which are contained in subgoals that are not in S_K are distinguished variables in the rewriting.

Basis of the induction ($\mathcal{K}$ contains one sub-core): by the definition of the sub-core, there is a one-to-one mapping as required.

Suppose the inductive hypothesis holds for the case $\mathcal{K}$ has less than n elements. We will prove that it holds also for the case $\mathcal{K}$ has n elements. Let C_b be one of the sub-cores in $\mathcal{K}$ and let $\mathcal{K}'$ be the collection $\mathcal{K}$ after deleting C_b. By the definition of sub-core there is a one-to-one mapping μ_b from C_b to the image of C_b in the rewriting. By inductive hypothesis, there is a one-to-one mapping μ from $\mathcal{K}$ to the image of $\mathcal{K}$ in the rewriting. It remains to be proven that μ_b and μ can be combined to create a one-to-one mapping μ' from $\mathcal{K}$ to the image of $\mathcal{K}$ in the rewriting. If every variable in the query maps on the same variable according to μ_b and μ, we are done. Suppose variable X of query maps on two variables. Then, according to the construction of sub-cores, the images of X are distinguished variables of the views, hence they are equated in the rewriting and $\mu_b(X) = \mu(X)$.

"Only If": assume R is a rewriting of Q using $\mathcal{V}$. By Lemma 2, there is a one-to-one containment mapping μ from $Q_{\mathcal{C},m}$ to $R_{\mathcal{C}}^{exp}$, which maps all arguments in $Q_{\mathcal{C},m}$ that appear in R under identity. Because the shared variable property holds, μ can be decomposed into k mappings (we assume that $R_{\mathcal{C}}^{exp}$ has k subgoals) $h_i : G_i \rightarrow (t_i^{exp})'$ $i = 1,\ldots,k$, with the properties of Definition 9, and such that $G_1,\ldots,G_k$ comprise a partition of $Q_{\mathcal{C},m}$. The subgoal set G_i and the "local" mapping μ satisfies the three properties in Definition 10, except that G_i might not be maximal. According to Lemma 1, the chased tuple-core is unique, hence $G_i \subseteq C(t_i)$. Thus the union of the k tuple-cores includes all query subgoals in $Q_{\mathcal{C},m}$.

Now we need in addition to prove that there is a collection $\mathcal{K}$ of sub-cores that comprise a partition of the query subgoals. The first observation is that targets of μ are either all variables in a certain sub-core or none at all. The reason is that otherwise, either condition (3) of Definition 10 is not satisfied or it is not a minimal set of subgoals that satisfy 1–3 of Definition 10. Thus, we consider all sub-cores that contain targets of μ and claim that they are pairwise disjoint. Suppose, towards contradiction, that there is a non-empty intersection of two sub-cores in $\mathcal{K}$. Then it is easy to see that the intersection has the properties 1–3 of Definition 10, hence one of the two sub-cores is not minimal.

Suppose that R_0 is an equivalent rewriting of Q using $\mathcal{V}$ under the set of dependencies $\mathcal{C}$, and that R_0 has m subgoals. We know from Theorem 2 that we can create an equivalent rewriting R_1 from R_0 such that R_1 uses only view tuples in $\mathcal{T}(Q,\mathcal{V},\mathcal{C})$, and by the construction of R_1 it is obvious that if the number of subgoals of R_1 is n_1, then $n_1 \leq m$. The existence of such an equivalent rewriting R_1 of Q (and thus $Q_{\mathcal{C},m}$) guarantees that there is a set of view tuples in $\mathcal{T}(Q,\mathcal{V},C)$ that covers $Q_{\mathcal{C},m}$'s subgoals. Thus, if R is an equivalent rewriting of $Q_{\mathcal{C},m}$ that uses only a minimum number of view tuples in $\mathcal{T}(Q,\mathcal{V},\mathcal{C})$, then as we have shown in the previous paragraph, there is a collection $\mathcal{K}$ of sub-cores of the view tuples in R such that it comprises a partition of all the query subgoals in $Q_{\mathcal{C},m}$, which means that CoreCover$\mathcal{C}$ will output R. In addition, if the number of subgoals of R is n, then it is clear that $n \leq n_1 \Rightarrow n \leq m$. □

Theorem 5 implies that CoreCover$\mathcal{C}$ will not work in the presence of equality generating dependencies (this class contains functional dependencies). The reason for that is that in the presence of equality generating dependencies the shared variable property does not always hold, as we have already seen in Example 7.

The following corollary is a direct consequence of Theorems 4 and 5:

Corollary 2. *Suppose $\mathcal{C}$ is a class of tgds such that $\mathcal{C} \subseteq \mathcal{C}^w_{\mathsf{LAV}}$. If R_0 is an equivalent rewriting of Q under $\mathcal{C}$ and R_0 has m subgoals, then* CoreCover$\mathcal{C}$ *will find an equivalent rewriting R of Q under $\mathcal{C}$ with n subgoals, such that $n \leq m$.*

5. Certain answers in the presence of dependencies under OWA

5.1. Algorithm for finding MCRs

In this section we give an efficient algorithm which finds a maximally contained rewriting with respect to UCQ of a UCQ Q in the presence of a set of tgds $\mathcal{C} \subseteq \mathcal{C}^w_{\mathsf{LAV}}$, if such an MCR exists, since in the presence of dependencies an MCR of Q with respect to UCQ may not always exist (see [20] for an example). The inverse rules algorithm [16] also finds MCRs but in the presence of functional and full dependencies and it creates datalog MCRs, whereas our algorithm creates UCQ MCRs (if they exist).

We first give some definitions from [4]. A *subgoal mapping* is a mapping from the query subgoals to view subgoals of a view such that the predicate names match. A subgoal mapping induces an *associated argument mapping*, which maps each query variable/constant to a variable/constant in the body of the view definition, such that for each query subgoal g that is mapped to a view subgoal, their variables and constants are also mapped argument-wise. Notice that an argument mapping is not restricted to map a query variable/constant to a single view variable/constant (as in a containment mapping), since it may map a query variable/constant to several view variables/constants. Given an argument mapping, we associate with it several containment mappings. An *associated containment mapping* is a mapping from query variables/constants to view variables/constants defined by a partition P on the set of the view variables/constants into equivalence classes, in such a way that each query variable/constant is mapped to elements of a single equivalence class, all variables/constants of a query subgoal are mapped on the variables/constants of a single copy of a view, and the following three conditions hold: (a) each equivalence class with more than one element is populated by either (identical) constants and/or distinguished variables; (b) an equivalence class which is the image of a constant has only distinguished variables (even if it contains only one element) and possibly the same constant. (c) Distinguished variables map to distinguished variables.

Given a total subgoal mapping and one of its associated containment mappings (if there exists any), we can define a conjunctive query over view subgoals that uses the view copies that are involved in the associated containment mapping, where the distinguished view variables are equated according to the partition that defines the associated containment mapping. We now present MiniCon$\mathcal{C}$:

1. Chase each view $V \in \mathcal{V}$ with $\mathcal{C}$ and create $V_{\mathcal{C}}$. Let $\mathcal{V}_{\mathcal{C}}$ be the set of the chased views. Set $\mathcal{P} = \emptyset$.
2. Generate the set of MCDs [26] $\mathcal{M}$ as follows:
   For each query subgoal g in Q, for each view $V \in \mathcal{V}_{\mathcal{C}}$ and for each subgoal g' in V, try to find a containment mapping $\mu : g \to g'$ and if it exists, add to $\mathcal{M}$ a new partial MCD: $(\{g\}, \mu)$. While there are partial MCDs with shared variables, for each partial MCD (G, μ) do the following: Choose a shared variable X in G and a query subgoal g not in G that contains X. For each view subgoal g' that has an argument mapping μ' from g, extend μ with μ' and create μ'', then replace the current partial MCD with new partial MCD: $(G' = G \cup \{g\}, \mu'')$.
3. Combine the set of MCDs $\mathcal{M}$ as follows:
   For each combination of MCDs that covers all query subgoals without overlapping (i.e. the combination of MCD mappings is a total subgoal mapping), let μ be the corresponding argument mapping. Check whether there exists an associated containment mapping. If it does exist, then find the most relaxed associated containment mapping as follows: For each query variable/constant X form a class that contains all view variables/constants that are images of X under μ, and while classes are not disjoint merge classes that share an element. If there is a class containing two distinct constants this procedure fails, else return the "relaxed" classes as the most relaxed associated containment mapping μ'. Use μ' to create a contained rewriting R. Set $\mathcal{P} := \mathcal{P} \cup R$.
4. Return the UCQ MCR $\mathcal{P}$.

In the absence of constraints, MiniCon$\mathcal{C}$ reduces to the MiniCon algorithm [26]. We next prove that the algorithm will always find an MCR in the language of finite unions of conjunctive queries, if such an MCR exists. In the proof we will use the following lemmas:

Lemma 3. *Suppose that R is a CQ contained rewriting of Q using $\mathcal{V}$, i.e. for all view instances $\mathcal{I}$ and for all databases D such that $\mathcal{I} \subseteq \mathcal{V}(D)$: $R(\mathcal{I}) \subseteq Q(D)$. Then, we have that $R^{exp} \sqsubseteq Q$.*

Proof. Let D be any database instance. We take $\mathcal{I} = \mathcal{V}(D)$. To prove $R^{exp}(D) \subseteq Q(D)$, we will first show that $R^{exp}(D) \subseteq R(\mathcal{V}(D))$. To do this, we take the homomorphism $h : Var(R^{exp}) \to Const(D)$ that computes an answer $t \in R^{exp}(D)$ and restrict it to the variables of R, in order to create a homomorphism $h' : Var(R) \to Const(\mathcal{V}(D)$ that will compute the same answer $t \in R(\mathcal{V}(D))$. This needs one extra step, which is to unify the subgoals of the body of R^{exp} that appear in the body of a view definition V_i with the subgoals in the body of the view definition V_i, since the variables in the two definitions need not be the same. Hence, $R^{exp}(D) \subseteq R(\mathcal{V}(D))$ and because R is a contained rewriting of Q we have that $R(\mathcal{V}(D) \subseteq Q(D)$. Thus, $R^{exp} \sqsubseteq Q$. □

Lemma 4. *Suppose that $R(\vec{x}) : -V_1(\vec{x_1}), \ldots, V_n(\vec{x_n})$ is a contained CQ rewriting of Q in the presence of a set of dependencies $\mathcal{C} \subseteq \mathcal{C}^w_{\mathsf{LAV}}$. If $R'(\vec{x}) : -V_1^{\mathcal{C}}(\vec{x_1}), \ldots, V_n^{\mathcal{C}}(\vec{x_n})$, where $V_i^{\mathcal{C}}$ is the view that we get if we chase V_i with $\mathcal{C}$, then R^{exp} and*

$(R')^{exp}$ *compute the same answers for all databases D that satisfy* $\mathcal{C}$.

Proof. Consider a subgoal $B(\vec{z_1}) \in Body(R_{\mathcal{C}}^{exp})$ that is introduced from a chase step from a tgd $d \in \mathcal{C}_{\mathsf{LAV}}^{w}$. Let $C(\vec{z_1})$ be the atom that appears in the left hand side of d, and let V be the view atom in the body of R such that $C(\vec{z_2})$ appears in the expansion of V. Then, it is clear that $B(\vec{z_1}) \in Body(V_{\mathcal{C}}^{exp})$. The other direction is obvious, since a subgoal $B(\vec{z_3}) \in Body(V_{\mathcal{C}}^{exp})$ always appears in the body of $R_{\mathcal{C}}^{exp}$. Thus, $R_{\mathcal{C}}^{exp}$ and $(R')^{exp}$ have the same subgoals up to variable renaming and hence R and R' compute the same answers for all databases D that satisfy $\mathcal{C}$. □

Lemma 5. *Suppose that Q is a UCQ query,* $\mathcal{V}$ *a set of CQ views and* $\mathcal{C}$ *a set of weakly acyclic LAV tgds. Let* $\mathcal{V}_{\mathcal{C}}$ *be the set of chased views.*

Let $\mathcal{P}'$ *be a UCQ query over the schema of the chased views* $\mathcal{V}_{\mathcal{C}}$ *and let* $\mathcal{P}$ *be the UCQ query that is obtained from* $\mathcal{P}'$ *where each view subgoal* $V_i^{\mathcal{C}}$ *has been replaced by* $V_i \in \mathcal{V}$. *The following are equivalent*:

1. $\mathcal{P}'$ *is an MCR of Q with respect to UCQ using* $\mathcal{V}_{\mathcal{C}}$.
2. $\mathcal{P}$ *is an MCR of Q with respect to UCQ using* $\mathcal{V}$ *under* $\mathcal{C}$.

Proof. Let $\mathcal{P}'$ be a maximally contained rewriting of Q with respect to UCQ using $\mathcal{V}_{\mathcal{C}}$ and let $\mathcal{P}$ be the UCQ query that is obtained from $\mathcal{P}'$ where each view subgoal $V_i^{\mathcal{C}}$ has been replaced by $V_i \in \mathcal{V}$. By Lemma 4, since $\mathcal{P}'$ is a contained rewriting of Q, we have that $\mathcal{P}$ is also a contained rewriting of Q.

Suppose now that R is a contained CQ rewriting of Q using $\mathcal{V}$ under $\mathcal{C}$. According to Lemma 4, if we consider the CQ query R' that is obtained from R where each view subgoal V_i has been replaced by $V_i^{\mathcal{C}} \in \mathcal{V}_{\mathcal{C}}$, we have that $R' \sqsubseteq \mathcal{P}'$. Since R and R' compute the same answers for all databases D that satisfy $\mathcal{C}$, we have that $R \sqsubseteq \mathcal{P}'$. By Lemma 4, we also have that $R \sqsubseteq \mathcal{P}$, and thus $\mathcal{P}$ is an MCR of Q with respect to UCQ using $\mathcal{V}$ under $\mathcal{C}$. The other direction is symmetrical. □

Theorem 6. *Let Q be a UCQ,* $\mathcal{V}$ *be a set of views in the language of CQs and* $\mathcal{C} \subseteq \mathcal{C}_{\mathsf{LAV}}^{w}$ *be a set of tgds. Suppose there exists a maximally contained rewriting of Q with respect to UCQ using V in the presence of* $\mathcal{C}$. *Then,* MiniCon$\mathcal{C}$ *outputs a maximally contained rewriting of Q with respect to UCQ using V in the presence of* $\mathcal{C}$.

Proof. It is a straight consequence of Lemma 5 and Theorem 4.1 in [6], that if a UCQ MCR of Q wrt UCQ under $\mathcal{C}$ exists, the algorithm will also create such an MCR. □

5.2. MCRs vs. certain answers

The first observation about the connection of MCRs and certain answers appeared in [1], where Datalog is assumed as the language of rewriting. Here we first extend this result (Theorem 7) to capture any rewriting language. Then, in Theorem 8 we prove a weaker result like that of Theorem 7 in the presence of dependencies. It is worth noting that in the cases we investigate here the MCRs are essentially what are called *perfect* rewritings in [13]. This is shown first in Theorem 7 where if $\mathcal{P}$ is a maximally contained rewriting with respect to UCQ of a UCQ query Q, then $\mathcal{P}$ computes the certain answers of Q. Then, in Theorem 8 in Section 5.2.2, we show that if there exists a UCQ MCR $\mathcal{P}$ of a UCQ query Q in the presence of dependencies $\mathcal{C}$ such that the chase of Q with $\mathcal{C}$ always terminates, then $\mathcal{P}$ computes the certain answers of Q.

5.2.1. Absence of dependencies

We will now prove that a maximally contained rewriting $\mathcal{P}$ with respect to UCQ of a UCQ query Q computes the certain answers of Q.

Theorem 7. *Let Q be a UCQ query,* $\mathcal{V}$ *a set of CQ views and* $\mathcal{P}$ *an MCR of Q with respect to UCQ. Let* $\mathcal{I}$ *be a view instance such that there exists a database instance D such that* $\mathcal{I} \subseteq \mathcal{V}(D)$. *Then, under the open world assumption,* $\mathcal{P}$ *computes all the certain answers of Q in any view instance* $\mathcal{I}$: $\mathcal{P}(\mathcal{I}) = \text{certain}(Q, \mathcal{I})$.

In Theorem 7 we require that the view instance $\mathcal{I}$ is such that there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$. It turns out that this condition is not always necessary, and it can be satisfied when there is at least one tuple in the certain answers of Q with respect to $\mathcal{I}$.

Corollary 3. *Let Q be a UCQ query,* $\mathcal{V}$ *be a set of views and* $\mathcal{P}$ *be a MCR of Q wrt UCQ. If* $\mathcal{I}$ *is a view instance such that* $\text{certain}(Q, \mathcal{I}) \neq \emptyset$, *then* $\mathcal{P}(\mathcal{I}) = \text{certain}(Q, \mathcal{I})$.

Proof. Since $\text{certain}(Q, \mathcal{I}) \neq \emptyset$ and $\text{certain}(Q, \mathcal{I}) = \bigcap_{D \text{ s.t. } \mathcal{I} \subseteq \mathcal{V}(D)} Q(D)$, we have that there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$, hence the result follows by Theorem 7. □

The proof of Theorem 7 uses Lemma 6, which essentially states that in the absence of dependencies there is a finite space of contained rewritings which compute the certain answers of $(Q, \mathcal{I})$. This is in accordance with the results in [22,27], where the authors essentially prove that a contained CQ rewriting of Q which is part of a maximally contained rewriting with respect to UCQ can have at most as many subgoals as Q.

Lemma 6. *Let Q be CQ query and* $\mathcal{V}$ *be a set of CQ views. Let* $\mathcal{I}$ *be a view instance such that there exists a database instance D such that* $\mathcal{I} \subseteq \mathcal{V}(D)$. *Then there is a finite space* $\mathcal{S}$ *of contained rewritings (of Q using* $\mathcal{V}$*) of size which is a function only of the sizes of the views and the query, such that the following happens*: *Given a tuple* $t_0 \in \text{certain}(Q, \mathcal{I})$, *then there is a contained CQ rewriting R in* $\mathcal{S}$ *such that* $t_0 \in R(\mathcal{I})$.

Before proving this lemma, we give an example that goes along the lines of the proof and in particular we demonstrate how we create a CQ rewriting R given a tuple $t \in \text{certain}(Q, \mathcal{I})$ such that $t \in R(\mathcal{I})$.

Example 9. Let $Q(x, y) :- a(x, z, z')$, $b(z, y, z'')$ and $\mathcal{V} = \{V_1, V_2\}$, where $V_1(x, y) :- a(x, y, z)$, $V_2(x, y) :- b(x, y, z)$. Suppose $\mathcal{I} = \{v_1(1, 1), v_2(1, 2), v_1(1, 2), v_2(2, 3)\}$. We have that $\text{certain}(Q, \mathcal{I}) = \bigcap\{Q(D) : I \subseteq \mathcal{V}(D)\} = \{(1, 2), (1, 3)\}$. We

create the following sets: $A_{v_1(1,1)} = \{a(1,1,z_1)\}$, $A_{v_1(1,2)} = \{a(1,2,z_2)\}$, $A_{v_2(1,2)} = \{b(1,2,z_3)\}$, $A_{v_2(2,3)} = \{b(2,3,z_4)\}$. We then combine them and create $D_{\mathcal{I}} = \{a(1,1,z_1), a(1,2,z_2), b(1,2,z_3), b(2,3,z_4)\}$.

We have that $(1,3) \in \text{certain}(Q,\mathcal{I}) \Rightarrow (1,3) \in Q(D_{\mathcal{I}})$, hence there is a homomorphism $h : Var(Q) \rightarrow Const(D_{\mathcal{I}})$, where $h(x) = 1$, $h(y) = 3$, $h(z) = 2$, $h(z') = z_2$ and $h(z'') = z_4$. We have that $a(h(x), h(z), h(z')) \in A_{v_1(1,2)}$, $b(h(z), h(y), h(z'')) \in A_{v_2(2,3)}$, and we create the set $\mathcal{I}' = \{v_1(1,2), v_2(2,3)\}$. We assign variable x_1 to 1, x_2 to 2, x_3 to 3, and we create the CQ rewriting $R(x_1,x_3) : -V_1(x_1,x_2), V_2(x_2,x_3)$. If we consider the homomorphism $h' : Var(R) \rightarrow Const(\mathcal{I})$, where $h(x_1) = 1$, $h(x_2) = 2$ and $h(x_3) = 3$, then we can verify that $(1,3) \in R(\mathcal{I})$.

Proof of Lemma 6. First we point out an interesting case which is essentially a technicality that we need to consider for the lemma to be correct:

Example 10. Under the OWA, if $\mathcal{I} \not\subseteq \mathcal{V}(D)$ for all databases D, and R is a rewriting of a query Q using views with respect to $\mathcal{L}$, then there are cases where $\text{certain}(Q,\mathcal{I}) = \emptyset$ and $R(\mathcal{I}) \neq \emptyset$.

Consider the case where the query is $Q(x,y) : -a(x,y)$, we have only one view $v(x,x,y) : -a(x,y)$, and the view instance is $\mathcal{I} = \{v(1,2,3), v(4,4,5)\}$. Since $v(1,2,3) \in \mathcal{I}$ and $v(1,2,3) \notin \mathcal{V}(D)$ for any database D, we have that $\mathcal{I} \supset \mathcal{V}(D)$ for all databases D.

There is only one rewriting $R(x,y) : -V(x,x,y)$. Then $R(\mathcal{I}) = \{(4,5)\}$ and we have that

$$\text{certain}(Q,\mathcal{I}) = \bigcap_{D \text{ s.t. } \mathcal{I} \subseteq \mathcal{V}(D)} Q(D) = \emptyset$$

because $\nexists D$ such that $\mathcal{I} \subseteq \mathcal{V}(D)$.

For the proof of Lemma 6 we will use the following three lemmas:

Lemma 7. *Let $\mathcal{V}$ be a set of CQ views. Then the following are equivalent*:

1. *None of the view definitions in $\mathcal{V}$ has repeated variables in the head.*
2. *For all view instances $\mathcal{I}$ there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$.*

Proof. (1)$\Longrightarrow$(2). We define the following database instance:

Definition 12 (*Canonical database instance of $\mathcal{I}$*). Suppose that $t = v_i(a_1,\ldots,a_n)$ is a fact of $\mathcal{I}$. We define A_t to be the set that contains exactly the facts that are produced from the predicates in the body of the view definition of V_i in order to yield the fact $v_i(a_1,\ldots,a_n)$. To construct these facts, we unify t with the head variables of view V_i and we use fresh labelled nulls for the remaining (non-distinguished) variables that appear in the body of V_i. The members of the set A_t are exactly the frozen subgoals of view V_i after the unification. If the unification is not possible, we set $A_t = \emptyset$. We define a database instance $D_{\mathcal{I}}$ as follows: $D_{\mathcal{I}} = \bigcup_{t \in \mathcal{I}} A_t$. Database $D_{\mathcal{I}}$ is called the *canonical database instance of $\mathcal{I}$*.

We now consider the canonical database instance $D_{\mathcal{I}}$ of $\mathcal{I}$. Since the head of V_i does not contain repeated variables, during the construction of the sets A_t the unification step will never fail, and thus $A_t \neq \emptyset$, $\forall t \in \mathcal{I}$. We observe that by the construction of $D_{\mathcal{I}}$, it has the property that $\mathcal{I} \subseteq \mathcal{V}(D_{\mathcal{I}})$.

(2)$\Longrightarrow$(1). Suppose that for all view instances $\mathcal{I}$ there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$. For the sake of contradiction, let $V_0 \in \mathcal{V}$ such that V has repeated variables in the head. Without loss of generality, assume that the head of V_0 has two repeated variables and is defined as $V_0(x,x,x_3 \ldots, x_n)$.

Let I_0 be the view instance $I_0 = \{t\}$, where $t = v_0(c_1,c_2,c_3,\ldots,c_n)$, $c_1 \neq c_2$ and $c_1,c_2,c_3,\ldots,c_n \in Const(\mathcal{I})$. From our hypothesis we have that there is a database instance D_0 such that $\mathcal{I} \subseteq \mathcal{V}(D_0)$, so $t \in \mathcal{V}(D_0)$. Thus, there is a homomorphism $h : Var(V_0) \longrightarrow Const(\mathcal{V}(D))$ with the properties of Definition 4, thus $t = v_0(h(x), h(x), h(x_3), \ldots, h(x_n))$.

Since it also holds that $t = v_0(c_1,c_2,c_3,\ldots,c_n)$, we have that $c_1 = h(x) = c_2$, which is a contradiction because we assumed that $c_1 \neq c_2$. □

Lemma 8. *Let $\mathcal{V}$ be a set of CQ views and $\mathcal{I}$ a view instance such that there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$. Then, for any view instance $\mathcal{I}$, the canonical database instance $D_{\mathcal{I}}$ of $\mathcal{I}$ is non-empty and we have that $\mathcal{I} \subseteq \mathcal{V}(D_{\mathcal{I}})$.*

Proof. We consider the following cases:

- None of the view definitions in $\mathcal{V}$ has repeated variables in the head. Then, the result follows from the proof of Lemma 7.
- There are some views $V_1,\ldots,V_k \in \mathcal{V}$ that have repeated variables in the head. Let V be one of those views, and suppose that its head is $V(x_1,\ldots,x_n)$ and $x_{\sigma(1)},\ldots,x_{\sigma(k)}$ are the same variable, where σ is a function $\sigma : \{1,\ldots,k\} \rightarrow \{1,\ldots,n\}$ and $k \leq n$. Let $t = v(a_1,\ldots,a_n) \in \mathcal{I}$. If $a_{\sigma(i)} = a_{\sigma(j)}$ for $i \neq j$ and $i,j = 1,\ldots,k$, then we can create the set A_t because we can unify the t with the head of view V.
  Suppose now that for some $i,j \in \{1,\ldots,k\}$ and $i \neq j$ we have that $a_{\sigma(i)} \neq a_{\sigma(j)}$. Then, since there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$, we have that $t \in \mathcal{V}(D)$. So, there is a homomorphism $h : Var(V) \rightarrow Const(\mathcal{V}(D))$ such that:
  - h maps all the variables of V to constants in $\mathcal{V}(D)$ and all constants of V to the same constants in $V(D)$.
  - If $R_i(y_1,\ldots,y_n) \in Body(V)$ then $R_i(h(y_1),\ldots,h(y_n)) \in \mathcal{V}(D)$.
  - $t = v(h(x_1),\ldots,h(x_n))$.

  Since $x_{\sigma(i)}$, $x_{\sigma(j)}$ are the same variable, we have that $h(x_{\sigma(i)}) = h(x_{\sigma(j)})$. But $t = v(a_1,\ldots,a_n)$, and thus $a_{\sigma(i)} = h(x_{\sigma(i)}) = h(x_{\sigma(j)}) = a_{\sigma(j)}$, which is a contradiction because we assumed that $a_{\sigma(i)} \neq a_{\sigma(j)}$.

Thus, for all for $i,j = 1,\ldots,k$ and $i \neq j$ we have that $a_{\sigma(i)} = a_{\sigma(j)}$, so we can always unify a tuple $v(a_1,\ldots,a_n) \in \mathcal{I}$ with the head of view V, i.e. we can always create $D_{\mathcal{I}}$ such that $D_{\mathcal{I}} \neq \emptyset$. □

Lemma 9. *Let Q be CQ query, $R(y_1,\ldots,y_m) \in Body(Q)$ and $\mathcal{V}$ be a set of CQ views. Let $\mathcal{I}$ be a view instance such that the canonical instance $D_{\mathcal{I}}$ is non-empty. Suppose that $t \in \text{certain}(Q,\mathcal{I})$, $t = (a_1,\ldots,a_n)$ and let h be the homomorphism $h : Var(Q) \to Const(D_{\mathcal{I}})$ that computes tuple t. The following holds*:

If $R(h(y_1),\ldots,h(y_m)) \in A_{t_1}$ for some fact $t_1 \in \mathcal{I}$ (where A_{t_1} is defined as in Definition 12) and y_j is a distinguished variable in the definition of Q, (which means that $a_i = h(y_j)$ for some $i \in \{1,\ldots,n\}$ and $j \in \{1,\ldots,m\}$), then constant $h(y_j)$ also appears in the fact t_1.

Proof. Suppose that $Head(Q) = \{x_1,\ldots,x_n\}$ and $t \in \text{certain}(Q,\mathcal{I})$. We have that $\text{certain}(Q,\mathcal{I}) = \bigcap_{D\in\mathcal{D}} Q(D)$ where $\mathcal{D} = \{D : \mathcal{I} \subseteq \mathcal{V}(D)\}$. Thus, if $t \in \text{certain}(Q,\mathcal{I})$, then $t \in Q(D)$, $\forall D \in \mathcal{D}$, which means that there exists a homomorphism $h_D : Var(Q) \to Const(D)$ such that

1. h_D maps all the variables of Q to constants in $Const(D)$ and all constants of Q to the same constants in $Const(D)$.
2. If $R_i(y_1,\ldots,y_m) \in Body(Q)$ then $R_i(h_D(y_1),\ldots,h_D(y_m))$ is a fact of D, i.e. $R_i(h_D(y_1),\ldots,h_D(y_m)) \in D$.
3. If $Head(Q) = (x_1,\ldots,x_k)$, then $t = (h_D(x_1),\ldots,h_D(x_k)) \in Q(D)$, that is $(a_1,\ldots,a_n) = (h_D(x_1),\ldots,h_D(x_k))$.

Suppose that x_1 is a distinguished variable that appears in subgoal $R(y_1,\ldots,y_m)$ in the body of Q. Without loss of generality, we assume that x_1 appears in the first position of R, i.e. we can write subgoal $R(y_1,\ldots,y_m)$ as $R(x_1,y_2,\ldots,y_m)$.

Hence, since t is a certain answer and $h(x_1) = a_1$, we have that all databases $D \in \mathcal{D}$ must contain a subgoal of the form $R(a_1,c_{(D,2)},\ldots,c_{(D,m)})$, where $c_{(D,2)},\ldots,c_{(D,m)}$ are constants that may belong only in D. Suppose now that h is the homomorphism $h : Var(Q) \to Const(D_{\mathcal{I}})$ that computes tuple t and that $R(h(y_1),\ldots,h(y_m)) \in A_{t_1}$ for some fact $t_1 = v_j(b_1,\ldots,b_k) \in \mathcal{I}$. We will prove that $a_1 = b_l$ for some $l \in \{1,\ldots,k\}$. We consider the following two cases:

- In the view definition of view V_j, the variable y_1 of $R(y_1,\ldots,y_m)$ is a distinguished variable. Then, since $R(h(y_1),\ldots,h(y_m)) \in A_{t_1}$, it is obvious that $a_1 = b_l$ for some $l \in \{1,\ldots,k\}$.
- The variable y_1 of R to which a_1 is assigned is a non-distinguished variable in the view definition of view V_j. We construct the canonical database instance $D_{\mathcal{I}}$ of $\mathcal{I}$, and we define D_0 to be the database instance that is derived from $D_{\mathcal{I}}$ where each labelled null has been replaced by a constant that is different from a_1. We observe that by the construction of $D_{\mathcal{I}}$, variable y_1 of R will be assigned a labelled null in $D_{\mathcal{I}}$ (since it is a non-distinguished variable). Thus, by the construction of D_0, y_1 will be assigned a constant that is different from a_1. We also have that D_0 has the property that $\mathcal{I} \subseteq \mathcal{V}(D_0)$. This is a contradiction, because by construction D_0 does not contain any subgoal of the form $R(a_1,c_{(D_0,2)},\ldots,c_{(D_0,m)})$, where $c_{(D_0,2)},\ldots,c_{(D_0,m)} \in const(D_0)$.
  Hence, the variable y_1 of R to which a_1 is assigned cannot be a non-distinguished variable in the view definition of view V_j, which means that it is a distinguished variable, and by the previous case $a_1 = b_l$ for some $l \in \{1,\ldots,k\}$. □

We now continue with the proof of Lemma 6: we define $\mathcal{S}$ to be the set of contained CQ rewritings of Q using $\mathcal{V}$ such that R has at most s subgoals, and where the different subgoals that form each one of these rewritings are the views from the set $\mathcal{V}$. Hence, the size of $\mathcal{S}$ is a function only of the sizes of the view definitions $\mathcal{V}$ and the query Q.

By Lemma 8 we have that the canonical database $D_{\mathcal{I}}$ of $\mathcal{I}$ is non-empty and that $\mathcal{I} \subseteq \mathcal{V}(D_{\mathcal{I}})$. Let $t_0 \in \text{certain}(Q,\mathcal{I})$. Since $\text{certain}(Q,\mathcal{I}) = \bigcap_{\mathcal{I}\subseteq\mathcal{V}(D)} Q(D)$, we have that $t_0 \in Q(D_{\mathcal{I}})$. This means that there is a homomorphism $h : Var(Q) \to Const(D_{\mathcal{I}})$ with the properties of Definition 4. Suppose now that Q has s subgoals and that $Q(\vec{x}) : -R_1(\vec{x_1}),\ R_2(\vec{x_2}),\ldots,\ R_s(\vec{x_s})$ where the variables $\vec{x}$ appear in the set of all variables $\{\vec{x_1},\ldots,\vec{x_s}\}$.

Let $D'_{\mathcal{I}} = \{R(h(\vec{x})), R(\vec{x}) \in Body(Q)\}$, $D'_{\mathcal{I}} \subseteq D_{\mathcal{I}}$. By the definition of $D_{\mathcal{I}}$, if $a \in D_{\mathcal{I}}$ then there exists a $t \in \mathcal{I}$ such that $a \in A_t$. We define I' as follows:

$$I' = \{t \in \mathcal{I} : \exists a \in D'_{\mathcal{I}} \text{ such that } a \in A_t\}$$

We have that $|I'| \leq s$ and for all $a \in D'_{\mathcal{I}}$. In addition, if a distinguished variable x_i appears in a subgoal $R_j(y_1,\ldots,y_m)$ of Q, then, since $h(x_i)$ is a constant that appears in t_0 and $t_0 \in \text{certain}(Q,\mathcal{I})$, by Lemma 9 we have that for all $t \in \mathcal{I}'$ such that $R_j(y_1,\ldots,y_m) \in A_t$, $h(x_i)$ will appear among the constants of fact t. Thus, if $C = Const(\mathcal{I}')$, then all the constants that appear in t_0 are members of the set C.

Suppose without loss of generality that $\mathcal{I}' = \{v_1(\vec{c_1}),\ldots,\ v_{s'}(\vec{c_{s'}})\}$, where $s' \leq s$. We assign a distinct variable x_c to each distinct constant $c \in C$, and create the query

$$R(x_1,\ldots,x_n) : -V_1(\vec{x_1}),\ldots,V_{s'}(\vec{x_{s'}})$$

Note that the query is safe, because we have already shown that all the constants in the result tuple t_0 appear in the set of constants C, and thus the variables that we will assign to them will also appear in the body of R. Let $Var(R)$ be the set of all variables of R and $g : C \to Var(R)$ be the mapping that assigns a variable from $Var(R)$ to each constant in C, i.e. $g(c) = x_c$. We have that the following two hold:

- $R^{exp} \sqsubseteq Q$.
  This holds because if we consider the homomorphism $h_1 : Var(Q) \to Var(R^{exp})$, $h_1 = g \circ h$, we have that h_1 maps each subgoal of Q to a subgoal of R^{exp} and h_1 also maps the head of Q to the head of R^{exp}.
- $t_0 \in R(\mathcal{I})$.
  We consider the homomorphism $g^{-1} : Var(R) \to C$. Because $C \subseteq Const(\mathcal{I})$, we can write $g^{-1} : Var(R) \to Const(\mathcal{I})$. We have that g^{-1} maps all the subgoals of the body of

R to facts in $\mathcal{I}$, and by the construction of query R we have that $(g(x_1),\ldots,g(x_n)) = (h(x_1),\ldots,\ h(x_n)) = t_0$, so $t_0 \in R(\mathcal{I})$.

Thus, there exists a contained rewriting R of Q using $\mathcal{V}$ such that R is a conjunctive query and that $t_0 \in R(\mathcal{I})$. □

Before proving Theorem 7, we will also need the following proposition:

Proposition 3. *Let Q be a CQ query, $\mathcal{V}$ be a set of CQ views, R be a CQ contained rewriting of Q and $\mathcal{I}$ a view instance such that there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$. Then $R(\mathcal{I}) \subseteq$ certain$(Q,\mathcal{I})$.*

Proof. We will prove the above under the Open and Closed World Assumptions.

- Under the OWA, because R is a contained rewriting, we have that for all database instances D such that $\mathcal{I} \subseteq \mathcal{V}(D)$: $R^{exp}(D) \subseteq Q(D)$. But $R(\mathcal{I}) \subseteq R(\mathcal{V}(D))$ and $R(\mathcal{V}(D)) = R^{exp}(D)$, so $R(\mathcal{I}) \subseteq Q(D)$. Thus:

$$R(\mathcal{I}) \subseteq \bigcap_{\mathcal{I} \subseteq \mathcal{V}(D)} Q(D) \Longrightarrow R(\mathcal{I}) \subseteq \text{certain}(Q,\mathcal{I})$$

- Under the CWA, because R is a contained rewriting, we have that for all database instances D such that $\mathcal{I} = \mathcal{V}(D)$: $R^{exp}(D) \subseteq Q(D)$. But $R(\mathcal{I}) = R(\mathcal{V}(D))$ and $R(\mathcal{V}(D)) = R^{exp}(D)$, so $R(\mathcal{I}) \subseteq Q(D)$. Thus:

$$R(\mathcal{I}) \subseteq \bigcap_{\mathcal{I} = \mathcal{V}(D)} Q(D) \Longrightarrow R(\mathcal{I}) \subseteq \text{certain}(Q,\mathcal{I})$$

In any case, $R(\mathcal{I}) \subseteq$ certain$(Q,\mathcal{I})$. □

We now give the proof of Theorem 7:

Proof of Theorem 7. We will show the following:

1. $\mathcal{P}(\mathcal{I}) \subseteq$ certain$(Q,\mathcal{I})$.
2. certain$(Q,\mathcal{I}) \subseteq \mathcal{P}(\mathcal{I})$.

Since $\mathcal{P}$ is a contained rewriting of Q, the first is a direct consequence of Proposition 3. To prove the second, we first consider the case where $\mathcal{I} = \emptyset$. Then, $\mathcal{P}(\mathcal{I}) = \emptyset$ and we have that certain$(Q,\mathcal{I}) = \bigcap_{D \text{ s.t. } \mathcal{I} \subseteq \mathcal{V}(D)} Q(D) = \bigcap_{D \text{ s.t.} \emptyset \subseteq \mathcal{V}(D)} Q(D) = \bigcap_{\text{for all } D} Q(D) = \emptyset$. Thus, certain$(Q,\mathcal{I}) = \emptyset \subseteq \emptyset = \mathcal{P}(\mathcal{I})$. Suppose now that $\mathcal{I} \neq \emptyset$. We consider the following two cases:

1. There is a contained rewriting $R \in \mathcal{S}$ such that $R(\mathcal{I}) \not\subseteq \mathcal{P}(\mathcal{I})$. But that is a contradiction because R is a contained rewriting of Q and we assumed that $\mathcal{P}$ is an MCR of Q using $\mathcal{V}$ with respect to UCQ.
2. There is no contained rewriting $R \in \mathcal{S}$ such that $R(\mathcal{I}) \not\subseteq \mathcal{P}(\mathcal{I})$. Thus, we have that $\forall R \in \mathcal{S} : R(\mathcal{I}) \subseteq \mathcal{P}(\mathcal{I})$. Suppose that $t_0 \in$ certain$(Q,\mathcal{I})$. By Lemma 6, there is a rewriting in $\mathcal{S}$ which computes t_0, i.e. $t_0 \in R(\mathcal{I})$. Hence, $t_0 \in P(\mathcal{I})$. □

5.2.2. Presence of dependencies $\mathcal{C}$

In the presence of a set of dependencies $\mathcal{C}$, we need to state and prove a weaker lemma than Lemma 6 to reflect the changes introduced by $\mathcal{C}$. This is necessary due to the fact that in this case, a UCQ MCR may not exist in general, as we already argued in the introduction (Example 2), and we give the proof below:

Proof. We revisit Example 2 from the introduction. In this example, there is no maximally contained rewriting of Q with respect to UCQ. The intuition behind this is that every contained rewriting Q_n returns all pilots that work in the same airline as Mike such that each pair of pilots have flown the same aircraft at some point, up to a maximum of n aircrafts. Thus, if we consider $\mathcal{P}$ to be an MCR of Q with respect to UCQ, then, if the maximum number of subgoals of each rewriting R_i in the conjunction is m, $\mathcal{P}$ will not be able to return the pilots that work in the same airline as Mike such that each pair of pilots have flown the same aircraft at some point, but up to a maximum of $n > m$ aircrafts.

To show this formally, we first consider the contained rewriting Q_{m+1}:

$$\begin{aligned} Q_{m+1}(P) : -V(D_1, mike, C_1), V(D_2, P_2, C_1),\\ V(D_3, P_2, C_2), V(D_4, P_3, C_2), V(D_5, P_3, C_3),\\ V(D_6, P_4, C_3), \ldots, V(D_{2m}, P_{m+1}, C_m),\\ V(D_{2m+1}, P_{m+1}, C_{m+1}), V(D_{2m+2}, P_{m+2}, C_{m+1}) \end{aligned}$$

Then, we construct a view instance V_{m+1} by freezing the subgoals in the body of Q_{m+1} to constants as it is shown below:

$$\begin{aligned} V_{m+1} = \{&V(d_1, mike, c_1), V(d_2, p_2, c_1),\\ &V(d_3, p_2, c_2), V(d_4, p_3, c_2), V(d_5, p_3, c_3)\\ &V(d_6, p_4, c_3), \ldots, V(d_{2m}, p_{m+1}, c_m),\\ &V(d_{2m+1}, p_{m+1}, c_{m+1}), V(d_{2m+2}, p_{m+2}, c_{m+1})\} \end{aligned}$$

It is easy to see that $Q_{m+1}(V_{m+1}) = \{mike, p_2, \ldots, p_{m+2}\}$. Since Q_{m+1} is contained in $\mathcal{P}$ there exists a rewriting $R \in \mathcal{P}$ such that $Q_{m+1} \sqsubseteq R$, and thus $p_{m+2} \in R(V_{m+1})$. If R has s subgoals, then $s \leq m$, which means that R uses at most $s \leq m$ facts from V_{m+1} to return the answer $\{p_{m+2}\}$.

By Definition 4, if P is the (only) variable in the head of R, there is a homomorphism $h : Var(R) \to Const(V_{m+1})$ such that $h(P) = p_{m+2}$ and if $R_i(\vec{y})$ is a subgoal in the body of R, then $R_i(h(\vec{y})) \in V_{m+1}$. We now consider a view instance $V_s = \{R_i(h(\vec{y})) : R_i(\vec{y}) \in Body(R)\}$ (V_s contains only the s tuples from V_{m+1} used to produce the answer $\{p_{m+2}\}$, i.e. $R(V_s) = \{p_{m+2}\}$). Since V_{m+1} contains $2m+2 > m \geq s$ subgoals, at least one tuple that was in V_{m+1} is not present in V_s. Let $V(d', p_i, c')$, $2 \leq i \leq m+2$ be that tuple. We now construct a database D' from V_s by replacing the tuples in V_s with their expansi ons, as it is shown here: $D' = \{schedule(a_1, n_1, d_1, mike, c_1)$, $schedule(a_2, n_2, d_2, p_2, c_1)$, $schedule(a_2, n_2, d_3, p_2, c_2), \ldots, schedule(a_{2i-2}, n_{2i-2}, d_{2i-2}, p_{i-1}, c_{i-1})$, $schedule(a_{2i+2}, n_{2i+2}, d_{2i+2}, p_{i+1}, c_{i+1}), \ldots, schedule(a_{2m}, n_{2m}, d_{2m}, p_{m+1}, c_m)$, $schedule(a_{2m+1}, n_{2m+1}, d_{2m+1}, p_{m+1}, c_{m+1})$, $schedule(a_{2m+2}, n_{2m+2}, d_{2m+2}, p_{m+2}, c_{m+1})\}$.

Since D' must satisfy the two functional dependencies, we have that $a_1 = a_2 = \cdots = a_{i-1}$ and $a_{i+1} = a_{i+2} = \cdots = a_{m+2}$, but not necessarily that $a_{i-1} = a_{i+1}$. Thus, it is clear that $p_{m+2} \notin Q(D')$, which is a contradiction, because R is a contained rewriting of Q and $R(V_s) = \{p_{m+2}\}$. □

We now state the lemma that we will use to prove the main result of this section, which is that a UCQ MCR $\mathcal{P}$ of a UCQ query Q in the presence of dependencies $\mathcal{C}$ (if it exists) computes the certain answers of Q.

Lemma 10. *Let Q be CQ query, $\mathcal{V}$ be a set of CQ views, $\mathcal{C}$ be a set of dependencies such that the chase of Q with $\mathcal{C}$ terminates and $\mathcal{P}$ be an UCQ MCR of Q wrt UCQ. Let $\mathcal{I}$ be a view instance such that there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$ and $D \models \mathcal{C}$. Then there is a finite space $\mathcal{S}'$ of contained rewritings (of Q using $\mathcal{V}$) of size which is a function of the sizes of $\mathcal{P}$ and $\mathcal{V}$, such that the following happens: given a tuple $t_0 \in \text{certain}(Q, \mathcal{I})$, then there is a contained CQ rewriting R in $\mathcal{S}'$ such that $t_0 \in R(\mathcal{I})$.*

Before proving Lemma 10, we give another example that goes along the lines of the proof and that demonstrates how we can create a CQ rewriting R of Q under $\mathcal{C}$ when we are given a tuple $t \in \text{certain}(Q, \mathcal{I})$, such that $t \in R(\mathcal{I})$.

Example 11. Suppose we have the CQ query $Q(x) : -a(x,x), b(x,y), c(x,y)$, the set of views $\mathcal{V} = \{V_1, V_2, V_3\}$, where $V_1(x) : -a(x,y)$, $V_2(x) : -b(x,y)$ and $V_3(x) : -c(x,x)$, and the set $\mathcal{C} = \{d_1, d_2, d_3\}$, where $d_1 : a(x,y) \wedge b(x,y') \rightarrow x = y$, $d_2 : a(x,y) \rightarrow d(x)$ and $d_3 : d(x) \wedge c(x,y) \rightarrow x = y$. Let $\mathcal{I}$ be the view instance $\mathcal{I} = \{v_1(1), v_2(1), v_3(1)\}$. By definition, we have that $\text{certain}(Q, \mathcal{I}) = \bigcap \{Q(D) : \mathcal{I} \subseteq \mathcal{V}(D), D \models \mathcal{C}\}$. If D is a database such that $\mathcal{I} \subseteq \mathcal{V}(D)$, $D \models \mathcal{C}$ and $k \in \text{certain}(Q, \mathcal{I})$, then $k \in Q(D)$ and also $a(k,k), b(k,n_1), c(k,k) \in D$, where n_1 is some constant. Thus, $\mathcal{V}(D) = \{v_1(k), v_2(k), v_3(k)\}$, and since $\mathcal{I} \subseteq \mathcal{V}(D)$, we have that $k = 1$. Hence, certain $(Q, \mathcal{I}) = \{1\}$.

We now create the set $D^{\mathcal{C}}_{\mathcal{I}}$. First we create the sets $A_{v_1(1)} = \{a(1,n_1)\}$, $A_{v_2(1)} = \{b(1,n_2\}$ and $A_{v_3(1)} = \{c(1,1)\}$, as in Example 11, and the set $D_{\mathcal{I}} = \{a(1,n_1), b(1,n_2), c(1,1)\}$ (note that $D_{\mathcal{I}}$ does not satisfy $\mathcal{C}$ since $d(1) \notin D_{\mathcal{I}}$). We chase $D_{\mathcal{I}}$ with $\mathcal{C}$ and get $D^{\mathcal{C}}_{\mathcal{I}} = \{a(1,1), b(1,n_2), c(1,1), d(1)\}$. We create the set $\text{CHASE}_{[a(1,1)] = \{a(1,n_1), b(1,n_2)\}}$ for the tuple $a(1,1)$ and the set $\text{CHASE}_{[d(1)] = \{a(1,1)\}}$ for the introduced tuple $d(1)$. We also chase Q with $\mathcal{C}$ and we get $Q_{\mathcal{C}}(x) : -a(x,x), b(x,y), c(x,x), d(x)$.

We will now create the contained CQ rewriting as follows: for the database $D^{\mathcal{C}}_{\mathcal{I}}$ we have that $\mathcal{I} \subseteq \mathcal{V}(D^{\mathcal{C}}_{\mathcal{I}})$ and $D^{\mathcal{C}}_{\mathcal{I}} \models \mathcal{C}$, so $1 \in \text{certain}(Q, \mathcal{I}) \Longrightarrow 1 \in Q(D^{\mathcal{C}}_{\mathcal{I}}) \Longrightarrow 1 \in Q_{\mathcal{C}}(D_{\mathcal{I}})$ and thus we have that there exists a homomorphism $h : Var(Q_{\mathcal{C}}) \rightarrow Const(D_{\mathcal{I}})$, $h(x) = 1$ and $h(y) = n_2$. Next, we create the set I that contains all the ground view atoms that cover the subgoals of $Q_{\mathcal{C}}$ for the certain answer 1. Tuple $a(h(x), h(x)) = a(1,1)$ is introduced from a chase step, so we add to I all tuples $t \in \mathcal{I}$ with the property that there exists a $t' \in \text{CHASE}_{[a(1,1)]}$ such that $t' \in A_t$, thus, $I = \{v(1), v(2)\}$. Since $b(h(x), h(y)) = b(1,n_2) \in A_{v(2)}$ and $c(h(x), h(x)) = c(1,1) \in A_{v_3(1)}$ so we add $v_2(1)$, $v_3(1)$ to I. Tuple $d(h(x)) = d(1)$ is introduced from a chase step and we proceed as before, but since the tuple $a(1,1) \in \text{CHASE}_{[d(1)]}$ is taken care of in the first bullet we do not need to add any more tuples in the set I. Thus $I = \{v_1(1), v_2(1), v_3(1)\}$. We assign variable x_1 to 1 and we create the contained rewriting $R(x_1) : -V_1(x_1), V_2(x_1), V_3(x_1)$. If we now consider the homomorphism $h' : Var(R) \rightarrow Const(\mathcal{I})$ where $h(x_1) = 1$, wecan verify that $1 \in R(\mathcal{I})$.

We also observe that the canonical database instance $D_{\mathcal{I}}$ may not always satisfy $\mathcal{C}$, which is shown in the following example:

Example 12. Suppose that the set of views $\mathcal{V}$ contains the following two views $V_1(x) : -a(x,y)$, $V_2(x) : -b(x,y)$, $\mathcal{C}$ contains the egd $d : a(x,y) \wedge b(x,y') \rightarrow x = y$ and that the view instance is $\mathcal{I} = \{v_1(1), v_2(1)\}$. Then, we have that $D_{\mathcal{I}} = \{a(1,y), b(1,y')\}$, where y and y' need not necessarily be the same constant. In the case where $y \neq y'$, it is clear that $D_{\mathcal{I}} \not\models \mathcal{C}$.

However, if the chase of $D_{\mathcal{I}}$ with $\mathcal{C}$ does not fail and results in the instance $D^{\mathcal{C}}_{\mathcal{I}}$, then $D^{\mathcal{C}}_{\mathcal{I}}$ is guaranteed to satisfy the constraints $\mathcal{C}$. It turns out that this holds if there exists a database D such that $\mathcal{I} \subseteq \mathcal{V}(D)$ and $D \models \mathcal{C}$.

Proposition 4. *Let $\mathcal{V}$ be a set of CQ views, $\mathcal{C}$ a set of dependencies such that the chase of $D_{\mathcal{I}}$ with $\mathcal{C}$ terminates and $\mathcal{I}$ a view instance such that there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$ and $D \models \mathcal{C}$. Then the following hold:*

1. *The canonical database instance $D_{\mathcal{I}}$ of $\mathcal{I}$ is non-empty.*
2. *The chase of $D_{\mathcal{I}}$ with $\mathcal{C}$ does not fail.*
3. *If the chase of $D_{\mathcal{I}}$ with $\mathcal{C}$ results in the instance $D^{\mathcal{C}}_{\mathcal{I}}$, then $\mathcal{I} \subseteq_{\mathcal{C}} \mathcal{V}(D^{\mathcal{C}}_{\mathcal{I}})$.*

Proof. In the proof we will use out a property of chasing instances:

Lemma 11. *Let D_1, D_2 be database instances such that $D_1 \subseteq D_2$ and the chase of D_1 with $\mathcal{C}$ fails. Then the chase of D_2 with $\mathcal{C}$ also fails and thus $D_2 \not\models \mathcal{C}$.*

We have that (1) holds because of Lemma 8 and (3) is a direct consequence of Lemma 8 and the fact that $D_{\mathcal{I}} \equiv_{\mathcal{C}} D^{\mathcal{C}}_{\mathcal{I}}$. We will now prove (2) by contradiction.

Suppose that the chase of $D_{\mathcal{I}}$ with $\mathcal{C}$ fails. We will show that for every database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$, there is a function $L : \mathcal{L}(D_{\mathcal{I}}) \rightarrow Const(D)$ (where $\mathcal{L}(D_{\mathcal{I}})$ is the set of labelled nulls of $D_{\mathcal{I}}$) such that $L(D_{\mathcal{I}}) \subseteq D$. Thus, if there is a database D such that $\mathcal{I} \subseteq \mathcal{V}(D)$ and $D \models \mathcal{C}$, we would then have that $L(D_{\mathcal{I}}) \subseteq D$. Since a chase fails only when at a chase step we need to equate two distinct variables (not labelled nulls) and the chase of $D_{\mathcal{I}}$ with $\mathcal{C}$ fails, it is clear that the chase of $L(D_{\mathcal{I}})$ with $\mathcal{C}$ will also fail. Thus by taking into account Lemma 11, we have that $D \not\models \mathcal{C}$, which is a contradiction.

We now show the existence of the function L. Let $t = v(c_1, \ldots, c_n) \in \mathcal{I}$. Since $\mathcal{I} \subseteq \mathcal{V}(D)$, we have that $t \in \mathcal{V}(D)$, thus there is a homomorphism $h_t : Var(V) \rightarrow Const(D)$ with the properties of Definition 4. Note that there may be more that one such homomorphisms, i.e. there is a set H_t of such

homomorphisms and $h_t \in H_t$. We also have that $t \in \mathcal{V}(D_\mathcal{I})$ (by the construction of $D_\mathcal{I}$), and thus there is a homomorphism $h'_t : Var(V) \to Const(D_\mathcal{I})$ with the properties of Definition 4. If x is a non-distinguished variable, then by the construction of $D_\mathcal{I}$ we have that $h'_t(x) = n_{t,x}$ where $n_{t,x}$ is a labelled null that appears only in the set A_t. We define $L(n_{t,x}) = h_t(x)$, and if we do this for all tuples $t \in \mathcal{I}$ we will create a function $L : \mathcal{L}(D_\mathcal{I}) \to Const(D)$. Since each labelled null appears in exactly one of the sets A_t, we have that L is well-defined, and thus $L(D_\mathcal{I}) \subseteq D$. □

Proof of Lemma 10. We chase Q with $\mathcal{C}$ and we create $Q_\mathcal{C}$. Suppose that $\mathcal{P} = \bigcup_{i=1}^{m} P_i$ and that s is the maximum number of subgoals that appear in each P_i, i.e. $s = \max|P_i|$. We define $\mathcal{S}'$ to be the set of contained CQ rewritings of Q using $\mathcal{V}$ under $\mathcal{C}$ such that R has at most s subgoals, and where the different subgoals that form each one of these rewritings are the views from the set $\mathcal{V}$. Hence, the size of $\mathcal{S}'$ is a function of the sizes of $\mathcal{P}$ and $\mathcal{V}$.

By Lemma 4 we have that the canonical database $D_\mathcal{I}$ of $\mathcal{I}$ is non-empty and that $\mathcal{I} \subseteq_\mathcal{C} \mathcal{V}(D_\mathcal{I})$. We chase $D_\mathcal{I}$ with $\mathcal{C}$ to obtain $D^\mathcal{C}_\mathcal{I}$, and for each $b \in D^\mathcal{C}_\mathcal{I} - D_\mathcal{I}$ we create a set CHASE[b] that contains exactly all the tuples from $\mathcal{I}$ that appear in the chase steps of b. Let $t_0 \in$ certain$(Q, \mathcal{I})$. Since certain$(Q, \mathcal{I}) = \bigcap_{\mathcal{I} \subseteq_\mathcal{C} \mathcal{V}(D)} Q(D)$, we have that $t_0 \in Q(D^\mathcal{C}_\mathcal{I})$, and thus $t_0 \in Q_\mathcal{C}(D^\mathcal{C}_\mathcal{I})$. This means that there is a homomorphism $h : Var(Q_\mathcal{C}) \to Const(D^\mathcal{C}_\mathcal{I})$ with the properties of Definition 4. Suppose now that $Q_\mathcal{C}(\vec{x}) : -R_1(\vec{x_1}), \ldots, R_{s_1}(\vec{x_{s_1}})$, where variables $\vec{x}$ appear in the set of all variables $\{\vec{x_1}, \ldots, \vec{x_{s_1}}\}$.

Let $D'^\mathcal{C}_\mathcal{I} = \{R(h(\vec{x})), R(\vec{x}) \in Body(Q_\mathcal{C})\}$, $D'^\mathcal{C}_\mathcal{I} \subseteq D^\mathcal{C}_\mathcal{I}$. We will now create a set I that contains only tuples from $\mathcal{I}$, and from the set I we will create the contained CQ rewriting R for t_0. By the definition of $D^\mathcal{C}_\mathcal{I}$, if $a \in D^\mathcal{C}_\mathcal{I}$ then there are three cases:

- There exists a $t \in \mathcal{I}$ such that $a \in A_t$. In this case we add t to I.
- a is introduced in some chase step when chasing $D_\mathcal{I}$ with $\mathcal{C}$, and the set CHASE[a] contains only tuples from $\mathcal{I}$. In this case, we add to I all tuples $t \in \mathcal{I}$ with the property that there exists a $t' \in$ CHASE[a] such that $t' \in A_t$.
- a is introduced in some chase step when chasing $D_\mathcal{I}$ with $\mathcal{C}$, and the set CHASE[a] also contains some tuples $a_1, \ldots, a_n$ that were introduced from a chase step. We act as in the previous case and we add to I all the tuples of $\mathcal{I}$ that appear in the chase steps of $a, a_1, \ldots, a_n$.

Hence, we have that $I' \subseteq \mathcal{I}$. Suppose without loss of generality that $I' = \{v_i(\vec{c_i}), i \in N\}$, where $N \subseteq \mathbb{N}$ and $\vec{c_i}$ are vectors of constants. We assign a distinct variable x_c to each distinct constant $c \in C$, and create the query

$$R(x_1, \ldots, x_n) : -\bigwedge_{i \in N} V_i(\vec{x_i})$$

Without loss of generality, we assume that $x_1, \ldots, x_n$ are the variables that we assign to the constants $c_1, \ldots, c_n$ that form the certain answer $t_0 = (c_1, \ldots, c_n)$. We need to show that the query R is safe, and for this to hold every constant that appears in the certain answer t_0 must appear in some view fact in I'. Let d be a constant that appears in t_0. Then one of the following must hold:

- d appears in some fact $R(h(\vec{x}))$ (where $R \in Body(Q_\mathcal{C})$) and $R(h(\vec{x})) \in A_t$ for some $t \in \mathcal{I}$, then from Lemma 9 we know that constant d appears in the view fact t.
- d appears in some fact $R(h(\vec{x}))$ and $R(h(\vec{x})) \notin A_t$, $\forall t \in \mathcal{I}$. Then, $R(h(\vec{x})) \in D^\mathcal{C}_\mathcal{I} - D_\mathcal{I}$, i.e. the fact $R(h(\vec{x}))$ was created from a chase step. In this case we have included in I all the view facts of $\mathcal{I}$ that appear in the chase sequence of $R(h(\vec{x}))$, thus we need to show that d appears in at least one of these view facts. This is indeed true, because if the opposite holds, then we would have that d is equal to some labelled null in $D^\mathcal{C}_\mathcal{I}$, but since a labelled null can have any value, this is a contradiction (the detailed proof is similar to Lemma 9).

Suppose now that $Var(R)$ is the set of all variables of R and $g : C \to Var(R)$ be the mapping that assigns a variable from $Var(R)$ to each constant in C, i.e. $g(c) = x_c$. We have that the following two hold:

- $R^{exp}_\mathcal{C} \sqsubseteq Q$.
  This holds because if we consider the homomorphism $h_1 : Var(Q) \to Var(R^{exp}_\mathcal{C})$, $h_1 = g \circ h$, we have that h_1 maps each subgoal of Q to a subgoal of $R^{exp}_\mathcal{C}$ and h_1 also maps the head of Q to the head of $R^{exp}_\mathcal{C}$.
- $t_0 \in R(\mathcal{I})$.
  We consider the homomorphism $g^{-1} : Var(R) \to C$. Because $C \subseteq Const(\mathcal{I})$, we can write $g^{-1} : Var(R) \to Const(\mathcal{I})$. We have that g^{-1} maps all the subgoals of the body of R to facts in $\mathcal{I}$, and by the construction of query R we have that $(g(x_1), \ldots, g(x_n)) = (h(x_1), \ldots, h(x_n)) = t_0$, so $t_0 \in R(\mathcal{I})$.

Thus, there exists a contained rewriting R of Q using $\mathcal{V}$ such that $t_0 \in R(\mathcal{I})$ (note that the conjunction that constructs R may not be finite). Since $R \sqsubseteq Q$ and $\mathcal{P} = \cup_{i=1}^{m} P_i$ is an MCR of Q wrt UCQ, we have that $R \sqsubseteq \mathcal{P}$. Hence, $t_0 \in R(\mathcal{I}) \Rightarrow t_0 \in P_j(\mathcal{I})$, for some $j \in \{1, \ldots, m\}$, and so we have that P_j is a contained CQ rewriting of Q (by the definition of $\mathcal{P}$) with a number of subgoals which is at most s, and $t_0 \in P_j(\mathcal{I})$. □

We will now prove that a UCQ MCR $\mathcal{P}$ of a UCQ query Q in the presence of dependencies $\mathcal{C}$ (if it exists) computes the certain answers of Q.

Theorem 8. *Let Q be a UCQ query, $\mathcal{V}$ be a set of CQ views, $\mathcal{C}$ be a set of dependencies such that the chase of Q with $\mathcal{C}$ terminates and $\mathcal{P}$ be a MCR of Q under $\mathcal{C}$ with respect to UCQ. Let $\mathcal{I}$ be a view instance such that there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$ and $D \models \mathcal{C}$. Then, under the Open World Assumption, $\mathcal{P}$ computes all the certain answers of Q in any view instance $\mathcal{I}$, i.e.* certain$(Q, \mathcal{I}) = \mathcal{P}(\mathcal{I})$.

Proof. The proof is similar to the proof of Theorem 7 and uses Lemma 10. □

Using the previous theorems we can also prove the following:

Theorem 9. *Suppose that Q is a UCQ query and $\mathcal{V}$ a set of CQ views.*

1. *Suppose that $\mathcal{I}$ is a view instance such that there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$. Then, we can compute the certain answers of Q with respect to any view instance $\mathcal{I}$ in* PTIME *data complexity.*
2. *Suppose that $\mathcal{C}$ is a set of dependencies such that the chase of Q with $\mathcal{C}$ terminates and that $\mathcal{P}$ is a maximally contained rewriting of Q under $\mathcal{C}$ with respect to UCQ. Suppose also that $\mathcal{I}$ is a view instance such that there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$ and $D \models \mathcal{C}$. Then, we can compute the certain answers of Q with respect to any view instance $\mathcal{I}$ in* PTIME *data complexity.*

Proof. 1. We know from Lemma 6 that a maximally contained rewriting of Q with respect to UCQ always exists. Since $\mathcal{I}$ is a view instance such that there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$, we know from Theorem 7 that for any view instance $\mathcal{I}$: $\mathcal{P}(\mathcal{I}) = \text{certain}(Q, \mathcal{I})$. Thus, we can compute the certain answers of Q with respect to any view instance $\mathcal{I}$ in PTIME data complexity.

2. Since $\mathcal{P}$ is a maximally contained rewriting of Q under $\mathcal{C}$ with respect to UCQ and $\mathcal{I}$ is a view instance such that there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$ and $D \models \mathcal{C}$, we know from Theorem 8 that for any view instance $\mathcal{I}$: $\mathcal{P}(\mathcal{I}) = \text{certain}(Q, \mathcal{I})$. Thus, we can compute the certain answers of Q with respect to any view instance $\mathcal{I}$ in PTIME data complexity. □

6. Conclusions

In this paper we considered conjunctive queries and views, and we investigated equivalent and maximally contained rewritings of a query using views under a set of dependencies. We proposed an efficient algorithm for finding equivalent rewritings of a query Q using a set of views $\mathcal{V}$ in the presence of a set of dependencies $\mathcal{C}$ that is such that every contained rewriting of Q using $\mathcal{V}$ under $\mathcal{C}$ has the shared variable property. Then we went on to introduce an efficient algorithm that finds maximally contained rewritings of a query Q with respect to the language of unions of conjunctive queries and in the presence of a set of weakly acyclic local-as-view tuple generating dependencies. In the rest of the paper we first proved that under the open world assumption, if no dependencies are present, a maximally contained rewriting of Q with respect to the language of unions of conjunctive queries computes exactly the certain answers of Q. Then, we considered the case where the maximally contained rewritings of Q with respect to the language of unions of conjunctive queries are considered in the presence of dependencies $\mathcal{C}$ such that the chase of Q with $\mathcal{C}$ always terminates, and we proved that such a maximally contained rewriting of Q, if it exists, computes exactly the certain answers of Q. For future work we plan to investigate the problem of finding equivalent rewritings and MCRs of a query with arithmetic comparisons in the presence of dependencies.

Acknowledgements

The first author would like to thank Rada Chirkova and Michael Compton who have read [8] carefully and have made critical comments. Both authors would also like to thank Rada Chirkova who has read the conference version of this paper [3] and has made insightful comments.

Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.is.2009.08.002.

References

[1] S. Abiteboul, O.M. Duschka, Complexity of answering queries using materialized views, in: PODS '98: Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, ACM, New York, NY, USA, 1998, pp. 254–263.
[2] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, Reading, MA, 1995.
[3] F.N. Afrati, N. Kiourtis, Query answering using views in the presence of dependencies, in: International Workshop on New Trends in Information Integration (NTII), in Conjunction with the 34th International Conference on Very Large Data Bases (VLDB 2008), , 2008, pp. 8–11.
[4] F. Afrati, C. Li, P. Mitra, Answering queries using views with arithmetic comparisons, in: PODS '02: Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, ACM, New York, NY, USA, 2002, pp. 209–220.
[5] F.N. Afrati, Rewriting conjunctive queries determined by views, in: MFCS, 2007, pp. 78–89.
[6] F.N. Afrati, C. Li, P. Mitra, Rewriting queries using views in the presence of arithmetic comparisons, Theor. Comput. Sci. 368 (1–2) (2006) 88–123.
[7] F.N. Afrati, C. Li, J.D. Ullman, Generating efficient plans for queries using views, in: SIGMOD '01: Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, 2001, pp. 319–330.
[8] F.N. Afrati, C. Li, J.D. Ullman, Using views to generate efficient evaluation plans for queries, J. Comput. Syst. Sci. 73 (5) (2007) 703–724.
[9] Q. Bai, J. Hong, M.F. McTear, Query rewriting using views in the presence of inclusion dependencies, in: WIDM '03: Proceedings of the 5th ACM International Workshop on Web Information and Data Management, ACM, New York, NY, USA, 2003, pp. 134–138.
[10] C. Beeri, M.Y. Vardi, A proof procedure for data dependencies, J. Assoc. Comput. Mach. 31 (4) (1984) 718–741.
[11] A. Calì. Query answering by rewriting in GLAV data integration systems under constraints, in: SWDB, 2004, pp. 167–184.
[12] A. Calì, D. Lembo, R. Rosati, Query rewriting and answering under constraints in data integration systems, in: IJCAI-03, MK, 2003, pp. 16–21.
[13] D. Calvanese, G.D. Giacomo, M. Lenzerini, M.Y. Vardi, View-based query processing: on the relationship between rewriting, answering and losslessness, Theor. Comput. Sci. 371 (3) (2007) 169–182.
[14] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., The MIT Press, Cambridge, MA, 2001.
[15] A. Deutsch, L. Popa, V. Tannen, Query reformulation with constraints, SIGMOD Records 35 (1) (2006) 65–73.
[16] O.M. Duschka, M.R. Genesereth, A.Y. Levy, Recursive query plans for data integration, J. Logic Programming 43 (1) (2000) 49–73.
[17] R. Fagin, P.G. Kolaitis, R.J. Miller, L. Popa, Data exchange: semantics and query answering, Theor. Comput. Sci. 336 (1) (2005) 89–124.
[18] G. Gou, M. Kormilitsin, R. Chirkova, Query evaluation using overlapping views: completeness and efficiency, in: SIGMOD '06:

Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, 2006, pp. 37–48.
[19] J. Gryz, Query rewriting using views in the presence of functional and inclusion dependencies, Inf. Syst. 24 (7) (1999) 597–612.
[20] A.Y. Halevy, Answering queries using views: a survey, VLDB J. 10 (4) (2001) 270–294.
[21] C. Koch, Query rewriting with symmetric constraints, in: FoIKS '02: Proceedings of the Second International Symposium on Foundations of Information and Knowledge Systems, Springer, London, UK, 2002, pp. 130–147.
[22] A.Y. Levy, A.O. Mendelzon, Y. Sagiv, D. Srivastava, Answering queries using views, in: Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, ACM Press, San Jose, CA, May 22–25, 1995, pp. 95–104.
[23] P. Mitra, An algorithm for answering queries efficiently using views, in: ADC '01: Proceedings of the 12th Australasian Database Conference, IEEE Computer Society, Washington, DC, USA, 2001, pp. 99–106.
[24] A. Nash, L. Segoufin, V. Vianu, Determinacy and rewriting of conjunctive queries using views: a progress report, in: ICDT, 2007, pp. 59–73.
[25] L. Popa, A. Deutsch, A. Sahuguet, V. Tannen, A chase too far?, in: Proceedings of the ACM SIGMOD Conference, , 2000, pp. 273–284.
[26] R. Pottinger, A. Halevy, Minicon: a scalable algorithm for answering queries using views, VLDB J. 10 (2–3) (2001) 182–198.
[27] A. Rajaraman, Y. Sagiv, J.D. Ullman, Answering queries using templates with binding patterns. in: Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, ACM Press, San Jose, CA, May 22–25, 1995, pp. 105–112.
[28] J.D. Ullman, H. Garcia-Molina, J. Widom, Database Systems: The Complete Book, Prentice-Hall PTR, Upper Saddle River, NJ, USA, 2001.