Consistent Answers from Integrated Data Sources

Leopoldo Bertossi[1], **Jan Chomicki**[2]
Alvaro Cortés[3], and **Claudio Gutiérrez**[4]

[1] Carleton University, School of Computer Science, Ottawa, Canada. bertossi@scs.carleton.ca
[2] University at Buffalo, Dept. of Computer Science and Engineering. chomicki@cse.buffalo.edu
[3] Pontificia Universidad Catolica de Chile, Departamento de Ciencia de Computación, Santiago, Chile. acortes@ing.puc.cl
[4] Universidad de Chile, Center for Web Research, Departamento de Ciencias de la Computación, Santiago, Chile. cgutierr@dcc.uchile.cl

Abstract. When data sources are integrated into a single global system, inconsistencies wrt global integrity constraints are likely to occur. In this paper, the notion of consistent answer to a global query in the context of the local-as-view approach to data integration is characterized. Furthermore, a methodology for generating query plans for retrieving consistent answers to global queries is introduced. For this purpose, an extension of the inverse-rules algorithm for deriving query plans for answering first-order queries is presented.

1 Introduction

In the last few years, due the increasing number of information sources that are available and may interact, the subject of data integration has been widely studied from different points of view. Topics like mediated schemas, query containment, answering queries using views, etc., have been considered in this context. However, the important issue of consistency of data derived from the integration process and query answering has attracted less attention.

A data integration system provides a uniform interface to several information sources. This interface, referred as *global schema* or *mediated schema*, is a context-dependent set of virtual relations used to formulate queries to the integrated system. When the user queries the system in terms of the global schema, a query processor or a mediator is in charge of rewriting the global query into a *query plan* that will eventually access the underlying information sources.

In order to perform this query rewriting, the processor needs a mapping between the mediated schema and the information sources. Two paradigms have been proposed to provide this mapping. One of them, called the *Local-as-View* (LAV) approach [15], considers each information source as a view defined in terms of relations in the global schema. The *Global-as-View* (GAV) approach considers each global predicate as a view defined in terms of the source relations [20, 21].

In this paper we concentrate on the LAV approach. This scenario is more flexible than GAV for adding new data sources into a global system. Actually, preexisting data sources in the system do not need to be considered when a new source is introduced. In consequence, inconsistencies are more likely to occur. Furthermore, from the point of view of studying the logical issues around consistency of data, the LAV paradigm seems to be more challenging than the GAV. The latter is closer to the classical problem of consistency of views defined over relational databases.

In the context of the LAV approach, several algorithms have been proposed to rewrite a global query into a query plan that accesses the data source relations to answer the original query [13]. Some of them assume that certain integrity constraints (ICs) hold at the global level, and they use the ICs in the query plan generation. In [12], a rewriting algorithm that uses functional and inclusion dependencies in the mediated schema is proposed. In [8], another algorithm for query plan generation that uses functional and full dependencies is introduced. This algorithm may take a global query written in Datalog as an input. In [10], a deductive, resolution based approach to data integration is presented. It may also use global integrity constraints in the deductive derivation of the query plan. Actually, there are situations where, without assuming that certain global ICs hold, no query plan can be generated [13].

However, it is not obvious that certain desirable, global ICs will hold at the global level. After all, the data is in the sources, the global relations are virtual and there may be no consistency checking mechanism at the global level. Furthermore, and particularly in the LAV approach, it is not clear what it means for the global system to be consistent, because we do not have a global *instance*. Actually, given a set of data sources, there may be several potential global instances that (re)produce the data sources as views according to the view definitions.

A *global system* consists of a global schema and a collection of materialized data sources that are described as views over the global schema. In this context, it is quite natural to pose queries at the global level, expecting to retrieve those answers that are consistent wrt a given set of global ICs, because, as we mentioned before, global ICs may be easily violated due to the lack of a global maintenance mechanism. As the following example shows, each data source, with its own, independent maintenance mechanism, can be consistent, but inconsistencies may arise when the sources are integrated.

Example 1. Consider the global relation $R(X,Y)$ and two source relations $\{V_1(a, b),\ V_1(c,d)\}$ and $\{V_2(a,c), V_2(d,e)\}$ described by the view definitions:

$$V_1(X,Y) \leftarrow R(X,Y) \qquad V_2(X,Y) \leftarrow R(X,Y).$$

Then, the global functional dependency (FD) $R: X \rightarrow Y$ is violated, but not the local FDs $V_1:\ X \rightarrow Y$, $V_2:\ X \rightarrow Y$. $\Box$

In [3], the problem of characterizing and computing consistent query answers from a single, inconsistent relational database instance was addressed. In this case, the database instance r is considered inconsistent when it does not satisfy a given set of ICs. Intuitively speaking, an answer to a query is considered to be consistent in r if it is an answer to the query in every possible *repair* of the original instance, where a repair is a new instance that satisfies the ICs and differs from r by a minimal set of tuples under set inclusion.

However, in a global integration system we do not have an explicit database instance, but we still have an intuition of what is a consistent answer to a query posed to that system.

Example 2. (example 1 continued) If we pose to the global system the query $Q:\ Ans(X,Y) \leftarrow R(X,Y)$, we obtain the answers $\{Ans(a,b), Ans(c,d), Ans(a,c), Ans(d,e)\}$. However, only the tuples $Ans(c,d), Ans(d,e)$ should be returned as consistent answers. $\square$

In this paper we will address the problem of posing queries to a possibly inconsistent global system, but retrieving as answers only those tuples that are consistent wrt the global ICs. To achieve this goal, we first formalize the notion of a consistent answer to a query posed to a global system. This notion is an adaptation of the notion proposed in [3] for a stand alone database. Next, we consider the problem of constructing algorithms for computing consistent answers from such a global system.

The computational mechanism proposed in [3] for a single relational database consists of rewriting the given query into a new query that, posed to the original, inconsistent database, gets as (normal) answers the consistent answers to the original query.

If we want to derive query plans for retrieving, hopefully all and only, those answers to a query posed, now, to a global integrated system, that are consistent wrt the desired, global ICs, we may try to use the approach in [3], rewriting the global query into a new query, and then pose the new query to the global system. The problem is that the rewritten query may not be handled by any of the existing query plan generation algorithms, e.g. [10, 15, 12], due to the presence of negation. In consequence, we need mechanisms for generating a query plan that can be applied to rewritten queries. For this purpose, we extend the "inverse-rules algorithm" from [8] to recursion-free Datalog$^\neg$ queries with built-in predicates, because this is the kind of rewritten queries we obtain. In this paper we restrict ourselves to the case of "open" sources [9], the most common scenario.

2 Preliminaries

2.1 Global schemas and view definitions

A *global schema* $\mathcal{R}$ is modeled by a finite set of relations $\{R_1, R_2, ..., R_n\}$ and a possibly infinite domain $\mathcal{D}$. With these relation symbols and the elements of $\mathcal{D}$ treated as constants, a first-order language $\mathcal{L}(\mathcal{R})$ can be defined. This language can be extended with new defined predicates and built-ins.

A *view*, denoted by a new predicate V, is defined by means of an $\mathcal{L}(\mathcal{R})$-formula of the form φ_V: $V(\bar{t}) \leftarrow body(\varphi_V)$, where $\bar{t}$ is a tuple containing variables and/or constants, and $body(\varphi_V)$ is a conjunction of $\mathcal{R}$-atoms. In consequence, we are considering views defined by conjunctive queries [1].

A *database instance* D over schema $\mathcal{R}$ can be considered as a first-order structure with domain $\mathcal{D}$, where the extensions (sets of tuples) of the relations R_i are finite (The extensions of built-in predicates may be infinite but fixed.). An *integrity constraint* is a first-order sentence ψ written in the language $\mathcal{L}(\mathcal{R})$. The instance D satisfies ψ, denoted $D \models \psi$, if ψ is true in D.

Given a database instance D over schema $\mathcal{R}$, and a view definition φ_V, $\varphi_V(D)$ denotes the extension of V obtained by applying the definition φ_V to D. If the view already has an extension v (given by the contents of a data source), it is possible that v is incomplete and stores only some of the tuples that satisfy the definition of V applied to D. In this case, we say the view extension v is *open* wrt D [2, 9]. Most mechanisms for deriving query plans are based on open sources [15, 8, 18].

Following [9], we say that a *source* S is a pair $<\varphi, v>$, where φ is the view definition, and v is an extension for φ. An *open global system* $\mathfrak{G}$ (called a *source collection* in [9]), is a finite set of open sources. The schema $\mathcal{R}$ of the global system can be read from the bodies of the view definitions. It consists of the relation names that do not have a definition in the global system. The underlying domain $\mathcal{D}$ for $\mathcal{R}$ contains all the constants appearing in view extension v_i's in the sources.

When we talk about consistency in databases wrt a set of ICs we think of instances satisfying ICs. However, in a global system for data integration there is not such a global instance, at least not explicitly. Instead, a global system $\mathfrak{G}$ defines a set of legal global instances.

Definition 1. [9] *Given an open global system* $\mathfrak{G} = \{<\varphi_1, v_1>, \ldots, <\varphi_n, v_n>\}$, *the set of legal global instances is* $Linst(\mathfrak{G}) = \{D$ *instance over* $\mathcal{R} \mid v_i \subseteq \varphi_i(D),\ i = 1, \ldots, n\}$, *where* v_i *is the extension in the source* S_i *of the view defined by* φ_i, *and* $\varphi_i(D)$ *is the set of tuples obtained by applying the view definition* φ_i *to instance* D. □

Example 3. (example 2 continued) $D = \{R_1(a, b), R_1(c, d), R_1(a, c), R_1(d, e)\}$ is a legal global instance, because $v_1 = \{(a, b), (c, d)\} \subseteq \varphi_1(D) = \{(a, b), (c, d), (a, c), (d, e)\}$ and $v_2 = \{(a, c), (d, e)\} \subseteq \varphi_2(D) = \{(a, b), (c, d), (a, c), (d, e)\}$. All supersets of D are also legal global instances. □

The inverse-rules algorithm [8] for generating query plans under the LAV approach assumes that sources are open and each source relation V has a source description that defines it as a view of the global schema: $V(\bar{X}) \leftarrow P_1(\bar{X}_1), \ldots, P_n(\bar{X}_n)$. Then, for $j = 1, \ldots n$, $P_j(\bar{X}'_j) \leftarrow V(\bar{X})$ is an inverse rule for P_j. The tuple $\bar{X}_j$ is transformed to obtain the tuple $\bar{X}'_j$ as follows: if X is a constant or is a variable in $\bar{X}$, then X is unchanged in $\bar{X}'_j$. Otherwise, X is one of the variables X_i appearing in the body of the definition of V, but not in $\bar{X}$.

In this case, X is replaced by a Skolem function term $f_{S,i}(\bar{X})$ in $\bar{X}'_j$. We denote the set of inverse rules of the collection $\mathcal{V}$ of source descriptions in $\mathfrak{G}$ by $\mathcal{V}^{-1}$.

Given a Datalog query Q and a set of conjunctive source descriptions in $\mathfrak{G}$, the construction of the query plan is as follows. All the rules from Q that contain global relations that cannot be defined (directly or indirectly) in terms of the global relations appearing in the source descriptions are deleted. To the resulting query, denoted as Q^-, the rules in $\mathcal{V}^{-1}$ are added; and the query so obtained is denoted by $(Q^-, \mathcal{V}^{-1})$. Notice that the global predicates can be seen as EDB predicates in the rules for Q. However, they become IDB predicates in $(Q^-, \mathcal{V}^{-1})$, because they appear in the heads of the rules in $\mathcal{V}^{-1}$. In consequence, the query plan is given essentially in terms of the source predicates.

3 Global Systems and Consistency

Intuitively speaking, *certain answers* to a query posed to a global system are those that can be obtained from every legal instance of the system. We expect *consistent answers* to the query to be a certain answers, but not necessarily the opposite will hold.

Definition 2. [2] *Given an open global system $\mathfrak{G}$ and a query $Q(\bar{X})$ to the system, a tuple $\bar{t}$ is a* certain answer *to Q in $\mathfrak{G}$ if for every global instance $D \in$ Linst$(\mathfrak{G})$, it holds $D \models Q[\bar{t}]$.* [1] □

We assume from here on that we have a fixed set of global first-order integrity constraints IC on a given global schema. We also assume that the set of ICs is consistent as a set of logical sentences. Furthermore, we will also assume that the set IC is *generic*, in the sense that IC does not determine, independently from the contents of the database, the presence/absence in/from the database of any piece of ground atomic knowledge. The ICs used in database praxis are always generic.

Definition 3. *A set IC of ICs is* generic *if there is no ground literal L in the language of IC such that $IC \models L$.* □

We need a precise definition of consistency of a global system that captures the intuitive notion of consistency related to the definitions of the sources and the data stored in the sources.

Definition 4. *Given a global system, $\mathfrak{G}$, a* minimal global instance *of $\mathfrak{G}$ is an instance $D \in$ Linst$(\mathfrak{G})$ that is minimal wrt set inclusion, i.e. there is no other instance in Linst$(\mathfrak{G})$ that is a proper subset of D (as a set of atoms). We denote by mininst$(\mathfrak{G})$ the set of minimal legal global instances of $\mathfrak{G}$ wrt set inclusion.* □

[1] $D \models Q[\bar{t}]$ means that query $Q(\bar{X})$ becomes true in instance D, when tuple of variables $\bar{X}$ is assigned the values in the tuple $\bar{t}$ of database elements.

Definition 5. *A global system $\mathfrak{G}$ is* consistent *wrt to a set of global integrity constraints, IC, if every minimal legal global instance of $\mathfrak{G}$ satisfies IC: for all $D \in mininst(\mathfrak{G})$, $D \models IC$.* □

Example 4. Consider $\mathfrak{G} = \{S_1, S_2\}$, with

$$S_1 = \langle V_1(X,Y) \leftarrow P(X,Z) \wedge Q(Z,Y), \{V_1(a,b)\}\rangle$$

$$S_2 = \langle V_2(X,Y) \leftarrow P(X,Y), \{V_2(a,c)\}\rangle.$$

In this case, the elements of $mininst(\mathfrak{G})$ are of the form $D_z = \{P(a,z), Q(z,b), P(a,c)\}$ for some $z \in \mathcal{D}$. The global FD $P(X,Y)$: $X \rightarrow Y$ is violated exactly in those minimal legal instances D_z for which $z \neq c$. Thus, the global system is inconsistent. □

A global system $\mathfrak{G}$ could be inconsistent in the sense of not satisfying the given set of ICs, but still be *possible* or *realizable* in the sense that $Linst(\mathfrak{G}) \neq \emptyset$.

Definition 6. *(a) The ground tuple $\bar{a}$ is a minimal answer to a query Q posed to a global system $\mathfrak{G}$ if for every minimal instance $D \in mininst(\mathfrak{G})$, $\bar{a} \in Q(D)$, where $Q(D)$ is the answer set for Q in D.*
(b) We denote by $Minimal_{\mathfrak{G}}(Q)$ the set of minimal answers to query Q in $\mathfrak{G}$. □

It is clearly the case that the minimal answers contain the certain answers. For monotone queries [1], the notions of minimal and certain answers coincide. Nevertheless, in example 4 the query $Ans(X,Y) \leftarrow \neg P(X,Y)$ has (b,a) as a minimal answer, but (b,a) is not a certain answer, because there are legal instances that contain $P(b,a)$. Later on our queries will be allowed to contain negation. Since consistent answers have been defined relative to minimal global instances, for us the relevant notion of answer is that of minimal answer. Notice that this assumption is like imposing a form of closed-world assumption on the global instances associated with the local sources.

Given a database instance D, we denote by $\Sigma(D)$ the set of ground formulas $\{P(\bar{a}) \mid P \in \mathcal{R} \text{ and } D \models P(\bar{a})\}$.

Definition 7. [3] *(a) Let D, D' be database instances over the same schema and domain. The distance, $\Delta(D,D')$, between D and D' is the symmetric difference:*

$$\Delta(D,D') = (\Sigma(D) \setminus \Sigma(D')) \cup (\Sigma(D') \setminus \Sigma(D)).$$

(b) For database instances D, D', D'', we define $D' \leq_D D''$ if $\Delta(D,D') \subseteq \Delta(D,D'')$, i.e., if the distance between D and D' is less than or equal to the distance between D and D''. □

Definition 8. *(based on* [3]*) Let $\mathfrak{G}$ be a global system and IC a set of global ICs. A* repair *of $\mathfrak{G}$ wrt IC is a global database instance D', i.e. an instance over the global schema $\mathcal{R}$, such that:*
(a) $D' \models IC$ and
(b) D' is $\leq_D$-minimal for some $D \in mininst(\mathfrak{G})$. □

We can see that a repair of a global system is a global database instance that minimally differs from a minimal legal global database instance. If $\mathfrak{G}$ is consistent (definition 5), then the repairs are exactly the elements in $mininst(\mathfrak{G})$.

Example 5. Consider the global system $\mathfrak{G}_4 = \{S_1, S_2\}$, with

$$S_1 = \langle V_1(X) \leftarrow R(X,Y), \{V_1(a)\}\rangle,$$
$$S_2 = \langle V_2(X) \leftarrow R(X,Y), \{V_2(a)\}\rangle,$$

and the global FD $R(X,Y)\colon\ X \rightarrow Y$. In this case, $D = \{R(a,b_1), R(a,b_2)\} \in Linst(\mathfrak{G}_4)$, but $D \not\models IC$. However, $D \notin mininst(\mathfrak{G}_4)$. Actually, the elements in $mininst(\mathfrak{G}_4)$ are of the form $\{R(a,b)\}$, for some b in the global database domain. The elements in $mininst(\mathfrak{G}_4)$ coincide with the repairs. Notice that $\mathfrak{G}_4$ is a consistent global system. □

Notice that in this definition of repair we are not requiring that a repair respects the property of the sources of being open, i.e. that the extension of each view in the repair contains the corresponding view extension in the source. Thus, it may be the case that a repair – still a global instance – does not belong to $Linst(\mathfrak{G})$. If we do not allow this, a global system might not be repairable. The notion of repair will be used as an auxiliary concept to define the notion of consistent answer.

Example 6. Consider the global system $\mathfrak{G}_5 = \{S_1, S_2\}$, with

$$S_1 = \langle V_1(X,Y) \leftarrow R(X,Y), \{V_1(a,b_1)\}\rangle,$$
$$S_2 = \langle V_2(X,Y) \leftarrow R(X,Y), \{V_2(a,b_2)\}\rangle,$$

and the FD $R(X,Y):\ X \rightarrow Y$. The only element in $mininst(\mathfrak{G}_5)$ is $D_0 = \{R(a,b_1), R(a,b_2)\}$, that does not satisfy IC. The global system is inconsistent. The only repairs are the global instances that minimally differ from D_0 and satisfy the FD, namely $D_0^1 = \{R(a,b_1)\}$ and $D_0^2 = \{R(a,b_2)\}$. Notice that they do not belong to $Linst(\mathfrak{G}_5)$. □

Definition 9. *(a) Given a global system $\mathfrak{G}$, a set of global integrity constraints IC, and a global first-order query $Q(\bar{X})$, we say that a (ground) tuple $\bar{t}$ is a consistent answer to Q wrt IC, denoted by $\mathfrak{G} \models_c Q[\bar{t}]$, iff for every repair D of $\mathfrak{G}$, $D \models Q[\bar{t}]$.*
(b) We denote by $Consis_{\mathfrak{G}}(Q)$ the set of consistent answers to query Q in $\mathfrak{G}$. □

Example 7. (example 6 continued) For the query $Q_1(X):\ \exists Y\ R(X,Y)$, a is a consistent answer, i.e. $\mathfrak{G}_5 \models_c\ \exists Y\ R(X,Y)[a]$. On the other hand, the query $Q_2(X,Y) : R(X,Y)$, does not have any consistent answer. □

Proposition 1. *Given an open global system $\mathfrak{G}$, a set of generic global integrity constraints IC, and a global query Q:*

(a) Every consistent answer is a minimal answer.
(b) If $\mathfrak{G}$ is consistent wrt IC, then every minimal answer is consistent.[2] □

[2] The proof of this proposition and of all the other explicitly stated results can be found in [5].

4 Computing Consistent Query Answers

After having given a semantic definition of consistent answer to a global query from a global system, we concentrate on the computational problem of consistent query answering (CQA).

In [3], in the context of a single, possibly inconsistent relational database, a consistent answer to a query was defined as an answer to that query that is obtained from every (minimal) repair of the database. There, for the purpose of computing consistent answers, an operator T^ω was introduced. It does the following: Given a first-order query $Q(\bar{X})$, a modified query $T^\omega(Q(\bar{X}))$ is computed. The new query $T^\omega(Q(\bar{X}))$ is posed to the original database, and the returned answers are consistent in the semantic sense.

We consider *universal* first-order integrity constraints expressed in the so-called "standard format" [3]: $\forall(\bigvee_{i=1}^{m} l_i(\bar{X}_i) \vee \varphi)$, where l_i is a database literal and φ is a formula containing built-in predicates only.

The new query $T^\omega(\varphi(\bar{x}))$ is computed by iterating an operator T which transforms an arbitrary query by appending the corresponding *residue* to each database literal appearing in the query, until a fixpoint is reached. The residue of a database literal forces the local satisfaction of the ICs for the tuples satisfying the literal. Residues can be obtained by resolution between the table and the ICs. The methodology based on the T operator is applicable to queries that are conjunctions of literals. For the usual universal ICs found in database praxis, e.g. functional dependencies, a fixpoint can be reached in a finite number of steps (see [3] for details). In the rest of this paper we will concentrate on this case.

Example 8. Consider $IC = \{R(X) \vee \neg P(X) \vee \neg Q(X),\ P(X) \vee \neg Q(X)\}$ and the query $\varphi(X) = Q(X)$. The residue of $Q(X)$ wrt the first IC is $(R(X) \vee \neg P(X))$, because if $Q(X)$ is to be satisfied, then that residue has to be true if the IC is to be satisfied too. Similarly, the residue of $Q(X)$ wrt the second IC is $P(X)$. The operator T iteratively appends the residues to the tables in the queries.

$$
\begin{aligned}
T^1(\varphi(X)) &= Q(X) \wedge (R(X) \vee \neg P(X)) \wedge P(X).\\
T^2(\varphi(X)) &= T(T^1(\varphi(X))) = Q(X) \wedge (T(R(X)) \vee T(\neg P(X))) \wedge T(P(X)).\\
&= Q(X) \wedge (R(X) \vee (\neg P(X) \wedge \neg Q(X))) \wedge P(X) \wedge (R(X) \vee \neg Q(X)).\\
T^3(\varphi(X)) &= Q(X) \wedge (R(X) \vee (\neg P(X) \wedge T(\neg Q(X)))) \wedge P(X) \wedge (T(R(X)) \vee \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad T(\neg Q(X))).
\end{aligned}
$$

Since $T(\neg Q(X)) = \neg Q(X)$ and $T(R(X)) = R(X)$, we obtain $T^2(\varphi(X)) = T^3(\varphi(X))$, and a fixpoint has been reached. Since $T^\omega(\varphi(X)) := \bigwedge_{n<w}\{T^n(\varphi(X))\}$, here we obtain $T^\omega(\varphi(X)) = T^1(\varphi(X)) \wedge T^2(\varphi(X))$. □

In the context of integration of open data sources under the LAV approach, we now present an algorithm to compute consistent answers to queries that are conjunctions of literals wrt global universal integrity constraints, IC.

Given a global query $Q(\bar{X})$, the algorithm iteratively applies the T operator, and next, the rewritten query is answered by producing an appropriate query plan. A high level description of the algorithm is as follows:

Algorithm for CQA
Input: a global query Q that is a conjunction of literals.
Output: a query plan to obtain consistent answers to Q.

1. Rewrite $Q(\bar{X})$ into the first-order query $T^\omega(Q(\bar{X}))$ using IC.
2. Using a standard methodology [1, 16], transform $T^\omega(Q(\bar{X}))$ into a recursion-free Datalog$^\neg$ program $\Pi(T^\omega(Q))$, possibly containing built-ins.
3. Find a query plan, $Plan(\Pi(T^\omega(Q)))$ to answer $\Pi(T^\omega(Q))$ seen as a query to the global system.
4. Evaluate the query plan on the view extensions of $\mathfrak{G}$ to compute a set of answers.

Program $\Pi(T^\omega(Q))$ may contain negation. There are currently no mechanisms to obtain query plans for global queries containing negation (although some heuristics are sketched in [10]). On the positive side, the generated Datalog$^\neg$ program does not contain recursion. $Plan(\Pi(T^\omega(Q)))$ is a new Datalog$^\neg$ query program that is run on top of the data sources that act as the extensional database.

Example 9. Consider the global system $\mathfrak{G}_8 = \{S_1, S_2\}$, with

$$S_1 = \langle V_1(X,Y) \leftarrow R(X,Y), \{V_1(a,b)\}\rangle$$

$$S_2 = \langle V_2(X,Y) \leftarrow R(X,Y), \{V_2(a,c), V_2(c,d)\}\rangle,$$

the FD $R(X,Y)$: $X \rightarrow Y$, and the global query $Q(X,Y)$: $R(X,Y)$.

Here, the only element in $mininst(\mathfrak{G}_8)$ is $D_0 = \{R(a,b), R(a,c), R(c,d)\}$. The only minimal instance violates FD through the tuples $[a,b]$, $[a,c]$. In consequence, the global system $\mathfrak{G}_8$ is inconsistent. It has two repairs, $D_0^1 = \{R(a,b), R(c,d)\}$ and $D_0^2 = \{R(a,c), R(c,d)\}$. The only consistent answer to the query is the tuple $[c,d]$.

We now apply the Algorithm for CQA. First of all, we need the FD expressed in a first-order language, it becomes $\forall XYZ(R_1(X,Y) \wedge R_1(X,Z) \rightarrow Y = Z)$. At step 1. the query has to be rewritten. For functional dependencies, the operator T has a finite fixpoint and we obtain

$$T^\omega(Q(X,Y))\colon\ R(X,Y) \wedge \neg\exists Z(R(X,Z) \wedge Z \neq Y). \tag{1}$$

This query can also be translated into the following Datalog$^\neg$ program $\Pi(T^\omega(Q))$:

$$Ans(X,Y) \leftarrow R(X,Y),\ not\ S(X,Y)$$

$$S(X,Y) \leftarrow R(X,Z), Y \neq Z.$$

We need a query plan to answer this query program containing negation. □

5 Plans for Queries with Negation

The inverse-rules algorithm in [8] for generating a query plans for a global Datalog query Q produces an answer set that is *maximally-contained* in the answer set for Q. In [17], it is shown that such a maximally-contained query plan retrieves all certain answers to the query Q.

A limitation of the inverse-rules algorithm for query plans is that it allows Datalog queries only. In our case, even if we start with a conjunctive global query, the query program obtained after transforming the query rewritten using the T operator may contain negation. In consequence, we need to find plans for recursion-free Datalog$^\neg$ query programs with built-in predicates, like the $\Pi(T_w(Q))$'s. Notice that the absence of recursion immediately makes the resulting query programs stratified [1].

Now, we present an extension of the algorithm in [8] to the case where the query program contains negation, but no recursion.

Given a recursion-free Datalog$^\neg$ query Q in terms of global and defined relations, the extended inverse-rules algorithm works analogously to the one presented in [8], except that in the case a rule in Q contains a negated literal in the body, say $S(\bar{X}) \leftarrow \ldots, L_1(\bar{X}), \neg G(\bar{X}), L_n(\bar{X}), \ldots$, it is checked if G can be eventually evaluated in terms of the global relations appearing in the source descriptions (because those are the global relations that can be eventually evaluated using the inverse rules). If this is not the case, then that goal is eliminated, obtaining the modified rule $S(\bar{X}) \leftarrow \ldots, L_1(\bar{X}), L_n(\bar{X}), \ldots$. We show the methodology by means of an example[3].

Example 10. Consider the global system $\mathfrak{G}_9 = \{S_1, S_2\}$ with

$$\begin{aligned} S_1 &= \langle V_1(X,Z) \leftarrow R_1(X,Y), R_2(Y,Z), \{V_1(a,b)\}\rangle \\ S_2 &= \langle V_2(X,Y) \leftarrow R_3(X,Y), \{V_2(c,d)\}\rangle. \end{aligned}$$

and the global query Q:[4]

$$\begin{aligned} Ans(X,Z) &\leftarrow R_1(X,Y), R_2(Y,Z),\ not\ R_4(X,Y) \\ R_4(X,Y) &\leftarrow R_3(X,Y),\ not\ R_5(X,Y) \qquad (2) \\ R_7(X) &\leftarrow R_1(X,Y), R_6(X,Y). \qquad (3) \end{aligned}$$

The inverses rules $\mathcal{V}^{-1}$ are obtained from the source descriptions:

$$\begin{aligned} R_1(X, f_1(X,Z)) &\leftarrow V_1(X,Z) \\ R_2(f_1(X,Z), Z) &\leftarrow V_1(X,Z) \\ R_3(X,Y) &\leftarrow V_2(X,Y). \end{aligned}$$

[3] It should be easy to extend this methodology to stratified Datalog$^\neg$ queries, but we do not need this extension here.

[4] This example, containing implicit existential quantifiers, is used to illustrate the extended inverse-rules algorithm only. In this paper we are not dealing with the problem of computing consistent answers to existentially quantified conjunctive global queries.

To compute a query plan for Q, we first need Q^-:

$$
\begin{aligned}
Ans(X,Z) &\leftarrow R_1(X,Y), R_2(Y,Z),\ not\ R_4(X,Y)\\
R_4(X,Y) &\leftarrow R_3(X,Y).
\end{aligned}
$$

The literal *not* $R_5(X,Y)$ was eliminated from rule (2) because it does not appear in any source description. For the same reason (the literal $R_6(X,Y)$ does not appear in any source description), rule (3) was eliminated. Then, the query plan $(Q^-, \mathcal{V}^{-1})$ is:

$$
\begin{aligned}
Ans(X,Z) &\leftarrow R_1(X,Y), R_2(Y,Z),\ not\ R_4(X,Y)\\
R_4(X,Y) &\leftarrow R_3(X,Y)\\
R_1(X, f_1(X,Z)) &\leftarrow V_1(X,Z)\\
R_2(f_1(X,Z), Z) &\leftarrow V_1(X,Z)\\
R_3(X,Y) &\leftarrow V_2(X,Y).
\end{aligned}
$$

Finally, the query plan can be evaluated in a bottom-up manner starting from the sources V_1, V_2 to retrieve $Ans(a,b)$ as final answer for the global query Q. Tuples at the answer level that contain Skolem function symbol f_1 can be pruned from the answer set. □

Now we will show that the generated plan can be finitely evaluated in a bottom-up manner, and that the query plan is maximally contained in the query.

5.1 Containment

In this section we will establish that the resulting query plan is maximally contained in the original query. We will concentrate on recursion free Datalog$^\neg$ programs (including built-ins).

Given a query plan P for Q using $\mathcal{V}$, the *expansion* P^{exp} of P is obtained from P by replacing all source relations in P with the bodies of the corresponding views $\mathcal{V}$, and using fresh variables for existential variables in the views. That is, the source relations are eliminated by reinverting the inverse rules, in order to be in position to compare the original query, expressed in terms of global predicates, and the query plan.

Example 11. (example 10 continued) To compute P^{exp}, we first rename the IDB predicates in P.

$$
\begin{aligned}
Ans(X,Z) &\leftarrow P_1(X,Y), P_2(Y,Z),\ not\ P_4(X,Y)\\
P_4(X,Y) &\leftarrow P_3(X,Y)\\
P_1(X, f_1(X,Z)) &\leftarrow V_1(X,Z) \qquad (4)\\
P_2(f_1(X,Z), Z) &\leftarrow V_1(X,Z) \qquad (5)\\
P_3(X,Y) &\leftarrow V_2(X,Y). \qquad (6)
\end{aligned}
$$

The expansion is the following Datalog$^\neg$ query P^{exp}:

$$\begin{aligned}
Ans(X,Z) &\leftarrow P_1(X,Y), P_2(Y,Z),\ not\ P_4(X,Y)\\
P_4(X,Y) &\leftarrow P_3(X,Y)\\
P_1(X, f_1(X,Z)) &\leftarrow R_1(X,U), R_2(U,Z)\\
P_2(f_1(X,Z),Z) &\leftarrow R_1(X,U), R_2(U,Z)\\
P_3(X,Y) &\leftarrow R_3(X,Y).
\end{aligned}$$

□

Remark 1. Notice that in the extended plan obtained in the previous example, we could keep the rules (4), (5), (6), plus the source definitions in S_1, S_2 of example 10. Sometimes we will do this in the rest of this section.

Theorem 1. *For every recursion free Datalog$^\neg$ program Q, every set of conjunctive source descriptions $\mathcal{V}$, and all finite instances of the source relations, the query plan $(Q^-, \mathcal{V}^{-1})$ has a unique finite minimal fixpoint. Furthermore, bottom-up evaluation is guaranteed to terminate, and produces this unique fixpoint.* □

Definition 10. *(maximally-contained plan* [17]*) A query plan P is maximally contained in a query Q using views $\mathcal{V}$ if $P^{exp} \subseteq Q$ and for every query plan P' such that $(P')^{exp} \subseteq Q$, it holds $P' \subseteq P$.*[5] □

If $S = \{v_1, \ldots, v_m\}$ is a set of extensions for the sources, and P is a query plan, then $P(S)$ denotes the extension of the query predicate Ans, obtained by evaluating P on S. Notice that $P(S)$ may contain tuples with Skolem function symbols. We denote by $P(S)\!\downarrow$ the set of tuples in $P(S)$ that do not contain function symbols, and by $P\!\downarrow$ the plan that computes this pruned extension.

In order to prove maximal containment, the notion of proof tree presented in [19] for Datalog queries can be extended to recursion free Datalog$^\neg$ queries (see [5]).

Theorem 2. *(compare* [8]*, thm. 8) For every recursion free Datalog$^\neg$ program Q and every set of conjunctive source descriptions $\mathcal{V}$, the query plan $(Q^-, \mathcal{V}^{-1})\!\downarrow$ is maximally contained in Q.* □

The previous theorem can be restricted to the case of minimal global instances (see [5] for details). That is, the query plan $(Q^-, \mathcal{V}^{-1})\downarrow$ is contained in Q, i.e. $(Q^-, \mathcal{V}^{-1})^{exp}(D) \subseteq Q(D)$ for every minimal global instance D (this is an immediate consequence of the theorem, that holds for arbitrary global instances); and every other plan, whose expansion is contained in Q for minimal global instances, is also contained in $(Q^-, \mathcal{V}^{-1})\downarrow$ (this follows from a particular minimal global instance D constructed in the proof). As a consequence, we obtain the following theorem.

Theorem 3. *Given an open global system $\mathfrak{G}$, IC a set of generic ICs, and a recursion free Datalog$^\neg$ query Q with built-ins, the plan $Plan(Q)$ obtained with the extended inverse-rules algorithm retrieves exactly $Minimal_{\mathfrak{G}}(Q)$.* □

[5] Here, $P^{exp} \subseteq Q$ means that the extension of the query predicate in P^{exp} in included in the extension of Q for every database instance over the global schema.

6 The CQA Algorithm Revisited

We have presented an algorithm for generation of query plans for global queries that are Datalog$^\neg$ queries without recursion.

Example 12. (example 9 continued) Applying the extended inverse-rules methodology we obtain the following query plan $Plan(\Pi(T^\omega(Q)))$:

$$\begin{aligned} Ans'(X,Y) &\leftarrow R(X,Y),\ not\ S(X,Y) \quad (7)\\ S(X,Y) &\leftarrow R(X,Z), Y \neq Z \quad (8)\\ R(X,Y) &\leftarrow V_1(X,Y)\\ R(X,Y) &\leftarrow V_2(X,Y). \end{aligned}$$

This query plan can be evaluated on the view extensions in $\mathfrak{G}_8$ either top-down or bottom-up. In the latter case, a domain predicate for variable Y in rule (8) can be used, that due to the first goal in (7) can be defined as containing the values of the second attribute of R.[6] In either case, the only answer obtained is $Ans'(c,d)$. This answer is consistent: $\mathfrak{G}_8 \models_c R(X,Y)[c,d]$. Notice that, instead of this query program, the original query (1) could be evaluated using SQL2, defining first R as a view that is the union of V_1 and V_2. □

Proposition 2. *Given an open global system $\mathfrak{G}$, IC a set of generic ICs, the consistent answers to a conjunctive global query Q correspond to $Minimal(T^\omega(Q))$. Furthermore, if $T^\omega(Q)$ is monotone, then its certain answers are the consistent answers to Q.* □

Theorem 4. *Given an open global system $\mathfrak{G}$, IC a set of generic ICs, and a conjunctive global query Q, $Plan(\Pi(T^\omega(Q))$ retrieves exactly $Consis_{\mathfrak{G}}(Q)$.* □

7 Conclusions

We have concentrated on open sources. In the future, it would interesting to extend the semantics of consistent query answers to the case where the global system contains also *closed* and *clopen* sources. The extension of the algorithm presented in this paper to those cases is left for future work. The work in [9] introduces a useful framework to deal with those more general global systems.

The methodology we have presented here is based on the methodology presented in [3]. In consequence, it applies to a limited class of queries and constraints. Other approaches to consistent query answering from a single database are based on logic programs with stable model semantics [4, 6, 11]. They can handle general first-order queries. It would be interesting to explore how the methodology presented there can be integrated with the methodology presented in this paper in order to consistently answer queries posed to global integrated

[6] A general solution consists in defining such a domain as containing the (appropriate) data values stored in the sources.

systems under the LAV approach. Consistency issues under the GAV approach are treated in [14].

Acknowledgments: Work supported by FONDECYT Grant 1000593, NSF Grant INT-9901877/CONICYT Grant 1998-02-083, NSF Grant IIS-0119186, Carleton University Start-Up Grant 9364-01, NSERC Grant 250279-02, Nucleus Millennium for Web Research (Mideplan, Grant P01-029-F). We are grateful to Alberto Mendelzon for stimulating and useful conversations.

References

1. Abiteboul, S.; Hull, R. and Vianu, V. *Foundations of Databases.* Addison-Wesley, 1995.
2. Abiteboul, A. and Duschka, O. Complexity of Answering Queries Using Materialized Views. In *Proc. 9th Annual ACM Symp. on the Theory of Computing*, ACM Press, 1998, pp. 254-263.
3. Arenas, M.; Bertossi, L. and Chomicki, J. Consistent Query Answers in Inconsistent Databases. In *Proc. ACM Symposium on Principles of Database Systems (ACM PODS'99)*, ACM Press, 1999, pp. 68–79.
4. Arenas, M.; Bertossi, L. and Chomicki, J. Specifying and Querying Database Repairs Using Logic Programs with Exceptions. In *Flexible Query Answering Systems. Recent Developments.* H. Larsen, J. Kacprzyk, S. Zadrozny, H. Christiansen (eds.), Springer, 2000, pp. 27–41.
5. Bertossi, L.; Chomicki, J.; Cortes, A. and Gutierrez, C. Consistent Answers from Integrated Data Sources. Extended version. May 2002. http://www.scs.carleton.ca/~bertossi/papers/paperleo14.ps
6. Barcelo, P. and Bertossi, L. Repairing Databases with Annotated Predicate Logic. In *Proc. Ninth International Workshop on Non-Monotonic Reasoning (NMR'2002). Special session on Changing and Integrating Information: From Theory to Practice.* S. Benferhat and E. Giunchiglia (eds.), Morgan Kaufmann Publishers, 2002, pp. 160 – 170.
7. Cali, A.; Calvanese, D.; De Giacomo, G. and Lenzerini, M. Data integration Under Integrity Constraints. In *Proc. of the 14th Conf. on Advanced Information Systems Engineering (CAiSE 2002)*, Springer LNCS 2348, 2002, pp. 262–279.
8. Duschka, O.; Genesereth, M. and Levy, A. Recursive Query Plans for Data Integration. *Journal of Logic Programming*, 2000, 43(1):49-73.
9. Grahne, G. and Mendelzon, A. Tableau Techniques for Querying Information Sources through Global Schemas. In *Proc. of the Int. Conf. on Database Theory (ICDT'99)*, Springer LNCS 1540, 1999, pp. 332–347.
10. Grant, J. and Minker, M. A Logic-based Approach to Data Integration. *Theory and Practice of Logic Programming*, 2002, 2(3):323-368.
11. Greco, G.; Greco, S. and Zumpano, E. A Logic Programming Approach to the Integration, Repairing and Querying of Inconsistent Databases. In *Proc. 17th International Conference on Logic Programming (ICLP'01)*, Ph. Codognet (ed.), Springer LNCS 2237, 2001, pp. 348–364.
12. Gryz, J. Query Rewriting Using Views in the Presence of Functional and Inclusion Dependencies. *Information Systems*, 1999, 24(7):597–612.
13. Halevy, A. Answering Queries using Views: A Survey. *VLDB Journal*, 2001, 10(4):270–294.

14. Lembo, D.; Lenzerini, M. and Rosati, R. Source Inconsistency and Incompleteness in Data Integration. In *Proc. of the 9th International Workshop Knowledge Representation meets Databases (KRDB'02)*, 2002. http://CEUR-WS.org/Vol-54/
15. Levy, A.; Rajaraman, A. and Ordille, J. Querying Heterogeneous Information Sources using Source Descriptions. In *Proceedings of the 22nd International Conference on Very Large Databases (VLDB'96)*, Morgan Kaufmann Publishing Co., 1996, pp. 251–262.
16. Lloyd, J. *Foundations of Logic Programming*. Springer, 1987.
17. Millstein, T.; Levy, A. and Friedman, M. Query Containment for Data Integration Systems. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'00)*, ACM Press, 2000, pp. 67–75.
18. Pottinger, R. and Levy, A. A Scalable Algorithm for Answering Queries Using Views. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB'00)*, Morgan Kaufmann Publishing Co., 2000, pp. 484–495.
19. Ramakrishnan, R.; Sagiv, Y.; Ullman, J.D. and Vardi, M.Y. Proof-tree Transformation Theorems and their Applications. In *Proceedings ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, (PODS'89)*, ACM Press, 1989, pp. 172–181.
20. Ullman, J. Information Integration using Logical Views. In *Proc. of the Int. Conf. on Database Theory (ICDT'97)*, Springer LNCS 1186, 1997, pp. 19–40.
21. Ullman, J.D. Information Integration Using Logical Views. *Theoretical Computer Science*, 2000, 239(2):189-210.