Data Integration under Integrity Constraints

Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini

Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
lastname@dis.uniroma1.it,
http://www.dis.uniroma1.it/~*lastname*

Abstract. Data integration systems provide access to a set of heterogeneous, autonomous data sources through a so-called global schema. There are basically two approaches for designing a data integration system. In the global-centric approach, one defines the elements of the global schema as views over the sources, whereas in the local-centric approach, one characterizes the sources as views over the global schema. It is well known that processing queries in the latter approach is similar to query answering with incomplete information, and, therefore, is a complex task. On the other hand, it is a common opinion that query processing is much easier in the former approach. In this paper we show the surprising result that, when the global schema is expressed in the relational model with integrity constraints, even of simple types, the problem of incomplete information implicitly arises, making query processing difficult in the global-centric approach as well. We then focus on global schemas with key and foreign key constraints, which represents a situation which is very common in practice, and we illustrate techniques for effectively answering queries posed to the data integration system in this case.

1 Introduction

Integrating heterogeneous data sources is a fundamental problem in databases, which has been studied extensively in the last two decades both from a formal and from a practical point of view [1,2,3,4,5,6]. Recently, mostly driven by the need to integrate data sources on the Web, much of the research on integration has focussed on so called *data integration* [7,8,6]. Data integration is the problem of combining the data residing at different sources, and providing the user with a unified view of these data. Such a unified view is structured according to a so-called global schema, which represents the intensional level of the integrated and reconciled data, and provides the elements for expressing the queries over the data integration system. It follows that, in formulating the queries, the user is freed from the knowledge on where data are, how data are structured at the sources, and how data are to be merged and reconciled to fit into the global schema.

The interest in this kind of systems has been continuously growing in the last years. Many organizations face the problem of integrating data residing in several sources. Companies that build a Data Warehouse, a Data Mining,

A. Banks Pidduck et al. (Eds.): CAISE 2002, LNCS 2348, pp. 262–279, 2002.

or an Enterprise Resource Planning system must address this problem. Also, integrating data in the World Wide Web is the subject of several investigations and projects nowadays. Finally, applications requiring accessing or re-engineering legacy systems must deal somehow with data integration.

The design of a data integration system is a very complex task, which requires addressing several different issues. Here, we concentrate on two basic issues:

1. specifying the mapping between the global schema and the sources,
2. processing queries expressed on the global schema.

With regard to issue (1), two basic approaches have been used to specify the mapping between the sources and the global schema [7,7,9]. The first approach, called *global-centric* [10,11,12], requires that the global schema is expressed in terms of the data sources. More precisely, to every element of the global schema, a view over the data sources is associated, so that its meaning is specified in terms of the data residing at the sources. In general, the views associated to the elements of the global schema are considered *sound*, i.e., all the data provided by a view satisfies the corresponding element of the global schema, but there may be additional data satisfying the element not provided by the view. The second approach, called *source-centric* [13,14,15], requires the global schema to be specified independently from the sources. In turn, the sources are defined as views over the global schema. Comparisons of the two approaches are reported in [8,16]. In this paper, we study global-centric data integration systems, and, according to the usual approach, we assume that the views associated to the elements of the global schema are sound.

Issue (2) is concerned with one of the most important problems in the design of a data integration system, namely, the choice of the method for computing the answer to queries posed in terms of the global schema. For this purpose, the system should be able to reformulate the query in terms of a suitable set of queries posed to the sources. These queries are then shipped to the sources, and the results are assembled into the final answer. It is well known that processing queries in the source-centric approach is a difficult task [8,17,14,18,19]. Indeed, in this approach the only knowledge we have about the data in the global schema is through the views representing the sources, and such views provide only partial information about the data. Therefore, extracting information from the data integration system is similar to query answering with incomplete information, which is a complex task [20]. On the other hand, query processing is considered much easier in the global-centric approach, where in general it is assumed that answering a query basically means unfolding its atoms according to their definitions in terms of the sources [7]. The reason why unfolding does the job is that the global-centric mapping essentially specifies a single database satisfying the global schema, and evaluating the query over this unique database is equivalent to evaluating its unfolding over the sources.

While this is a common opinion in the literature, we show in this paper that the presence of integrity constraints in the global schema poses new challenges, specially related to the need of taking the semantics of constraints into account during query processing. The importance of allowing integrity constraints in the

global schema has been stressed in several work on data integration [15,21,22]. Since the global schema acts as the interface to the user for query formulation, it should mediate among different representations of overlapping worlds, and therefore the schema definition language should incorporate flexible and powerful representation mechanisms, such as the ones based on semantic integrity constraints.

The first contribution in this paper is to show that, when the global schema contains integrity constraints, even of simple forms, the semantics of the data integration system is best described in terms of a set of databases, rather than a single one, and this implies that, even in the global-centric approach, query processing is intimately connected to the notion of querying *incomplete databases*. The fact that the problem of incomplete information is overlooked in current approaches can be explained by observing that traditional data integration systems follow one of the following strategies: they either express the global schema as a set of plain relations without integrity constraints, or consider the sources as exact (see, e.g., [23,24]), as opposed to sound. On the contrary, the goal of our work is to study the more general setting where the global schema contains integrity constraints, and sources are considered sound (but not necessarily complete). The above result demonstrates that, in this case, we have to account for multiple global databases.

The second contribution of the paper is to study the case of global schemas expressed in the relational model with key and foreign key constraints, which represents a situation very common in practice. Although the problem of multiple global databases arises in this case, we have devised techniques for effectively answering queries posed to the data integration system. The resulting algorithm runs in polynomial time with respect to data complexity, i.e., with respect to the size of data at the sources.

The paper is organized as follows. In Section 2 we describe a formal framework for data integration. In Section 3 we show that the presence of integrity constraints in the global schema complicates the task of query processing. In Sections 4 and 5 we present our query processing algorithm for the case of global relational schema with key and foreign key constraints. Section 6 concludes the paper.

2 Framework for Data Integration

In this section we illustrate our formalization of a data integration system, which is based on the relational model with integrity constraints.

In the relational model, predicate symbols are used to denote the relations in the database, whereas constant symbols denote the objects and the values stored in relations. We assume to have a fixed (infinite) alphabet Γ of constants, and, if not specified otherwise, we will consider only databases over such alphabet. We adopt the so-called *unique name assumption*, i.e., we assume that different constants denote different objects. A *relational schema* $\mathcal{C}$ is constituted by:

- An *alphabet* $\mathcal{A}$ of predicate (or relation) symbols, each one with the associated arity, i.e., the number of arguments of the predicate (or, attributes of the relation). We do not use names for referring to attributes, rather, we simply use the numbers corresponding to their positions.
- A set of *integrity constraints*, i.e., assertions on the symbols of the alphabet $\mathcal{A}$ that express conditions that are intended to be satisfied in every database coherent with the schema.

A relational database (or simply, database) $\mathcal{DB}$ for a schema $\mathcal{C}$ is a set of relations with constants as atomic values, and with one relation $r^{\mathcal{DB}}$ of arity n for each predicate symbol r of arity n in the alphabet $\mathcal{A}$. It is well known that a database can be seen as a first-order interpretation for the relation symbols in the schema: the relation $r^{\mathcal{DB}}$ is the interpretation of the predicate symbol r in $\mathcal{DB}$, in the sense that it contains the set of tuples that satisfy the predicate r in $\mathcal{DB}$. A database $\mathcal{DB}$ for a schema $\mathcal{C}$ is said to be *legal* if every constraints of $\mathcal{C}$ is satisfied by $\mathcal{DB}$. The notion of satisfaction depends on the type of constraints.

In our framework we consider the relational model with two kinds of constraints:

- *Key constraints*: given a relation r in the schema, a key constraint over r is expressed in the form $key(r) = \mathbf{A}$, where $\mathbf{A}$ is a set of attributes of r. Such a constraint is satisfied in a database $\mathcal{DB}$ if for each $t_1, t_2 \in r^{\mathcal{DB}}$ we have $t_1[\mathbf{A}] \neq t_2[\mathbf{A}]$, where $t[\mathbf{A}]$ is the projection of the tuple t over $\mathbf{A}$.
- *Foreign key constraints*: a foreign key constraint is a statement of the form $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$, where r_1, r_2 are relations, $\mathbf{A}$ is a sequence of distinct attributes of r_1, and $\mathbf{B}$ is a sequence formed by the distinct attributes forming the key of r_2. Such a constraint is satisfied in a database $\mathcal{DB}$ if for each tuple t_1 in $r_1^{\mathcal{DB}}$ there exists a tuple t_2 in $r_2^{\mathcal{DB}}$ such that $t_1[\mathbf{A}] = t_2[\mathbf{B}]$.

A *relational query* is a formula that specifies a set of tuples to be retrieved from a database. In this work, we restrict our analysis to the class of conjunctive queries. Formally, a *conjunctive query* (CQ) q of *arity* n is written in the form

$$q(x_1, \ldots, x_n) \leftarrow conj(x_1, \ldots, x_n, y_1, \ldots, y_m)$$

where: q belongs to a new alphabet $\mathcal{Q}$ (the alphabet of queries, that is disjoint from both Γ and $\mathcal{A}$); $conj(x_1, \ldots, x_n, y_1, \ldots, y_m)$ is a conjunction of atoms involving the variables $x_1, \ldots, x_n, y_1, \ldots, y_m$, and a set of constants from Γ; and the predicate symbols of the atoms are in $\mathcal{C}$.

The answer to a query q of arity n over a database $\mathcal{DB}$ for $\mathcal{G}$, denoted $q^{\mathcal{DB}}$, is the set of n-tuples of constants $(c_1, \ldots, c_n)$, such that, when substituting each c_i for x_i, the formula $\exists(y_1, \ldots, y_n).conj(x_1, \ldots, x_n, y_1, \ldots, y_m)$ evaluates to true in $\mathcal{DB}$. Note that the answer to q over $\mathcal{DB}$ is a relation whose arity is equal to the arity of the query q.

We now turn our attention to the notion of data integration system.

Definition 1. *A* data integration system $\mathcal{I}$ *is a triple* $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$*, where* $\mathcal{G}$ *is the* global schema*,* $\mathcal{S}$ *is the* source schema*, and* $\mathcal{M}_{\mathcal{G},\mathcal{S}}$ *is the* mapping *between* $\mathcal{G}$ *and* $\mathcal{S}$*.*

Now we describe the characteristics of the components of a data integration system in our approach. In particular, we specialize the general framework as follows:

- The *global schema* is expressed in the relational model with both key and foreign key constraints. We assume that in the global schema there is exactly one key constraint for each relation.
- The *source schema* is expressed in the relational model without integrity constraints. In other words, we conceive each source as a relation, and we consider the set of all relations as a unique schema, called source schema.
- The *mapping* $\mathcal{M}_{\mathcal{G},\mathcal{S}}$ is defined in the global-centric approach: to each relation r of $\mathcal{G}$ we associate a query $\rho(r)$ over the source schema. No limitation is posed on the language used to express queries in the mapping $\mathcal{M}_{\mathcal{G},\mathcal{S}}$.
- Queries over the global schema are *conjunctive queries*.

Example 1. An example of data integration system is $\mathcal{I}^1 = \langle \mathcal{G}^1, \mathcal{S}^1, \mathcal{M}^1_{\mathcal{G},\mathcal{S}} \rangle$ where $\mathcal{G}^1$ is constituted by the relation symbols $\mathsf{student}(Scode, Sname, Scity)$, $\mathsf{university}(Ucode, Uname)$, and $\mathsf{enrolled}(Scode, Ucode)$ and the constraints

$$\begin{aligned} key(\mathsf{student}) &= \{Scode\} \\ key(\mathsf{university}) &= \{Ucode\} \\ key(\mathsf{enrolled}) &= \{Scode, Ucode\} \end{aligned} \qquad \begin{aligned} \mathsf{enrolled}[Scode] &\subseteq \mathsf{student}[Scode] \\ \mathsf{enrolled}[Ucode] &\subseteq \mathsf{university}[Ucode] \end{aligned}$$

$\mathcal{S}^1$ consists of three sources. Source s_1, of arity 4, contains information about students with their code, name, city, and date of birth. Source s_2, of arity 2, contains codes and names of universities. Finally, Source s_3, of arity 2, contains information about enrollment of students in universities. The mapping $\mathcal{M}^1_{\mathcal{G},\mathcal{S}}$ is defined by

$$\begin{aligned} \rho(\mathsf{student}) &= \mathsf{st}(X, Y, Z) \leftarrow \mathsf{s}_1(X, Y, Z, W) \\ \rho(\mathsf{university}) &= \mathsf{un}(X, Y) \leftarrow \mathsf{s}_2(X, Y) \\ \rho(\mathsf{enrolled}) &= \mathsf{en}(X, W) \leftarrow \mathsf{s}_3(X, W) \end{aligned}$$

In order to define the semantics of a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$, we start from the data at the sources, and specify which are the data that satisfy the global schema. A *source database* $\mathcal{D}$ for $\mathcal{I}$ is constituted by one relation $r^{\mathcal{D}}$ for each source r in $\mathcal{S}$. We call *global database* for $\mathcal{I}$, or simply *database* for $\mathcal{I}$, any database for $\mathcal{G}$. A database $\mathcal{B}$ for $\mathcal{I}$ is said to be *legal* with respect to $\mathcal{D}$ if:

- $\mathcal{B}$ satisfies the integrity constraints of $\mathcal{G}$.
- $\mathcal{B}$ satisfies $\mathcal{M}_{\mathcal{G},\mathcal{S}}$ with respect to $\mathcal{D}$, i.e., for each relation r in $\mathcal{G}$, the set of tuples $r^{\mathcal{B}}$ that $\mathcal{B}$ assigns to r is a subset of the set of tuples $\rho(r)^{\mathcal{D}}$ computed by the associated query $\rho(r)$ over $\mathcal{D}$, i.e., $\rho(r)^{\mathcal{D}} \subseteq r^{\mathcal{B}}$.

Note that the above definition amounts to consider any view $\rho(r)$ as *sound*, which means that the data provided by the sources are not necessarily complete. Other assumptions on views are possible (see [14,18]). In particular, views may be *complete*, i.e., for each r in $\mathcal{G}$, we have $\rho(r)^{\mathcal{D}} \supseteq r^{\mathcal{B}}$, or *exact*, i.e., for each r in $\mathcal{G}$, we have $\rho(r)^{\mathcal{D}} = r^{\mathcal{B}}$. In this paper, we restrict our attention to sound

views only, which are typically considered the most natural in a data integration setting.

At this point, we are able to give the semantics of a data integration system, which is formally defined as follows.

Definition 2. *If $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$, and $\mathcal{D}$ is a source database for $\mathcal{I}$, the* semantics *of $\mathcal{I}$ w.r.t. $\mathcal{D}$, denoted $sem^{\mathcal{D}}(\mathcal{I})$, is the set of databases for $\mathcal{I}$ that are legal w.r.t. $\mathcal{D}$, i.e., that satisfy both the constraints of $\mathcal{G}$, and the mapping $\mathcal{M}_{\mathcal{G},\mathcal{S}}$ with respect to $\mathcal{D}$. If $sem^{\mathcal{D}}(\mathcal{I}) \neq \emptyset$, then $\mathcal{I}$ said to be* consistent *w.r.t. $\mathcal{D}$.*

By the above definition, it is clear that the semantics of a data integration systems is formulated in terms of a *set* of databases, rather than a single one. Indeed, as we will show in the sequel, the cardinality of $sem^{\mathcal{D}}(\mathcal{I})$ is in general greater than one. The impact of this property on query answering will be studied in the next section.

3 Query Answering in the Presence of Constraints

The ultimate goal of a data integration system is to answer queries posed by the user in terms of the global schema. Answering a query posed to a system representing a set of databases, is a complex task, as shown by the following example.

Example 2. Referring to Example 1, suppose to have the following source database $\mathcal{D}^1$:

$s_1^{\mathcal{D}^1}$:

12	*anne*	*florence*	21
15	*bill*	*oslo*	24

$s_2^{\mathcal{D}^1}$:

AF	*bocconi*
BN	*ucla*

$s_3^{\mathcal{D}^1}$:

12	*AF*
16	*BN*

Now, due to the integrity constraints in $\mathcal{G}_1$, 16 is the code of some student. Observe, however, that nothing is said by $\mathcal{D}^1$ about the name and the city of such student. Therefore, we must accept as legal all databases that differ in such attributes of the student with code 16. Note that this is a consequence of the assumption of having sound views. If we had exact or complete views, this situation would have lead to an inconsistency of the data integration system. Instead, when dealing with sound views, we can think of extending the data contained in the sources in order to satisfy the integrity constraint over the global schema. The fact that, in general, there are several possible ways to carry out such extension implies that there are several legal databases for the data integration systems.

Let us now turn our attention to the notion of answer to a query posed to the data integration system. In our setting, a query q to a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ is a conjunctive query, whose atoms have symbols in $\mathcal{G}$ as predicates. Our goal is to specify which are the tuples that form the answer to a certain query posed to $\mathcal{I}$. The task is complicated by the existence of several global databases which are legal for $\mathcal{I}$ with respect to a source database $\mathcal{D}$. In order to address this problem, we adopt the following approach: a tuple $(c_1, \ldots, c_n)$ is considered an answer to the query only if it is a *certain* answer, i.e., it satisfies the query in *every* database that belongs to the semantics of the data integration system.

Definition 3. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be a data integration system, let $\mathcal{D}$ be a source database for $\mathcal{I}$, and let q be a query of arity n to $\mathcal{I}$. The* set of certain answers *$q^{\mathcal{I},\mathcal{D}}$ to q with respect to $\mathcal{I}$ and $\mathcal{D}$ is the set of tuples $(c_1, \ldots, c_n)$ such that $(c_1, \ldots, c_n) \in q^{\mathcal{B}}$, for each $\mathcal{B} \in sem^{\mathcal{D}}(\mathcal{I})$.*

As mentioned, it is generally assumed that query answering is an easy task in the global-centric approach. Indeed, the most common technique for query answering in this approach is based on *unfolding*, i.e. substituting to each relation symbol r in the query its definition $\rho(r)$ in terms of the sources. We now show a simple unfolding strategy is not sufficient for providing all correct answers in the presence of integrity constraints.

Example 3. Referring again to Example 1, consider the query

$$\mathsf{q}(X) \leftarrow \mathsf{student}(X, Y, Z) \wedge \mathsf{enrolled}(X, W)$$

The correct answer to the query is $\{12, 16\}$, because, due to the integrity constraints in $\mathcal{G}_1$, we know that 16 appears in the first attribute of student in all the databases for $\mathcal{I}$ that are legal w.r.t. $\mathcal{D}_1$. However, we do not get this information from $\mathsf{s}_1^{\mathcal{D}_1}$, and, therefore, a simple unfolding strategy retrieves only the answer $\{12\}$ from $\mathcal{D}^1$, thus proving insufficient for query answering in this framework. Notice that, if the query asked for the student name instead of the student code (i.e., the head is $q(Y)$ instead of $q(X)$), then one could *not* make use of the dependencies to infer additional answers.

The above example shows that, in the presence of integrity constraints, even in the global-centric approach we have to deal with incomplete information during query processing.

4 General Description of the Approach

We present the general ideas that are at the basis of our method for query answering in data integration systems.

Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be a data integration system. In this paper we assume that, for each relation r of the global schema, the query $\rho(r)$ over the source schema that the mapping $\mathcal{M}_{\mathcal{G},\mathcal{S}}$ associates to r preserves the key constraint of r. This may require that $\rho(r)$ implements a suitable duplicate record elimination strategy that ensures that, for every source database $\mathcal{D}$ no pairs of tuples are extracted from $\mathcal{D}$ by $\rho(r)$ with the same value for the key of r. The problem of duplicate record elimination, and, more generally, of data cleaning, is a critical issues in data integration systems, however it is orthogonal to the problem addressed here. We refer to [25,26] for more details.

Let q be a query posed to $\mathcal{I}$, and $\mathcal{D}$ a source database for $\mathcal{I}$. We illustrate a naive method for computing the answer $q^{\mathcal{I},\mathcal{D}}$ to q w.r.t. $\mathcal{I}$ and $\mathcal{D}$. The naive computation of $q^{\mathcal{I},\mathcal{D}}$ proceeds as follows.

1. For each relation r of the global schema, we compute the relation $r^{\mathcal{D}}$ by evaluating the query $\rho(r)$ over the source database $\mathcal{D}$. The various relations so obtained form what we call the *retrieved global database* $ret(\mathcal{I}, \mathcal{D})$. Note

that, since we assume that $\rho(r)$ does not violate the key constraints, it follows that the retrieved global database satisfies all key constraints in $\mathcal{G}$.

2. If, additionally, the retrieved global database satisfies all foreign key constraints in $\mathcal{G}$, then we are basically done: we simply evaluate q over $ret(\mathcal{I}, \mathcal{D})$, and we obtain the answer to the query.
   Otherwise, based on the retrieved global database, we can build a database for $\mathcal{I}$ still satisfying the key constraints by suitably adding tuples to the relations of the global schema in such a way that also the foreign key constraints are satisfied.[1] Obviously, there are several possible ways to add tuples to the global relations.
   We may try to infer all the legal databases for $\mathcal{I}$ that are coherent with the retrieved global database, and we compute the tuples that satisfy the query q in all such legal databases. However, such a solution is not easy to pursue. Indeed, the direct way to implement it, i.e., building all the legal databases for $\mathcal{I}$ that are coherent with the retrieved global database, is not feasible: in general, there is an infinite number of legal databases that are coherent with the retrieved global database. Fortunately, starting from the retrieved global database, we can build another database, that we call *canonical*, that has the interesting property of faithfully representing all legal databases that are coherent with the retrieved global database.

Let us start by showing how to build the canonical database. First of all, we define the domain of such database, which we denote $HD(\mathcal{D})$, as follows. Based on the global schema $\mathcal{G}$ of $\mathcal{I}$, we introduce the following set of function symbols:

$$HT(\mathcal{G}) = \{f_{r,i} \mid r \in \mathcal{G} \text{ and } i \leq arity(r) \text{ and } i \notin key(r)\}$$

Thus, each $f_{r,i}$ is a function symbol, and such a function symbol has the same arity as the number of attributes of $key(r)$, i.e., $arity(f_{r,i}) = arity(key(r))$. From $\mathcal{D}$, we now define the domain $HD(\mathcal{D})$ as the smallest set satisfying the following conditions:

- $\Gamma \subseteq HD(\mathcal{D})$,
- if $\alpha_1, \ldots, \alpha_k \in HD(\mathcal{D})$, and $f_{R,i} \in HT(\mathcal{G})$, with $arity(f_{R,i}) = k$, then $f_{R,i}(\alpha_1, \ldots, \alpha_k) \in HD(\mathcal{D})$.

Now, given the retrieved global database $ret(\mathcal{I}, \mathcal{D})$, we obtain the canonical database $can(\mathcal{I}, \mathcal{D})$ over the domain $HD(\mathcal{D})$ by repeatedly applying the following rule:

> **if** $(x_1, \ldots, x_h) \in r[\mathbf{A}]$, and the foreign key constraint $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$ is in $\mathcal{G}$,
> **then** insert the tuple t in r_2 such that

[1] Note that, since views are sound, i.e., they return a *subset* of the tuples in a global relation, we cannot conclude that the data integration system violates the foreign key constraints of $\mathcal{G}$. Indeed, it may be the case that the tuples needed to satisfy such constraints are not part of the retrieved subsets.

- $t[\mathbf{B}] = (x_1, \ldots, x_h)$, and
- for each $i \leq \mathit{arity}(r_2)$ not in $\mathbf{B}$, $t[i] = f_{r_2,i}(x_1, \ldots, x_h)$.

Observe that $\mathit{can}(\mathcal{I}, \mathcal{D})$ is indeed a database over the domain $\mathit{HD}(\mathcal{D})$, and that, in general, $\mathit{can}(\mathcal{I}, \mathcal{D})$ is infinite. However, it enjoys important properties, as shown below. The first property is related to the satisfaction of the constraints of $\mathcal{G}$.

Theorem 1. *If $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$, and $\mathcal{D}$ is a source database for $\mathcal{I}$, then $\mathit{can}(\mathcal{I}, \mathcal{D})$ does not violate any foreign key constraint in $\mathcal{G}$.*

Proof. Suppose by contradiction that the foreign key constraint $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$ is violated in $\mathit{can}(\mathcal{I}, \mathcal{D})$. This implies that there is a tuple t in r_1 such that for no tuple t' in r_2 $t'[\mathbf{B}] = t[\mathbf{A}]$. But this would imply that we can apply the rule and insert a new tuple t'' in r_2 such that $t''[\mathbf{B}] = t[\mathbf{A}]$, and for each $i \leq \mathit{arity}(r_2)$ not in $\mathbf{B}$, $t'[i] = f_{r_2,i}(t[\mathbf{A}])$. But this contradicts the assumption.

We now show that there exists a legal database for $\mathcal{I}$ w.r.t. $\mathcal{D}$ (called $\mathit{can}^-(\mathcal{I}, \mathcal{D})$), which implies that $\mathcal{I}$ is consistent w.r.t. $\mathcal{D}$, if and only if $\mathit{ret}(\mathcal{I}, \mathcal{D})$ does not violate any key constraint in $\mathcal{G}$.

Theorem 2. *If $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$, and $\mathcal{D}$ is a source database for $\mathcal{I}$, then there exists a legal database for $\mathcal{I}$ w.r.t. $\mathcal{D}$ if and only if $\mathit{ret}(\mathcal{I}, \mathcal{D})$ does not violate any key constraint in $\mathcal{G}$.*

Proof. It is immediate to see that if $\mathit{ret}(\mathcal{I}, \mathcal{D})$ violates some key constraint in $\mathcal{G}$, then no legal database exists for $\mathcal{I}$ w.r.t. $\mathcal{D}$.

It remains to show that, if $\mathit{ret}(\mathcal{I}, \mathcal{D})$ does not violate any key constraint in $\mathcal{G}$, then there exists a legal database $\mathit{can}^-(\mathcal{I}, \mathcal{D})$ for $\mathcal{I}$ w.r.t. $\mathcal{D}$, which implies that $\mathcal{I}$ is consistent w.r.t. $\mathcal{D}$. We construct $\mathit{can}^-(\mathcal{I}, \mathcal{D})$ from $\mathcal{G}$ and $\mathcal{D}$ similarly to $\mathit{can}(\mathcal{I}, \mathcal{D})$, with the only difference that we use the rule:

If $(x_1, \ldots, x_h) \in r_1[\mathbf{A}]$, $(x_1, \ldots, x_h) \notin r_2[\mathbf{B}]$, and the foreign key constraint $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$ is in $\mathcal{G}$,
then insert the tuple t in r_2 such that
- $t[\mathbf{B}] = (x_1, \ldots, x_h)$, and
- for each $i \leq \mathit{arity}(r_2)$ different from $\mathbf{B}$, $t[i] = f_{r_2,i}(x)$.

It is easy to see that $\mathit{can}^-(\mathcal{I}, \mathcal{D}) \subseteq \mathit{can}(\mathcal{I}, \mathcal{D})$. To show that $\mathit{can}^-(\mathcal{I}, \mathcal{D})$ is indeed a legal database for $\mathcal{I}$ w.r.t. $\mathcal{D}$, we consider key and foreign key constraints separately. As for key constraints, it is easy to see that the tuples inserted during the process of computing $\mathit{can}^-(\mathcal{I}, \mathcal{D})$ cannot violate any key constraints of $\mathcal{G}$. Indeed, in computing $\mathit{can}^-(\mathcal{I}, \mathcal{D})$, we insert a tuple into a relation r only when the key component of that tuple is not already present in r. Since $\mathit{ret}(\mathcal{I}, \mathcal{D})$ does not violate any key constraint in $\mathcal{G}$, it follows that no key constraint of $\mathcal{G}$ is violated in $\mathit{can}^-(\mathcal{I}, \mathcal{D})$. As for foreign key constraints, suppose by contradiction that the foreign key constraint $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$ is violated in $\mathit{can}^-(\mathcal{I}, \mathcal{D})$. This implies that there is a tuple t in r_1 such that for no tuple t' in r_2 $t'[\mathbf{B}] = t[\mathbf{A}]$. But this would imply that we can apply the above rule and insert a new tuple t'' in r_2 such that $t''[\mathbf{B}] = t[\mathbf{A}]$, and for each $i \leq \mathit{arity}(r_2)$ not in $\mathbf{B}$, $t'[i] = f_{r_2,i}(t[\mathbf{A}])$. But this contradicts the assumption.

The canonical database $can(\mathcal{I}, \mathcal{D})$ has the interesting property of faithfully representing all legal databases that are coherent with the retrieved global database $ret(\mathcal{I}, \mathcal{D})$.

Theorem 3. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$, let $\mathcal{D}$ be a source database for $\mathcal{I}$, and let $\mathcal{B}$ be a legal database for $\mathcal{I}$ w.r.t. $\mathcal{D}$. There is a total function ψ from $HD(\mathcal{D})$ to Γ such that, for each relation r of arity n in $\mathcal{G}$, and each tuple $(c_1, \ldots, c_n)$ constituted by elements in $HD(\mathcal{D})$, if $(c_1, \ldots, c_n) \in r^{can(\mathcal{I},\mathcal{D})}$, then $(\psi(c_1), \ldots, \psi(c_n)) \in r^{\mathcal{B}}$.*

Proof. We define the function ψ from $HD(\mathcal{D})$ to Γ inductively, and we simultaneously show that for each relation r of arity n in $\mathcal{G}$, and each tuple $(c_1, \ldots, c_n)$ constituted by elements in $HD(\mathcal{D})$, if $(c_1, \ldots, c_n) \in r^{can(\mathcal{I},\mathcal{D})}$, then $(\psi(c_1), \ldots, \psi(c_n)) \in r^{\mathcal{B}}$.

We proceed by induction on the application of the rule used during the construction of $can(\mathcal{I}, \mathcal{D})$. As a base step, the function ψ maps each constant in $ret(\mathcal{I}, \mathcal{D})$ into itself. It follows that, for each r, if $c_1, \ldots, c_n$ are constants, and $(c_1, \ldots, c_n) \in r^{ret(\mathcal{I},\mathcal{D})}$, then it is obvious that both $(c_1, \ldots, c_n) \in r^{can(\mathcal{I},\mathcal{D})}$, and $(\psi(c_1), \ldots, \psi(c_n)) \in r^{\mathcal{B}}$.

Inductive step. Suppose, without loss of generality, that in the application of the rule, we are inserting the tuple $(\alpha, f_{r,i_1}(\alpha), f_{r,i_2}(\alpha))$ in $r^{can(\mathcal{I},\mathcal{D})}$ where r has arity 3, $key(r) = \{1\}$, and the tuple is inserted in $r^{can(\mathcal{I},\mathcal{D})}$ because of the foreign key constraint $w[j] \subseteq r[1]$. Since we are applying the rule because of the constraint $w[j] \subseteq r[1]$, we have that there is a tuple t in $w^{can(\mathcal{I},\mathcal{D})}$ such that $t[j] = \alpha$. For the induction hypothesis, there is a β in Γ such that $\psi(\alpha) = \beta$, and there is a tuple $t' \in w^{\mathcal{B}}$ such that for each i, $t'[i] = \psi(t[i])$, and $t'[j] = \psi(\alpha) = \beta$. Because of the constraint $w[j] \subseteq r[1]$, and because $\mathcal{B}$ is legal, there is one and only one tuple (β, γ, δ) in $r^{\mathcal{B}}$ (since 1 is a key of r, β appears once in $r^{\mathcal{B}}[1]$). Then, we set $\psi(f_{r,i_1}(\alpha)) = \gamma$, $\psi(f_{r,i_2}(\alpha)) = \delta$, and we can conclude that $(\psi(\alpha), \psi(f_{r,i_1}(\alpha)), \psi(f_{r,i_2}(\alpha))) \in r^{\mathcal{B}}$.

Finally, we show that, if $\mathcal{I}$ is consistent w.r.t. $\mathcal{D}$, then a tuple t of constants is in $q^{\mathcal{I},\mathcal{D}}$ if and only if t is in the answer to q over the database $can(\mathcal{I}, \mathcal{D})$.

Theorem 4. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$, let q be a query posed to $\mathcal{I}$, $\mathcal{D}$ a source database for $\mathcal{I}$, and t a tuple of constants of the same arity as q. If $\mathcal{I}$ is consistent w.r.t. $\mathcal{D}$, then $t \in q^{\mathcal{I},\mathcal{D}}$ if and only if t is in the answer to q over $can(\mathcal{I}, \mathcal{D})$.*

Proof. For the "if" direction, we show that if t is in the answer to q over $can(\mathcal{I}, \mathcal{D})$, then $t \in q^{\mathcal{I},\mathcal{D}}$. Indeed, consider any $\mathcal{B}$ that is a legal database for $\mathcal{I}$ w.r.t. $\mathcal{D}$. By theorem 3, there is a total function ψ from $HD(\mathcal{D})$ to Γ such that, for each relation r of arity n in $\mathcal{G}$, and each tuple $(c_1, \ldots, c_n)$ constituted by elements in $HD(\mathcal{D})$, if $(c_1, \ldots, c_n) \in r^{can(\mathcal{I},\mathcal{D})}$, then $(\psi(c_1), \ldots, \psi(c_n)) \in r^{\mathcal{B}}$. The fact that t is in the answer to q over $can(\mathcal{I}, \mathcal{D})$ means that there is an assignment α from the variables of q to objects in $HD(\mathcal{D})$ such that all atoms of q are true with respect to the assignment. It is easy to see that the assignment $\alpha \cdot \psi$ can be used to show that t is in the answer to q over $\mathcal{B}$.

As for the "only-if" direction, first note that, by hypothesis $\mathcal{I}$ is consistent w.r.t. $\mathcal{D}$, and, therefore, by theorem 2, $ret(\mathcal{I}, \mathcal{D})$ does not violate any key constraint in $\mathcal{G}$, which implies that $can^-(\mathcal{I}, \mathcal{D})$ is a legal database for $\mathcal{I}$ w.r.t. $\mathcal{D}$. Now, since $can^-(\mathcal{I}, \mathcal{D}) \subseteq can(\mathcal{I}, \mathcal{D})$, and since q is a conjunctive query, the fact that t is not in the answer to q over $can(\mathcal{I}, \mathcal{D})$ implies that t is not in the answer of q over $can^-(\mathcal{I}, \mathcal{D})$. Therefore, we can conclude that $t \notin q^{\mathcal{I},\mathcal{D}}$.

Based on the above results, we can conclude that $can(\mathcal{I}, \mathcal{D})$ is the right abstraction for answering queries posed to the data integration system. In the next section we show that, in processing a query q posed to the data integration system, we can find the answers to q over $can(\mathcal{I}, \mathcal{D})$ without actually building $can(\mathcal{I}, \mathcal{D})$.

5 Query Reformulation

The naive computation described in the previous section is impractical, because it requires to build the canonical database, which is in general infinite. In order to overcome the problem, we have devised an algorithm, whose main ideas are as follows.

1. First, as we said in the previous section, we assume that, for each relation r of the global schema, the query $\rho(r)$ over the source schema that the mapping $\mathcal{M}_{\mathcal{G},\mathcal{S}}$ associates to r preserve the key constraint of r.
2. Instead of referring explicitly to the canonical database for query answering, we transform the original query q into a new query $exp_{\mathcal{G}}(q)$ over the global schema, called the *expansion of q w.r.t. $\mathcal{G}$*, such that the answer to $exp_{\mathcal{G}}(q)$ over the retrieved global database is equal to the answer to q over the canonical database.
3. In order to avoid building the retrieved global database, we do not evaluate $exp_{\mathcal{G}}(q)$ on the retrieved global database. Instead, we unfold $exp_{\mathcal{G}}(q)$ to a new query, called $unf_{\mathcal{M}_{\mathcal{G},\mathcal{S}}}(exp_{\mathcal{G}}(q))$, over the source relations on the basis of $\mathcal{M}_{\mathcal{G},\mathcal{S}}$, and we use the unfolded query $unf_{\mathcal{M}_{\mathcal{G},\mathcal{S}}}(exp_{\mathcal{G}}(q))$ to access the sources.

We refer to steps 1 and 2 as the "query reformulation" step. Step 3 is called the "source access". In the rest of the section we discuss the first two steps.

Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be a data integration system, let $\mathcal{D}$ be a source database, and let q be a query over the global schema $\mathcal{G}$. We show how to reformulate the original query q into a new query $exp_{\mathcal{G}}(q)$ over the global schema, called the expansion of q w.r.t. $\mathcal{G}$, such that the answer to $exp_{\mathcal{G}}(q)$ over the (virtual) retrieved global database is equal to the answer to q over the canonical database.

The basic idea to do so is that the constraints in $\mathcal{G}$ can be captured by a suitable *logic program* $\mathcal{P}_{\mathcal{G}}$. To build $\mathcal{P}_{\mathcal{G}}$, we introduce a new relation p' (called primed relation) for each relation p in $\mathcal{G}$. Then, from the semantics of $\mathcal{G}$ we devise the following rules for $\mathcal{P}_{\mathcal{G}}$ (expressed in Logic Programming notation [27]):

– for each relation r, we have:

$$r'(X_1, \ldots, X_n) \leftarrow r(X_1, \ldots, X_n)$$

– for each foreign key constraint $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$ in $\mathcal{G}$, where $\mathbf{A}$ and $\mathbf{B}$ are sets of attributes and $\mathbf{B}$ is a key for r_2 (assuming for simplicity that the attributes involved in the foreign key are the first h):

$$r'_2(X_1, \ldots, X_h, f_{h+1}(X_1, \ldots, X_h), \ldots, f_n(X_1, \ldots, X_h)) \leftarrow r'_1(X_1, \ldots, X_h, \ldots, X_m)$$

where f_i are fresh function symbols, called Skolem functions.

We can use the logic program $\mathcal{P}_\mathcal{G}$ to generate the query $exp_\mathcal{G}(q)$ associated to the original query q. This is done as follows.

1. First, we rewrite q by substituting each relation symbol r in the body $body(q)$ of q with a new symbol r'. We denote by q' the resulting query. In the following we call "primed atom" every atom whose relation symbol is primed, i.e., it has the form r' for some r.
2. Then we build a *partial evaluation tree* for q', i.e., a tree having each node labeled by a conjunctive query g, with one of the atoms in $body(g)$ marked as "*selected*", obtained as follows.
   (a) The root is labeled by q', and has one (primed) atom (for example the first in left-to-right order) marked as selected.
   (b) Except if condition (2c) below is satisfied, a node, labeled by a query g having a "selected" atom α, has one child for each rule ϕ in $\mathcal{P}_\mathcal{G}$ such that there exists a most general unifier[2] $mgu(\alpha, head(\phi))$ between the atom α and the head $head(\phi)$ of the rule ϕ, such that the distinguished variables are not assigned to terms involving Skolem functions. Each of such children has the following properties:
      – it is labeled by the query obtained from g by replacing the atom α with $body(\phi)$ and by substituting the variables with $mgu(\alpha, head(\phi))$;
      – it has as marked "selected" one of the primed atoms (for example the first in left-to-right order).
   (c) If a node d is labeled by a query g, and there exist a predecessor d' of d labeled by a query g' and a substitution θ of the variables of g' that makes g' equal to g, then d has a single child, which is labeled by the empty query (a query whose body is false).
3. Finally we return as result the query $exp_\mathcal{G}(q)$ formed as the union of all non-empty queries in the leaves of the partial evaluation tree.

Theorem 5 (Termination). *The algorithm above always terminates.*

[2] We recall that given two atoms α and β the most general unifier $mgu(\alpha, \beta)$ is a most general substitution for the variables in α and β that makes α and β equal [27].

Proof. The termination of the algorithm follows directly from the following observations:

- The queries in all nodes on the tree have exactly the same number of atoms as the original query q. This is an immediate consequence of the fact that for rule ϕ in $\mathcal{P}_\mathcal{G}$, $body(\phi)$ is formed by exactly one atom.
- Condition (2c) guarantees a finite bound on the nesting of Skolem functions in the queries in the nodes.

As a consequence, the number of queries along each branch of the partial evaluation tree must be finite, hence the thesis holds.

Our goal now is to show that if $\mathcal{I}$ is consistent w.r.t. $\mathcal{D}$, then $t \in q^{\mathcal{I},\mathcal{D}}$ if and only if t is in the answer to $unf_{\mathcal{M}_{\mathcal{G},\mathcal{S}}}(exp_\mathcal{G}(q))$ over $\mathcal{D}$. We will prove such result by applying results from the logic programming theory [27] and, in particular, results on the partial evaluation of logic programs [28]. We first observe that $ret(\mathcal{I},\mathcal{D})$ can be seen as a (finite) set of ground facts in logic programming terms. We proceed by proving a series of lemmas, each dealing with a particular aspect of the proof. The relationship between the *logic program* $\mathcal{P}_\mathcal{G}$ and the canonical database of the data integration system $\mathcal{I}$ is characterized by the following lemma.

Lemma 1. *Up to the renaming of each relation symbol r by the corresponding primed symbol r', $can(\mathcal{I},\mathcal{D})$ coincides with the minimal model of $\mathcal{P}_\mathcal{G} \cup ret(\mathcal{I},\mathcal{D})$.*

Proof. The thesis is an immediate consequence of the semantics of $can(\mathcal{I},\mathcal{D})$ and $\mathcal{P}_\mathcal{G} \cup ret(\mathcal{I},\mathcal{D})$ [27].

Next we focus on SLD-refutation. We observe that, since the query q is a conjunctive query, the query q' is a union of conjunctive queries:

$$q'(X_1,\ldots,X_n) \leftarrow disj_1 \vee \cdots \vee disj_k$$

An SLD-refutation for $\mathcal{P}_\mathcal{G} \cup ret(\mathcal{I},\mathcal{D}) \cup \neg q'(t)$ is defined as an SLD-refutation for $\mathcal{P}_\mathcal{G} \cup \mathcal{P}'_q \cup ret(\mathcal{I},\mathcal{D}) \cup \neg q'(t)$, where $\mathcal{P}'_q$ is constituted by the rules:

$$\begin{aligned} q(X_1,\ldots,X_n) &\leftarrow disj_1 \\ &\cdots \\ q(X_1,\ldots,X_n) &\leftarrow disj_k \end{aligned}$$

one for each disjunct $disj_i$ of the query q' (see [27]).

Lemma 2. *$q'(t)$ is true in the minimal model of $\mathcal{P}_\mathcal{G} \cup ret(\mathcal{I},\mathcal{D})$ iff there is an SLD-refutation for $\mathcal{P}_\mathcal{G} \cup ret(\mathcal{I},\mathcal{D}) \cup \{\neg q'(t)\}$.*

Proof. The thesis follows directly from the soundness and completeness of SLD-resolution for definite logic programs, see e.g., [27].

Next, let us consider a slight modification of the algorithm above where Condition (2c) is replaced by the following one:

If a node d that is labeled by a query g and there exists a predecessor d' of d labeled by a query g' and a substitution θ of the variables of g' that makes g' equal to g, then d has a single child, which is labeled by g itself but without any atom marked as selected.

Let us call $exp^{-}_{\mathcal{G}}(q)$ the query obtained from such a modified algorithm. For $exp^{-}_{\mathcal{G}}(q)$, we have the following result.

Lemma 3. *$\mathcal{P}_{\mathcal{G}} \cup ret(\mathcal{I},\mathcal{D}) \cup \{\neg q'(t)\}$ has an SLD-refutation iff $ret(\mathcal{I},\mathcal{D}) \cup \{\neg exp^{-}_{\mathcal{G}}(q)(t)\}$ has an SLD-refutation.*

Proof. It is easy to see that the modified algorithm generates a so-called partial evaluation [28] of the program $\mathcal{P}_{\mathcal{G}}$ w.r.t. the query q'. From the results in [28] on soundness and completeness of partial evaluation of logic programs, the thesis follows.

Lemma 4. *$\mathcal{P}_{\mathcal{G}} \cup ret(\mathcal{I},\mathcal{D}) \cup \{\neg q'(t)\}$ has an SLD-refutation iff $ret(\mathcal{I},\mathcal{D}) \cup \{\neg exp_{\mathcal{G}}(q)(t)\}$ has an SLD-refutation.*

Proof. The difference between $exp^{-}_{\mathcal{G}}(q)$ and $exp_{\mathcal{G}}(q)$ is that in $exp_{\mathcal{G}}(q)$ we drop the disjuncts coming from those nodes labeled by a query g such that there exists a query g' and a substitution θ of the variables of g' that makes g' equal to g. Next we show that, in doing this we do not loose any potential SLD-refutation of $\mathcal{P}_{\mathcal{G}} \cup ret(\mathcal{I},\mathcal{D}) \cup \{\neg q'(t)\}$.

Suppose that the shortest (possibly the only one) SLD-refutation for $\mathcal{P}_{\mathcal{G}} \cup ret(\mathcal{I},\mathcal{D}) \cup \{\neg q'(t)\}$ goes through a node labeled by one such g. Let us say the length of the SLD-refutation is n, and that node labeled by g is the k-th node along the SLD-refutation. From such SLD-refutation we get an SLD-refutation for $\mathcal{P}_{\mathcal{G}} \cup ret(\mathcal{I},\mathcal{D}) \cup \{\neg g(t)\}$ of length $n-k$. Observe that, by the so called Lifting Lemma [27], such an SLD-refutation is also an SLD-refutation for $\mathcal{P}_{\mathcal{G}} \cup ret(\mathcal{I},\mathcal{D}) \cup \{\neg g'(t)\}$. Hence there exists an SLD-refutation for which occurs in a node of the SLD-refutation for $\mathcal{P}_{\mathcal{G}} \cup ret(\mathcal{I},\mathcal{D}) \cup \{\neg q(t)\}$ that is shorter than n, which leads to contradiction. It follows that for each SLD-refutation for $\mathcal{P}_{\mathcal{G}} \cup ret(\mathcal{I},\mathcal{D}) \cup \{\neg q'(t)\}$ going through a node satisfying Condition (2c) there is also another (a shorter one in fact) that does not go through that node. Hence we may drop from the partial evaluation $exp^{-}_{\mathcal{G}}(q)$ all the conjuncts involving such nodes, thus getting $exp_{\mathcal{G}}(q)$ without loosing any SLD-refutation for the original query.

Finally, we observe that, since $exp_{\mathcal{G}}(q)$ does not involve any prime atom, the rules in $\mathcal{P}_{\mathcal{G}}$ cannot be applied along an SLD-refutation for $\mathcal{P}_{\mathcal{G}} \cup ret(\mathcal{I},\mathcal{D}) \cup \{\neg exp_{\mathcal{G}}(q)(t)\}$. Hence every SLD-refutation for $\mathcal{P}_{\mathcal{G}} \cup ret(\mathcal{I},\mathcal{D}) \cup \{\neg exp_{\mathcal{G}}(q)(t)\}$ is also an SLD-refutation for $\mathcal{P}_{\mathcal{G}} \cup ret(\mathcal{I},\mathcal{D}) \cup \{\neg exp_{\mathcal{G}}(q)(t)\}$.

With this lemma in place we can finally present our main theorem.

Theorem 6 (Soundness and Completeness). *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$, let q be a query posed to $\mathcal{I}$, $\mathcal{D}$ a source database for $\mathcal{I}$, and t a tuple of constants of the same arity as q. If $\mathcal{I}$ is consistent w.r.t. $\mathcal{D}$, then $t \in q^{\mathcal{I},\mathcal{D}}$ if and only if t is in the answer to $unf_{\mathcal{M}_{\mathcal{G},\mathcal{S}}}(exp_{\mathcal{G}}(q))$ over $\mathcal{D}$.*

Proof. By Lemma 1, Lemma 2, Lemma 4, we have that $q(t)$ is true in $can(\mathcal{I}, \mathcal{D})$ iff $ret(\mathcal{I}, \mathcal{D}) \cup \{\neg exp_{\mathcal{G}}(q)(t)\}$ has an SLD-refutation. That is by, again applying Lemma 2, $q(t)$ is true in $can(\mathcal{I}, \mathcal{D})$ iff t is in the answer to $exp_{\mathcal{G}}(q)$ over $ret(\mathcal{I}, \mathcal{D})$, i.e., by the semantics of $ret(\mathcal{I}, \mathcal{D})$, iff t is in the answer to $unf_{\mathcal{M}_{\mathcal{G},\mathcal{S}}}(exp_{\mathcal{G}}(q))$ over $\mathcal{D}$.

With regard to the characterization of the computational complexity of the algorithm, we observe that the number of disjuncts in $exp_{\mathcal{G}}(q)$ can be exponential in the number of rules in the logic program $\mathcal{P}_{\mathcal{G}}$ (and therefore in the size of the global schema $\mathcal{G}$), and in the number of variables in the original query q. Note, however, that this bound is independent of the size of $\mathcal{D}$, i.e., the size of data at the sources. We remind the reader that the evaluation of a union of conjunctive queries can be done in time polynomial with respect to the size of the data. Since $exp_{\mathcal{G}}(q)$ is a union of conjunctive queries, as the queries associated by $\mathcal{M}_{\mathcal{G},\mathcal{S}}$ to the elements of $\mathcal{G}$ are, then evaluating $unf_{\mathcal{M}_{\mathcal{G},\mathcal{S}}}(exp_{\mathcal{G}}(q))$ over $\mathcal{D}$ is also polynomial in the size of the data at the sources. It follows that our query answering algorithm is polynomial with respect to data complexity.

The following example illustrates the application of the expansion algorithm in a simple case.

Example 4. Suppose we have the following relations in the global schema $\mathcal{G}$ of a data integration system:

$$\begin{aligned} &\mathsf{person}(Pcode, Age, CityOfBirth)\\ &\mathsf{student}(Scode, University)\\ &\mathsf{city}(Name, Major) \end{aligned}$$

with the following integrity constraints:

$$\begin{aligned} key(\mathsf{person}) &= \{Pcode\} & \mathsf{person}[CityOfBirth] &\subseteq \mathsf{city}[Name]\\ key(\mathsf{student}) &= \{Scode\} & \mathsf{city}[Major] &\subseteq \mathsf{person}[PCode]\\ key(\mathsf{city}) &= \{Name\} & \mathsf{student}[SCode] &\subseteq \mathsf{person}[PCode] \end{aligned}$$

The logic program $\mathcal{P}_{\mathcal{G}}$ makes use of the predicates $\mathsf{person}'/3$, $\mathsf{student}'/1$, $\mathsf{city}'/2$ and constitutes of the following rules:

$$\begin{aligned} \mathsf{person}'(X,Y,Z) &\leftarrow \mathsf{person}(X,Y,Z) & \mathsf{city}'(X, f_1(X)) &\leftarrow \mathsf{person}'(Y,Z,X)\\ \mathsf{student}'(X,Y) &\leftarrow \mathsf{student}(X,Y) & \mathsf{person}'(Y, f_2(Y), f_3(Y)) &\leftarrow \mathsf{city}'(X,Y)\\ \mathsf{city}'(X,Y) &\leftarrow \mathsf{city}(X,Y) & \mathsf{person}'(X, f_4(X), f_5(X)) &\leftarrow \mathsf{student}'(X,Y) \end{aligned}$$

Suppose the user query is $\mathsf{q}(X) \leftarrow \mathsf{person}(X,Y,Z)$.

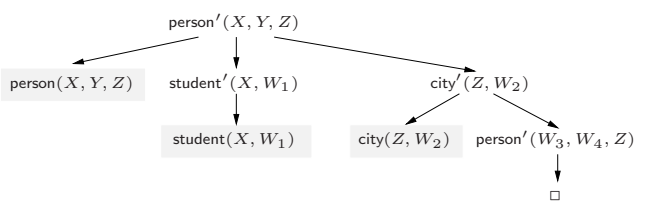

Fig. 1. Partial evaluation tree for the query of Example 4

The partial evaluation tree of q is shown in Figure 1. Note that in the rightmost branch, Condition (2c) is verified and hence the evaluation stops, producing the empty clause $\Box$. This prevents the evaluation process to get into an infinite branch. The new variables W_1, W_2, and W_3 are introduced in order to avoid variable clashes when performing unification. The non-empty leaves, shaded in the figure, provide the following expansion $\mathsf{q}' = exp_{\mathcal{G}}(\mathsf{q})$ of the query q:

$$\begin{aligned} \mathsf{q}'(X) &\leftarrow \mathsf{person}(X, Y, Z) \\ \mathsf{q}'(X) &\leftarrow \mathsf{student}(X, W_1) \\ \mathsf{q}'(W_2) &\leftarrow \mathsf{city}(Z, W_2) \end{aligned}$$

Intuitively, we see that the expanded query searches for codes of persons not only in the relation person, but also in student and city, where, due to the integrity constraints, it is known that codes of persons are stored.

6 Conclusions

While it is a common opinion that query processing is an easy task in the global-centric approach to data integration, we have shown the surprising result that, when the global schema contains integrity constraints, even of simple forms, query processing becomes more difficult. The difficulties basically arise because of the need of dealing with incomplete information, similarly to the case of the source-centric approach to data integration. We have studied the case of global schemas expressed in the relational model with key and foreign key constraints, and we have presented techniques for effectively answering queries posed to the data integration system in this case.

As future work, we aim at considering more forms of integrity constraints in the global schema, with the goal of modifying the algorithm described in this paper in order to take into account the new classes of constraints during query processing.

References

1. Batini, C., Lenzerini, M., Navathe, S. B.: A comparative analysis of methodologies for database schema integration. ACM Computing Surveys **18** (1986) 323–364 262
2. Sheth, A. P., Larson, J. A.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys **22** (1990) 183–236 262
3. Thomas, G., Thompson, G. R., Chung, C. W., Barkmeyer, E., Carter, F., Templeton, M., Fox, S., Hartman, B.: Heterogeneous distributed database systems for production use. ACM Computing Surveys **22** (1990) 237–266 262
4. Litwin, W., Mark, L., Roussopoulos, N.: Interoperability of multiple autonomous databases. ACM Computing Surveys **22** (1990) 267–293 262
5. Catarci, T., Lenzerini, M.: Representing and using interschema knowledge in cooperative information systems. J. of Intelligent and Cooperative Information Systems **2** (1993) 375–398 262
6. Hull, R.: Managing semantic heterogeneity in databases: A theoretical perspective. In: Proc. of PODS'97. (1997) 262

7. Halevy, A. Y.: Answering queries using views: A survey. VLDB Journal **10** (2001) 270–294 262, 263
8. Ullman, J. D.: Information integration using logical views. In: Proc. of ICDT'97. Volume 1186 of LNCS., Springer (1997) 19–40 262, 263
9. Li, C., Chang, E.: Query planning with limited source capabilities. In: Proc. of ICDE 2000. (2000) 401–412 263
10. Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J. D., Vassalos, V., Widom, J.: The TSIMMIS approach to mediation: Data models and languages. J. of Intelligent Information Systems **8** (1997) 117–132 263
11. Anthony Tomasic, Louiqa Raschid, P. V.: Scaling access to heterogeneous data sources with DISCO. IEEE Trans. on Knowledge and Data Engineering **10** (1998) 808–823 263
12. Goh, C. H., Bressan, S., Madnick, S. E., Siegel, M. D.: Context interchange: New features and formalisms for the intelligent integration of information. ACM Trans. on Information Systems **17** (1999) 270–293 263
13. Kirk, T., Levy, A. Y., Sagiv, Y., Srivastava, D.: The Information Manifold. In: Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments. (1995) 85–91 263
14. Abiteboul, S., Duschka, O.: Complexity of answering queries using materialized views. In: Proc. of PODS'98. (1998) 254–265 263, 266
15. Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., Rosati, R.: Data integration in data warehousing. Int. J. of Cooperative Information Systems **10** (2001) 237–271 263, 264
16. Calì, A., De Giacomo, G., Lenzerini, M.: Models for information integration: Turning local-as-view into global-as-view. In: Proc. of Int. Workshop on Foundations of Models for Information Integration (10th Workshop in the series Foundations of Models and Languages for Data and Objects). (2001) 263
17. Gryz, J.: Query folding with inclusion dependencies. In: Proc. of ICDE'98. (1998) 126–133 263
18. Grahne, G., Mendelzon, A. O.: Tableau techniques for querying information sources through global schemas. In: Proc. of ICDT'99. Volume 1540 of LNCS., Springer (1999) 332–347 263, 266
19. Calvanese, D., De Giacomo, G., Lenzerini, M., Vardi, M. Y.: Query processing using views for regular path queries with inverse. In: Proc. of PODS 2000. (2000) 58–66 263
20. van der Meyden, R.: Logical approaches to incomplete information. In Chomicki, J., Saake, G., eds.: Logics for Databases and Information Systems. Kluwer Academic Publisher (1998) 307–356 263
21. Fernandez, M. F., Florescu, D., Levy, A., Suciu, D.: Verifying integrity constraints on web-sites. In: Proc. of IJCAI'99. (1999) 614–619 264
22. Fernandez, M. F., Florescu, D., Kang, J., Levy, A. Y., Suciu, D.: Catching the boat with strudel: Experiences with a web-site management system. In: Proc. of ACM SIGMOD. (1998) 414–425 264
23. Carey, M. J., Haas, L. M., Schwarz, P. M., Arya, M., Cody, W. F., Fagin, R., Flickner, M., Luniewski, A., Niblack, W., Petkovic, D., Thomas, J., Williams, J. H., Wimmers, E. L.: Towards heterogeneous multimedia information systems: The Garlic approach. In: Proc. of the 5th Int. Workshop on Research Issues in Data Engineering – Distributed Object Management (RIDE-DOM'95), IEEE CS Press (1995) 124–131 264

24. Li, C., Yerneni, R., Vassalos, V., Garcia-Molina, H., Papakonstantinou, Y., Ullman, J. D., Valiveti, M.: Capability based mediation in TSIMMIS. In: Proc. of ACM SIGMOD. (1998) 564–566 264
25. Galhardas, H., Florescu, D., Shasha, D., Simon, E.: An extensible framework for data cleaning. Technical Report 3742, INRIA, Rocquencourt (1999) 268
26. Bouzeghoub, M., Lenzerini, M.: Introduction to the special issue on data extraction, cleaning, and reconciliation. Information Systems **26** (2001) 535–536 268
27. Lloyd, J. W.: Foundations of Logic Programming (Second, Extended Edition). Springer, Berlin, Heidelberg (1987) 272, 273, 274, 275
28. Lloyd, J. W., Shepherdson, J. C.: Partial evaluation in logic programming. J. of Logic Programming **11** (1991) 217–242 274, 275