Query Answering by Rewriting in GLAV Data Integration Systems Under Constraints

Andrea Calì

Faculty of Computer Science,
Free University of Bolzano/Bozen,
piazza Domenicani 3 – I-39100 Bolzano, Italy
ac@andreacali.com

Abstract. In the Semantic Web, the goal is offering access to information that is distributed over the Internet. Data integration is highly relevant in this context, since it consists in providing a uniform access to a set of data sources, through a unified representation of the data called *global schema*. Integrity constraints (ICs) are expressed on the global schema in order to better represent the domain of interest, yet such constraints may not be satisfied by the data at the sources. In this paper we address the problem of answering queries posed to a data integration system where the mapping is specified in the so-called GLAV approach, and when tuple-generating dependencies (TGDs) and functional dependencies (FDs) are expressed over the global schema. We extend previous results by first showing that, in the case of TGDs without FDs, known query rewriting techniques can be applied in a more general case, and can take into account also the GLAV mapping in a single rewriting step. Then we introduce FDs with TGDs, identifying a novel class of ICs for which query answering is decidable, and providing a query answering algorithm based on query rewriting also in this case.

1 Introduction

In the Semantic Web [15, 8] the goal is to enrich data accessible on the Web in order to provide semantic knowledge that facilitates users in retrieving and accessing information that is relevant for them. Ideally, the aim is to allow users to pose queries to the Web as if they were querying a single, local knowledge base. Hence, one of the fundamental issues of the Semantic Web is that of integrating the information present in the various sources, and therefore data integration techniques prove to be higly useful for this task. Conceptual modelling formalisms, such as the Entity-Relationship model, have proved to be highly effective for representing intensional information about data, and are now widely accepted means of enriching data with semantic knowledge about the domain of interest. When the data are represented in the relation model, as in the case of this paper, *integrity constraints (ICs)*, like key and foreign key constraints, are able to capture most of the information expressed by conceptual modelling formalisms.

C. Bussler et al. (Eds.): SWDB 2004, LNCS 3372, pp. 167–184, 2005.

A data integration system has the goal of providing a uniform access to a set of heterogeneous data sources, through a unified view of all underlying data, called *global schema*. Once the user issues a query over the global schema, the system carries out the task of suitably accessing the different sources and assemble the retrieved data into the final answer to the query.

In data integration, the specification of the relationship between the global schema and the sources, which is called *mapping* [18], is a significant issue. There are two basic approaches for specifying a mapping in a data integration system [14, 18]. The first one, called *global-as-view (GAV)*, requires that to each element of the global schema a view over the sources is associated. The second approach, called *local-as-view (LAV)*, requires that to each source a view over the global schema is associated. Besides GAV and LAV, a mixed approach, called GLAV [11, 10], consists in associating views over the global schema to views over the sources.

In a data integration system, the global schema is a representation of the domain of interest of the system, therefore there is the need of having a global schema that fits the fragment of real world that is modelled by the system. To this aim, integrity constraints are expressed on the global schema.

It is important to notice that integrity constraints are used to enhance the expressiveness of the global schema, and their presence is not due to constraints on the sources, which in our approach are supposed to be enforced by the systems that manage the local data. Therefore, in general, the data at the sources may not satisfy the constraints on the global schema; in this case a common assumption (which is the one adopted in this paper) is to to consider the sources as *sound*, i.e., they provide a *subset* of the data that satisfy the global schema. Answering queries posed over the global schema in this setting requires to consider a *set* of databases for the global schema, and in particular all those that contain the data provided by the sources through the mapping, and that satisfy the ICs on the global schema. Therefore, query answering requires reasoning on incomplete information.

In this paper we address the problem of query answering in data integration in the relational context, where the mapping is GLAV, and in the presence of *tuple-generating dependencies (TGDs)* and *functional dependencies (FDs)* on the global schema; TGDs and FDs are an extension of two important classes of dependencies in relational database schemata, namely *inclusion dependencies* and *key dependencies* respectively [1, 16].

First we consider TGDs alone: since the query answering under general TGDs is undecidable [17], we solve the problem for a restricted class of TGDs, that we call *cycle-harmless TGDs*, that extends a decidable class known in the literature. Our approach is purely intensional here: we do query answering by *query rewriting*, i.e. by reformulating the query into a new one, which encodes the information about the integrity constraints, and then proceeding as if there were no integrity constraints. The rewriting algorithm used here is the same as in [3]. The form of the TGDs we consider is general enough to consider the GLAV mapping assertion as TGDs over a unified schema constituted by the union of the

global schema and the source schema; therefore, we are able to apply the rewriting technique in a single step, taking into account the TGDs and the mapping at the same time.

Then, we address the problem of query answering when also FDs are expressed on the global schema. Even if we consider cycle-harmless TGDs, the presence of FDs causes an interaction that makes query answering again undecidable. We need to consider a new, more restricted class of TGDs, which we call *non-functional-conflicting TGDs*, that do not interact with FDs, and again we present a technique for query answering. In this case, however, we cannot take into account the mapping in an intensional fashion, because it is not realistic to assume that the mapping assertions are non-functional-conflicting TGDs. Therefore, in order to answer queries, we need to construct the *retrieved global database (RGD)* [4], that is the minimum global database that satisfies the mapping. If the RGD satisfies the FDs, we can proceed with the same rewriting technique used for TGDs, simply disregarding the presence of FDs.

The rest of the paper is organised as follows. In Section 2 we present a formal framework for data integration; in Section 3 we address the query answering problem for TGDs alone; in Section 4 we introduce FDs together with TGDs, showing a decidable class of constraints for this case and providing a query answering technique. Section 5 concludes the paper.

2 Framework

In this section we define a logical framework for data integration, based on the relational model with integrity constraints.

2.1 Syntax

We consider to have an infinite, fixed alphabet Γ of constants representing real world objects, and will take into account only databases having Γ as domain.

A *relational schema* $\mathcal{R}$ is a set of first-order predicates, called *relation symbols*, each with an associated arity; a relation symbol R of arity n is denoted by R/n. A *relational database instance* of a schema $\mathcal{R}$ is a set of facts of the form $R(c_1, \ldots, c_n) \leftarrow$, where $R/n \in \mathcal{R}$ and $c_1, \ldots, c_n \in \Gamma$. In the following, for the sake of conciseness, we will use the term "database" instead of "database instance".

Formally, a data integration system $\mathcal{I}$ is a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where:

1. $\mathcal{G}$ is the *global schema* expressed in the relational model with integrity constraints. In particular:
   (a) $\mathcal{G}$ is constituted by a set of relations, each with an associated arity that indicates the number of its attributes. A relation R of arity n is denoted by R/n.
   (b) A set Σ_T of *tuple-generating dependencies (TGDs)* is expressed over $\mathcal{G}$. A TGD [1, 10] is a first-order formula of the form

$$\forall \boldsymbol{X}(\exists \boldsymbol{Y}\ \chi(\boldsymbol{X}, \boldsymbol{Y}) \ \rightarrow\ \exists \boldsymbol{Z}\ \psi(\boldsymbol{X}, \boldsymbol{Z}))$$

where $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ are sets of variables or constants of Γ, and χ and ψ are conjunctions of atoms whose predicate symbols are in $\mathcal{G}$. Henceforth, for the sake of conciseness, we will omit the quantifiers in TGDs.

(c) A set Σ_F of *functional dependencies (FDs)* is expressed over $\mathcal{G}$. A FD [1] is written in the form

$$R: \ \mathbf{A} \ \rightarrow \ B$$

where R is a relation symbol, $\mathbf{A}$ is a set of attributes of R, and B is an attribute of R.

2. $\mathcal{S}$ is the *source schema*, constituted by the schemata of the different sources. We assume that the sources are relational, and in particular that each source is represented by a single relation. Assuming sources to be relational is not a restriction, since we may assume that sources that are not relational are suitably accessible in relational form by means of software modules called *wrappers*. Furthermore, we assume that no integrity constraint is expressed on the source schema. This because integrity constraints on the sources are local to the data source, and they are enforced by the source itself.
3. $\mathcal{M}$ is the *mapping* between $\mathcal{G}$ and $\mathcal{S}$, specifying the relationship between the global schema and the source schema. The mapping $\mathcal{M}$ is a set of first-order formulae, which we call *mapping assertions*, of the form

$$\forall \boldsymbol{X}(\exists \boldsymbol{Y} \varphi_{\mathcal{S}}(\boldsymbol{X}, \boldsymbol{Y}) \ \rightarrow \ \exists \boldsymbol{Z} \varphi_{\mathcal{G}}(\boldsymbol{X}, \boldsymbol{Z}))$$

where $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ are sets of variables or constants, and $\varphi_{\mathcal{S}}$ and $\varphi_{\mathcal{G}}$ are conjunctions of atoms whose predicate symbols are in $\mathcal{S}$ and $\mathcal{G}$ respectively. Henceforth, we will omit quantifiers in mapping formulae. Note that this kind of mapping assertions is a generalisation of both LAV and GAV assertions; in particular, in a LAV assertion a view (conjunction of atoms) over the global schema is associated to a source relation, while in a GAV assertion a view over the source schema is associated to a relation symbol in $\mathcal{G}$. Henceforth, consistently with [11], we will call *GLAV (global-local-as-view)* this approach.

Example 1 ([19]). Consider a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where the global schema $\mathcal{G}$ consists of the following relations (names of relations and attributes are self-explanatory):

$$\begin{array}{l} \mathsf{work}(\mathit{Person}, \mathit{Project}) \\ \mathsf{area}(\mathit{Project}, \mathit{Field}) \\ \mathsf{employed}(\mathit{Person}, \mathit{Institution}) \\ \mathsf{runsProject}(\mathit{Institution}, \mathit{Project}) \end{array}$$

The source schema $\mathcal{S}$ contains the following relations:

$$\begin{array}{l} \mathsf{hasjob}(\mathit{Person}, \mathit{Field}) \\ \mathsf{teaches}(\mathit{Professor}, \mathit{Course}) \\ \mathsf{infield}(\mathit{Course}, \mathit{Field}) \\ \mathsf{getgrant}(\mathit{Researcher}, \mathit{Grant}) \\ \mathsf{grantfor}(\mathit{Grant}, \mathit{Project}) \end{array}$$

The GLAV mapping $\mathcal{M}$ is as follows:

$$\begin{array}{ll} \mathsf{hasjob}(R, F) & \rightarrow \mathsf{work}(R, P) \wedge \mathsf{area}(P, F) \\ \mathsf{getgrant}(R, G) \wedge \mathsf{grantfor}(G, P) & \rightarrow \mathsf{work}(R, P) \\ \mathsf{teaches}(R, C) \wedge \mathsf{infield}(C, F) & \rightarrow \mathsf{work}(R, P) \wedge \mathsf{area}(P, F) \end{array}$$

Note that the first assertion is in fact a LAV assertion, the second one is a GAV one, while the third assertion is a general GLAV assertion.

The set Σ_T of TGDs is constituted by the following TGD:

$$\mathsf{work}(R, P) \rightarrow \mathsf{employed}(R, I) \wedge \mathsf{runsProject}(I, P)$$

This TGD imposes that, if a person R works on a project P, he/she needs to be employed in some institution I that is running the project P. ■

Example 2. Consider a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where the global schema $\mathcal{G}$ is constituted by the relations $R_1/2$ and $R_2/2$, the source schema by relations $S_1/2, S_2/1$. The set of TGDs Σ_T contains the single TGD $\theta : R_1(X, Y) \rightarrow R_1(Y, W) \wedge R_2(Y, X)$. Note that θ introduces a cycle in the dependencies. The mapping $\mathcal{M}$ consists of the assertions $S_1(X, c) \rightarrow R_1(X, Y) \wedge R_2(Y, Z)$ and $S_2(X) \rightarrow R_2(X, Y)$. ■

Now we come to queries expressed over the global schema; a n-ary *relational query* (relational query of arity n) is a formula that is intended to specify a set of n-tuples of constants in Γ, that constitute the *answer* to the query. In our setting, we assume that queries over the global schema are expressed in the language of *union of conjunctive queries (UCQs)*. A conjunctive query (CQ) of arity n is a formula of the form $q(\boldsymbol{X}) \leftarrow \omega(\boldsymbol{X}, \boldsymbol{Y})$ where $\boldsymbol{X}$ is a set of variables called *distinguished variables*, $\boldsymbol{Y}$ is a set of symbols that are either variables (called *non-distinguished*) or constants, q is a predicate symbol not appearing in $\mathcal{G}$ or $\mathcal{S}$, and ω is a conjunction of atoms whose predicate symbols are in $\mathcal{G}$. The atom $q(\boldsymbol{X})$ is called *head* of the query (denoted $head(q)$), while $\omega(\boldsymbol{X}, \boldsymbol{Y})$ is called *body* (denoted $body(q)$). A UCQ of arity n is a set of conjunctive queries Q such that each $q \in Q$ has the same arity n and uses the same predicate symbol in the head.

2.2 Semantics

A *database instance* (or simply *database*) $\mathcal{C}$ for a relational schema $\mathcal{R}$ is a set of facts of the form $R(t)$ where R is a relation of arity n in $\mathcal{R}$ and t is an n-tuple of constants of the alphabet Γ. We denote as $R^{\mathcal{C}}$ the set of tuples of the form $\{t \mid R(t) \in \mathcal{C}\}$.

In the following, we shall often make use of the notion of substitution. A *substitution* of variables σ is a partial function that associates to a variable either a constant or a variable, and to each constant the constant itself. In the following, given a first-order formula F, we will denote with $\sigma(F)$ the formula obtained by replacing in F each variable (or constant) x with $\sigma(x)$. Given an

atomic formula $R(x_1, \ldots, x_n)$, where R is a n-ary predicate and $x_1, \ldots, x_n$ are variables or constants, we say that a substitution σ sends $R(x_1, \ldots, x_n)$ to the fact $\sigma(R(x_1, \ldots, x_n))$, that we denote with $R(\sigma(x_1), \ldots, \sigma(x_n)) \leftarrow$. Moreover, given a conjunction $C = A_1 \wedge \ldots \wedge a_m$ of atomic formulae, we will say that a substitution σ sends C to the set of facts $\{\sigma(A_1), \ldots, \sigma(A_m)\}$.

Given a CQ q of arity n and a database instance $\mathcal{C}$, we denote as $q^{\mathcal{C}}$ the evaluation of q over $\mathcal{C}$, i.e., the set of n-tuples $\overline{t}$ of constants of Γ such that there exists a substitution that sends the atoms of q to facts of $\mathcal{C}$ and the head to $q(\overline{t})$. Moreover, given a UCQ Q, we define the evaluation of Q over $\mathcal{C}$ as $Q^{\mathcal{C}} = \bigcup_{q \in Q} q^{\mathcal{C}}$

Now we come to the semantics of a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$. Such a semantics is defined by first considering a *source database* for $\mathcal{I}$, i.e., a database $\mathcal{D}$ for the source schema $\mathcal{S}$. We call *global database* for $\mathcal{I}$ any database for $\mathcal{G}$. Given a source database $\mathcal{D}$ for $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, the semantics $sem(\mathcal{I}, \mathcal{D})$ of $\mathcal{I}$ w.r.t. $\mathcal{D}$ is the set of global databases $\mathcal{B}$ for $\mathcal{I}$ such that:

1. $\mathcal{B}$ satisfies the ICs Σ_T and Σ_F (TGDs and FDs) in $\mathcal{G}$. In particular:
   - $\mathcal{B}$ satisfies a TGD $\chi(\boldsymbol{X}, \boldsymbol{Y}) \rightarrow \psi(\boldsymbol{X}, \boldsymbol{Z})$ when, if there exists a substitution σ that sends $\chi(\boldsymbol{X}, \boldsymbol{Y})$ to a set of facts of $\mathcal{B}$, then there exists another substitution σ' that sends $\psi(\boldsymbol{X}, \boldsymbol{Z})$ to $\sigma(\chi(\boldsymbol{X}, \boldsymbol{Y}))$ and those of $\chi(\boldsymbol{X}, \boldsymbol{Y})$ to sets of facts of $\mathcal{B}$. In other words, σ' is an extension of σ that sends the atoms of $\psi(\boldsymbol{X}, \boldsymbol{Z})$ to sets of facts of $\mathcal{B}$.
   - $\mathcal{B}$ satisfies a FD $R : \mathbf{A} \rightarrow B$ if there are no two tuples $t_1, t_2 \in R^{\mathcal{B}}$ such that $t_1[\mathbf{A}] = t_2[\mathbf{A}]$ and $t_1[B] \neq t_2[B]$.
2. $\mathcal{B}$ satisfies $\mathcal{M}$ w.r.t. $\mathcal{D}$. In particular, $\mathcal{B}$ satisfies a GLAV mapping $\mathcal{M}$ w.r.t. $\mathcal{D}$ if for each mapping formula $\varphi_{\mathcal{S}}(\boldsymbol{X}, \boldsymbol{Y}) \rightarrow \varphi_{\mathcal{G}}(\boldsymbol{X}, \boldsymbol{Z})$ we have that, if there exists a substitution σ that sends $\varphi_{\mathcal{S}}(\boldsymbol{X}, \boldsymbol{Y})$ to a set of facts of $\mathcal{D}$, then there exists an extension σ' of σ that sends $\varphi_{\mathcal{G}}(\boldsymbol{X}, \boldsymbol{Z})$ to a set of facts of $\mathcal{B}$. Note that the above definition amounts to consider the mapping as *sound* but not necessarily complete; intuitively, for each mapping formula, the data retrievable at the sources by means of the conjunctive query in the left-hand side are a *subset* of the global data that satisfy the conjunctive query on the right-hand side.

We now give the semantics of queries. Formally, given a source database $\mathcal{D}$ for $\mathcal{I}$ we call *certain answers* to a query q of arity n w.r.t. $\mathcal{I}$ and $\mathcal{D}$, the set

$$cert(Q, \mathcal{I}, \mathcal{D}) = \{\overline{t} \mid \overline{t} \in Q^{\mathcal{B}} \text{ for each } \mathcal{B} \in sem(\mathcal{I}, \mathcal{D})\}$$

or, equivalently, $cert(Q, \mathcal{I}, \mathcal{D}) = \bigcap_{\mathcal{B} \in sem(\mathcal{I}, \mathcal{D})} Q^{\mathcal{B}}$.

3 Query Rewriting Under Tuple-Generating Dependencies Alone

In this section we present a technique for query answering based on query rewriting, in the case of a GLAV data integration system where only TGDs are expressed over the global schema. We show that such technique, first presented

in [3], is applicable to a more general class of constraints, so that it can take into account the GLAV mapping together with the dependencies on the global schema.

We first introduce the concept of *retrieved global database (RGD)*. Given a source database $\mathcal{D}$ for a data integration system $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, the RGD is the "minimum" global database that satisfies the mapping. Intuitively, the RGD is obtained by "filtering" the data from the sources through the mapping, thus populating the global schema.

Definition 1 ([3]). *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a GLAV data integration system, and $\mathcal{D}$ a source database for $\mathcal{I}$. The* retrieved global database *$ret(\mathcal{I}, \mathcal{D})$ is defined constructively as follows. For every mapping assertion $\varphi_{\mathcal{S}}(\boldsymbol{X}, \boldsymbol{Y}) \rightarrow \varphi_{\mathcal{G}}(\boldsymbol{X}, \boldsymbol{Z})$, and for each set H of facts of $\mathcal{D}$ such that there exists a substitution σ that sends the atoms of $\varphi_{\mathcal{S}}(\boldsymbol{X}, \boldsymbol{Y})$ to H: (i) we first define a substitution σ' such that $\sigma'(X_i) = \sigma(X_i)$ for each X_i in $\boldsymbol{X}$, and $\sigma'(Z_j) = z_j$ for each Z_j in $\boldsymbol{Z}$, where z_j is a fresh constant, not introduced before and not appearing in $\mathcal{D}$; (ii) we add to $ret(\mathcal{I}, \mathcal{D})$ the set of facts that are in $\sigma'(\varphi_{\mathcal{G}}(\boldsymbol{X}, \boldsymbol{Z}))$.*

Note that, given a a data integration system and a source database $\mathcal{D}$, the RGD is unique, since it is constructed by evaluating the left-hand side of every mapping assertion on $\mathcal{D}$, and by adding suitable tuples according to the right-hand side of the mapping assertion, regardless of the already added tuples and of the other mapping assertions. However, differently from the case of GAV mappings, the RGD is not strictly minimal, since in some cases it is possible to have redundant tuples that can be eliminated while preserving all the properties of the RGD. Minimisation of the RGD is not a significant issue, therefore we will not consider it in the following.

When no constraints are expressed over the global schema, the RGD is a representative of all global databases that satisfy the mapping (and therefore of all databases in $sem(\mathcal{I}, \mathcal{D})$): in fact in this case it can be proved that, for every query Q posed over the global schema, $Q^{ret(\mathcal{I}, \mathcal{D})} = cert(Q, \mathcal{I}, \mathcal{D})$.

Now we come to integrity constraints. Given a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and a source database $\mathcal{D}$, since sources are autonomous and in general do not know each other, the retrieved global database $ret(\mathcal{I}, \mathcal{D})$ does not satisfy the integrity constraints (TGDs in this case) on the global schema. In this case we may think of *repairing* the RGD so as to make it satisfy Σ_T; intuitively, the adoption of the *sound* semantics for the mapping $\mathcal{M}$ allows us to repair the violations of TGDs by adding suitable tuples to the RGD. This is done by building the *chase* [4, 10, 3] of $ret(\mathcal{I}, \mathcal{D})$, a database that we denote with $chase_{\Sigma_T}(ret(\mathcal{I}, \mathcal{D}))$, and that is built by repeatedly applying, as long as it is applicable, the *TGD chase rule*.

> TGD Chase Rule [3]. Consider a database $\mathcal{B}$ for a schema Ψ, and a TGD θ of the form $\chi(\boldsymbol{X}, \boldsymbol{Y}) \rightarrow \psi(\boldsymbol{X}, \boldsymbol{Z})$. The TGD θ is *applicable* to $\mathcal{B}$ if there is a substitution σ that sends the atoms of $\chi(\boldsymbol{X}, \boldsymbol{Y})$ to tuples of $\mathcal{B}$, and there does not exist a substitution $\bar{\sigma}$ that sends the atoms of $\chi(\boldsymbol{X}, \boldsymbol{Y})$ to $\sigma(\chi(\boldsymbol{X}, \boldsymbol{Y}))$, and the atoms of $\psi(\boldsymbol{X}, \boldsymbol{Z})$ to tuples of $\mathcal{B}$. In this

case: *(i)* we define a substitution σ' such that $\sigma'(X_i) = \sigma(X_i)$ for each X_i in $\boldsymbol{X}$, and $\sigma'(Z_j) = z_j$ for each Z_j in $\boldsymbol{Z}$, where z_j is a fresh constant of Γ, not already introduced in the construction and not appearing in $\mathcal{B}$; *(ii)* we add to $\mathcal{B}$ the facts of $\sigma'(\varphi_{\mathcal{G}}(\boldsymbol{X}, \boldsymbol{Z}))$ that are not already in $\mathcal{B}$.

Note that in the case of cyclic TGDs, the chase may be infinite.

Example 3. Consider Example 2, and let $\mathcal{B}$ be a RGD constituted by a single fact $R_1(a, b)$. Let us construct $chase_{\Sigma_T}(\mathcal{B})$: at the first step we add the facts $R_1(b, z_1), R_2(b, a)$; at the second step the facts $R_1(z_1, z_2), R_2(z_1, b)$, and so on; note that in this case the construction process is infinite. ∎

In [3] it is proved that the chase of the RGD, constructed according to the TGDs, is a representative of all databases in $sem(\mathcal{I}, \mathcal{D})$: in particular, for every global query Q we have that $Q^{chase_{\Sigma_T}(ret(\mathcal{I},\mathcal{D}))} = cert(Q, \mathcal{I}, \mathcal{D})$. Unfortunately the chase of the RGD may be of infinite size, and therefore building it is not only impractical, but sometimes even impossible. In [3], along the lines of [6], the problem is solved in an intensional fashion: query answering is done by query rewriting, i.e., the global query Q is reformulated into another query Q_R that, evaluated over the RGD, returns the certain answers to Q. The function that reformulates Q is denoted as TGDrewrite, and takes as input $\mathcal{G}$, the set of TGDs Σ_T and Q. Formally, we have that

$$\mathsf{TGDrewrite}(\mathcal{G}, \Sigma_T, Q)^{ret(\mathcal{I},\mathcal{D})} = Q^{chase_{\Sigma_T}(ret(\mathcal{I},\mathcal{D}))} = cert(Q, \mathcal{I}, \mathcal{D})$$

Such technique avoids the construction of the RGD. The technique is applicable for a restricted class of TGDs, namely the *weakly-joined TGDs (WJTGDs)*; in fact, it is known that query answering under general TGDs is undecidable [17]. WJTGDs are defined as follows.

Definition 2 ([3]). *A TGD of the form* $\chi(\boldsymbol{X}, \boldsymbol{Y}) \rightarrow \psi(\boldsymbol{X}, \boldsymbol{Z})$ *is a* weakly-joined TGD *(WJTGD) if each* $Y_i \in \boldsymbol{Y}$ *appears at most once in it.*

We give a description of how the algorithm TGDrewrite works. The idea is that the algorithm repeatedly executes a basic rewriting step (together with a minimisation step that we will not see in detail), until there are no more CQs to be added to the rewritten query. In the basic rewriting step, TGDrewrite verifies (besides some other conditions) whether there is a substitution σ that sends a subset of the atoms of the right-hand side of some TGD θ to a subset G of the atoms of the body of some CQ q in Q. If this happens, a new CQ is added to Q, obtained by replacing in q the set of atoms G with the left-hand side of θ, where substitution σ has been applied.

Example 4. Consider Example 2 and a CQ $q(X_1) \leftarrow R_1(X_1, X_2), R_2(X_1, X_3)$, represented as $q(X_1) \leftarrow R_1(X_1, \star), R_2(X_1, \star)$, where (similarly to the Logic Programming notation, where the symbol "_" is used) we indicate with $\star$ the variables appearing only once in the query. The WJTGD θ is applicable to $G = \{R_1(X_1, \star), R_2(X_1, \star)\}$, because the substitution $\sigma = \{X \rightarrow X_3, Y \rightarrow$

$X_1, W \rightarrow X_2\}$ sends the right-hand side of θ to G. Therefore, we apply the basic rewriting step by replacing G with the left-hand side of θ and by applying σ to the obtained query. The result is the CQ $q(X_1) \leftarrow R_1(X_3, X_1)$, which we represent as $q(X_1) \leftarrow R_1(\star, X_1)$.

Suppose there is another WJTGD θ_1 on $\mathcal{G}$, of the form $R_2(Y, W) \rightarrow R_1(X, Y)$. Though there is a substitution sending the right-hand side of θ to $G_1 = \{R_1(X_1, X_2)\}$, the basic rewriting step cannot be executed because the variable X_1 would "disappear", and this is not allowed since X_1 appears outside G_1 (in particular, both in the $head(q)$ and in another atom of $body(q)$). ∎

In [3] the mapping $\mathcal{M}$ is taken into account in a separate step, by first transforming the GLAV system into a GAV system; query answering in GAV systems is then done by a traditional rewriting technique called *unfolding* [18]. However, the form of the GLAV mapping assertions, similar to TGDs, suggests that the algorithm TGDrewrite can be used for the mapping as well.

Now we introduce a more general class of constraints that will allow us to deal with the mapping assertions together with the constraints on the global schema. First, we consider the global schema $\mathcal{G}$ and the source schema $\mathcal{S}$ as a single database schema, on which are expressed the TGDs in Σ_T, plus the assertions in $\mathcal{M}$, that now we can see as TGDs on $\mathcal{G} \cup \mathcal{S}$, and that we denote with $\Sigma_{\mathcal{M}}$. The TGDs in $\Sigma_{\mathcal{M}}$ are in general not WJTGDs, but the following results, extending those of [3], allows us to deal with a more general class of constraints. We give some preliminary definitions.

Definition 3. *Given a set Σ_T of TGDs expressed over a global schema $\mathcal{G}$, the* TGD-graph *G_{Σ_T} associated to it is defined as follows:*

- *the set of nodes is the set of relation symbols in $\mathcal{G}$;*
- *an arc (R_1, R_2) exists if R_1 and R_2 appear respectively in the left-hand and right-hand side of some TGD in Σ_T.*

Definition 4. *Given a set Σ_T of TGDs expressed over a global schema $\mathcal{G}$, and the corresponding TGD-graph G_{Σ_T}, a TGD θ is said to be* cycle-harmless *w.r.t. Σ_T if at least one of the following conditions holds:*

1. *for any two relation symbols R_1, R_2 appearing in the body and in the head of θ respectively, the arc (R_1, R_2) is not part of any cycle in G_{Σ_T}, or*
2. *θ is a WJTGD.*

Now we come to the results.

Lemma 1. *Let B be a relational database for a schema $\mathcal{R}$, and Σ_T a set of cycle-harmless TGDs expressed over $\mathcal{R}$. Then for every query Q on $\mathcal{R}$, expressed in UCQs, we have that* $\mathsf{TGDrewrite}(\mathcal{R}, \Sigma_T, Q)^{\mathcal{B}} = Q^{chase_{\Sigma_T}(\mathcal{B})}$.

Proof (sketch). We want to prove that, being n the arity of Q, for any n-tuple t of constants of Γ, we have

$$t \in \mathsf{TGDrewrite}(\mathcal{R}, \Sigma_T, Q)^{\mathcal{B}} \text{ iff } t \in Q^{chase_{\Sigma_T}(\mathcal{B})}$$

"⇐"The proof is by induction on the number of applications of the TGD chase rule. By hypothesis, there is a CQ q in Q such that there exists a query homomorphism ψ (see [16]) that sends the body of q to tuples of $chase_{\Sigma_T}(\mathcal{B})$ and the head of q to t. At the base step, ψ sends the body of q to tuples of $\mathcal{B}$, and since q is part of $\mathsf{TGDrewrite}(\mathcal{R}, \Sigma_T, Q)$, the thesis follows. As for the inductive step, suppose that the result holds when ψ sends the body of q to tuples of the chase (that we briefly denote as $\psi(q)$) that are generated after k applications of the chase rule. It is possible to show that, denoting by H the set of tuples from which $\psi(q)$ is generated (and that are therefore generated from $\mathcal{B}$ with $k-1$ applications of the chase rule), there exists a query homomorphism ψ_1 that sends the body of q_1 to tuples in H and the head of q_1 to t, where q_1 is obtained from q by application of the basic rewriting step of $\mathsf{TGDrewrite}$.

"⇒"The proof is analogous to the previous one, by induction on the number of applications of the basic rewriting rule of $\mathsf{TGDrewrite}$.

Lemma 2. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system; let Σ_T be a set of TGDs expressed over $\mathcal{G}$. If for every TGD θ in Σ_T it holds that θ is a cycle-harmless TGD w.r.t. Σ_T, then for every source database $\mathcal{D}$ and for every global query Q we have*

$$Q^{chase_{\Sigma_T}(ret(\mathcal{I},\mathcal{D}))} = cert(Q, \mathcal{I}, \mathcal{D})$$

Proof (sketch). Analogously to what is done in [4] for the chase in the presence of inclusion dependencies, it is possible to prove that, for every global database $\mathcal{B}$ that is in $sem(\mathcal{I}, \mathcal{D})$, there exists a homomorphism λ that sends tuples of $chase_{\Sigma_T}(ret(\mathcal{I}, \mathcal{D}))$ to those of $\mathcal{B}$.

We now prove that, being n the arity of Q, for every n-tuple of constants of Γ,

$$t \in Q^{chase_{\Sigma_T}(ret(\mathcal{I},\mathcal{D}))} \text{ iff } t \in cert(Q, \mathcal{I}, \mathcal{D})$$

"⇐"If $t \notin Q^{chase_{\Sigma_T}(ret(\mathcal{I},\mathcal{D}))}$, since the chase is a database in $sem(\mathcal{I}, \mathcal{D})$ because it satisfies both the mapping and the ICs, we immediately deduce that $t \in cert(Q, \mathcal{I}, \mathcal{D})$.

"⇒"By hypothesis, there exists a query homomorphism μ that sends the body of some CQ q in Q to tuples of $chase_{\Sigma_T}(ret(\mathcal{I}, \mathcal{D}))$. Recalling the existence of the homomorphism λ for any global database $\mathcal{B}$ in $sem(\mathcal{I}, \mathcal{D})$, we consider the composition $\mu \circ \lambda$. The existence of such homomorphism for any $\mathcal{B}$ in $sem(\mathcal{I}, \mathcal{D})$ guarantees that $t \in Q^{\mathcal{B}}$ for every $\mathcal{B} \in sem(\mathcal{I}, \mathcal{D})$, and therefore $t \in cert(Q, \mathcal{I}, \mathcal{D})$. ∎

Theorem 1. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system; let Σ_T be a set of TGDs expressed over $\mathcal{G}$. If for every TGD θ in Σ_T it holds that θ is a cycle-harmless TGD w.r.t. Σ_T, then for every source database $\mathcal{D}$ and for every global query Q we have*

$$\mathsf{TGDrewrite}(\mathcal{G}, \Sigma_T, Q)^{ret(\mathcal{I},\mathcal{D})} = cert(Q, \mathcal{I}, \mathcal{D})$$

Proof. By Lemma 2 we have $\mathsf{TGDrewrite}(\mathcal{G}, \Sigma_T, Q)^{ret(\mathcal{I},\mathcal{D})} = Q^{chase_{\Sigma_T}(ret(\mathcal{I},\mathcal{D}))}$. The thesis follows immediately from Lemma 2. ∎

The following theorem, directly derived from Theorem 1, allows us to take the mapping into account in a single step with the algorithm TGDrewrite.

Theorem 2. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system; let Σ_T be a set of TGDs expressed over $\mathcal{G}$ and $\Sigma_\mathcal{M}$ the TGDs that constitute $\mathcal{M}$. If for every TGD θ in Σ_T it holds that θ is cycle-harmless w.r.t. Σ_T, then for every source database $\mathcal{D}$ and for every global query Q we have*

$$\mathsf{TGDrewrite}(\mathcal{G}, \Sigma_T \cup \Sigma_\mathcal{M}, Q)^\mathcal{D} = cert(Q, \mathcal{I}, \mathcal{D})$$

Proof. The proof is done by observing that $chase_{\Sigma_\mathcal{M}}(\mathcal{D}) = ret(\mathcal{I}, \mathcal{D})$, and that $chase_{\Sigma_T}(ret(\mathcal{I}, \mathcal{D})) = chase_{\Sigma_T \cup \Sigma_\mathcal{M}}(\mathcal{D})$. Note that all TGDs in $\Sigma_\mathcal{M}$ are cycle-harmless by construction. By Lemma 1 we have

$$\mathsf{TGDrewrite}(\mathcal{G}, \Sigma_T \cup \Sigma_\mathcal{M}, Q)^\mathcal{D} = Q^{chase_{\Sigma_T \cup \Sigma_\mathcal{M}}(\mathcal{D})}$$

and therefore

$$\mathsf{TGDrewrite}(\mathcal{G}, \Sigma_T \cup \Sigma_\mathcal{M}, Q)^\mathcal{D} = Q^{chase_{\Sigma_T}(chase_{\Sigma_\mathcal{M}}(\mathcal{D}))} = Q^{chase_{\Sigma_T}(ret(\mathcal{I},\mathcal{D}))}$$

The thesis follows immediately from Lemma 2. ∎

Note that we have in some way abused the notation in the statement of the previous theorem; in fact we are evaluating TGDrewrite, which in general is formulated over $\mathcal{G} \cup \mathcal{S}$, over a source database $\mathcal{D}$ that is a database for $\mathcal{S}$. However, we can consider $\mathcal{D}$ as a database for $\mathcal{G} \cup \mathcal{S}$ where for each $g \in \mathcal{G}$ we have $g^\mathcal{D} = \emptyset$. Indeed, this observation leads us to the obvious conclusion that, once $\mathsf{TGDrewrite}(\mathcal{G}, \Sigma_T \cup \Sigma_\mathcal{M}, Q)$ is computed, in its evaluation over $\mathcal{D}$ we can omit to consider all the CQs in which at least one atom with a relation symbol of $\mathcal{G}$ appears. This can save computation time in the query rewriting phase.

Example 5. Recall Example 1. Suppose the source database $\mathcal{D}$ contains a single fact $\mathsf{hasjob}(anne, maths) \leftarrow$. Consider the global query

$$Q(X) \leftarrow \mathsf{employed}(X, Y) \wedge \mathsf{runsProject}(Y, Z)$$

asking for persons employed in institutions that run some project. A rewriting step, according to the single TGD expressed on $\mathcal{G}$, will produce the CQ

$$Q_1(X) \leftarrow \mathsf{work}(X, Z)$$

Applying the mapping assertions as rewriting rules, we obtain the following CQs:

$$Q_2(X) \leftarrow \mathsf{hasjob}(X, W_1)$$
$$Q_3(X) \leftarrow \mathsf{getgrant}(X, W_2) \wedge \mathsf{grantfor}(W_2, Z)$$
$$Q_4(X) \leftarrow \mathsf{teaches}(X, W_3) \wedge \mathsf{infield}(W_3, W_4)$$

The final rewriting is $Q_R = Q \vee Q_1 \vee Q_2 \vee Q_3 \vee Q_4$ (however, Q and Q_1 will not be considered since they contain relation names not appearing in $\mathcal{S}$). The evaluation of the rewriting over the source database $\mathcal{D}$ returns the answer $Q_R^\mathcal{D} = \{(anne)\}$. ∎

4 Query Rewriting Under Tuple-Generating Dependencies and Functional Dependencies

In this section we address the problem of query answering in GLAV systems where two sets of TGDs and FDs, that we will denote with Σ_T and Σ_F respectively, are expressed over the global schema. In this case, even if we restrict to TGDs that are either cycle-harmless w.r.t. Σ_T, the problem of query answering is undecidable.

Theorem 3. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system, where two sets of TGDs and FDs, Σ_T and Σ_F respectively, are expressed over $\mathcal{G}$; let Σ_T be a set of cycle-harmless TGDs w.r.t. Σ_T itself. We have that the problem of query answering is undecidable.*

Proof. The proof is derived from the undecidability result for query answering in the presence of inclusion dependencies and key dependencies [5], which is clearly a particular case of the one considered here. Note that inclusion dependencies are cycle-harmless TGDs, since they cannot have joins in the left-hand side. In turn, the result of [5] is derived from the undecidability result about implication of functional and inclusion dependencies [7]. ∎

Here we consider a slightly restricted class of TGDs and FDs: in particular, similarly to what is done in [5], we consider a class of TGDs that "does not conflict" with the FDs, and for which query answering is decidable. In the following we will make use of the notion of *freezing* a formula; given a conjunction C of atomic formulae, freezing C consists in defining a substitution σ that sends each distinct variable to a distinct constant; the frozen formula $\sigma(C)$ is a set of facts.

Definition 5. *Given a set of FDs Σ_F over a relational schema $\mathcal{R}$, a TGD θ of the form*

$$\chi(\boldsymbol{X}, \boldsymbol{Y}) \rightarrow \psi(\boldsymbol{X}, \boldsymbol{Z})$$

is a non-functional-conflicting TGD (NFCTGD) *w.r.t. Σ_F if the following conditions hold:*

1. *the database constructed by "freezing" the variables of $\psi(\boldsymbol{X}, \boldsymbol{Z})$ and considering the the obtained facts satisfies Σ_F;*
2. *for each atom $R(\boldsymbol{X}, \boldsymbol{Z})$ in ψ, and for every FD of the form $R : \mathbf{A} \rightarrow B$ in Σ_F defined on R, the symbols that are either constants or are in $\boldsymbol{X}$ (we recall that the symbols in $\mathbf{X}$ appear both sides of the TGD) are placed in a set of attributes of R that is not a superset of $\mathbf{A}$.*

Example 6. Consider a relational schema $\mathcal{R} = \{R_1/3, R_2/1, R_3/2\}$, let Σ_F a set of FDs over $\mathcal{R}$, constituted by a single FD ϕ of the form $R_1 : 1 \rightarrow 2$ (we have indicated the attributes of R_1 with integer numbers). The TGD $\theta : \ R_3(X, Y) \rightarrow R_1(X, Y, Z), R_2(Y)$ is not a NFCTGD w.r.t. Σ_F because in the first two attributes of the atom $R_1(X, Y, Z)$ are covering a superset of the

left-hand-side of ϕ, and X and Y appear in the left-hand side of θ. Moreover, the TGD $\theta_1 : \ R_3(Z,Y) \rightarrow R_1(X,Y,Z), R_1(X,b,W)$ (where b is a constant) is not a NFCTGD w.r.t. Σ_F because if we freeze its right-hand side we obtain two facts $R_1(c_X, c_Y, c_Z) \leftarrow$ and $R_1(c_X, b, c_W) \leftarrow$ that violate ϕ.

Non-functional-conflicting TGDs are a generalisation of *non-key-conflicting inclusion dependencies (NKCIDs)* [5]; similarly to NKCIDs, the NFCTGDs enjoy the following property.

Proposition 1. *Consider a relational database $\mathcal{B}$ on which a set Σ_F of FDs and a set Σ_T of NFCTGDs w.r.t. Σ_F are defined. If $\mathcal{B} \models \Sigma_F$ then $chase_{\Sigma_T}(\mathcal{B}) \models \Sigma_F$.*

Proof. The proof is by induction on the structure of $chase_{\Sigma_T}(\mathcal{B})$. At the base step, $\mathcal{B}$ satisfies Σ_F, that is true by hypothesis. As for the inductive step, consider the addition of a set of facts $f_1, \ldots, f_n$, due to the application of the TGD chase rule. By Condition 1 of Definition 5, it is straightforward to see that the facts $f_1, \ldots, f_n$ are such that no two facts f_i, f_j $(1 \leq i, j \leq n)$ that violate a FD in Σ_F. Moreover, none of the facts in $f_1, \ldots, f_n$ will violate a FD ϕ in Σ_F together with another fact f_1 already present in the segment of $chase_{\Sigma_T}(\mathcal{B})$ until the insertion of $f_1, \ldots, f_n$. In fact, let f and f_1 be of the form $R(c_1, \ldots, c_m) \leftarrow$ and $R(d_1, \ldots, d_m) \leftarrow$ respectively; by Condition 2 of Definition 5, for any FD ϕ in Σ_F of the form $R : \ \mathbf{A} \rightarrow B$, f and f_1 will never have the same values in the attributes of $\mathbf{A}$. This ends the proof of the claim.

Intuitively, from the previous property follows that, if a global database satisfies the set Σ_F FDs, and the TGDs are all non-functional-conflicting, we can ignore Σ_F w.r.t. to query answering, since the chase is indifferent to the presence of FDs.

At this point, we come to the problem of query answering in a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$. Recalling Theorem 2 and assuming that both Σ_T and $\Sigma_\mathcal{M}$ contain only NFCTGDs that are also cycle-harmless, we have that the source database $\mathcal{D}$ satisfies Σ_F by construction, since the FDs are defined only on $\mathcal{G}$. Therefore, given a global query Q, we may think of applying the algorithm TGDrewrite to Q under the set of TGDs $\Sigma_T \cup \Sigma_\mathcal{M}$, thus solving immediately the problem of query answering by means of rewriting. Unfortunately, while assuming that the TGDs in Σ_T are NFCTGDs is reasonable in practical cases, assuming the same for the dependencies in $\Sigma_\mathcal{M}$ is not. Nevertheless, the TGDs in $\Sigma_\mathcal{M}$ are the only ones having source relations appearing in them (in the left-hand side); such property ensures that we are still in luck when they are not non-functional conflicting. In fact, the problem of query answering is still decidable when TGDs in Σ_T are cycle-harmless and NFCTGDs, and those in $\Sigma_\mathcal{M}$ are cycle-harmless (this is by construction) but in general not non-functional-conflicting. The query answering technique, in this case, requires the construction of the RGD; in particular the algorithm for query answering, given a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, a source database $\mathcal{D}$ and a global query Q, consists of the following steps:

1. We build the RGD $ret(\mathcal{I}, \mathcal{D})$.
2. We check whether $ret(\mathcal{I}, \mathcal{D}) \models \Sigma_F$; if $ret(\mathcal{I}, \mathcal{D}) \not\models \Sigma_F$ we are done: in this case there is no global database satisfying both the constraints and the mapping, so $sem(\mathcal{I}, \mathcal{D}) = \emptyset$ (see e.g. [18]); therefore, query answering is trivial, since every tuple of the same arity of Q is in $cert(Q, \mathcal{I}, \mathcal{D})$. If, on the contrary, $ret(\mathcal{I}, \mathcal{D}) \models \Sigma_F$, we proceed with the following steps.
3. We calculate $\mathsf{TGDrewrite}(\mathcal{G}, \Sigma_T, Q)$.
4. We evaluate $\mathsf{TGDrewrite}(\mathcal{G}, \Sigma_T, Q)$ over $ret(\mathcal{I}, \mathcal{D})$: the result is $cert(Q, \mathcal{I}, \mathcal{D})$.

In the presence of FDs on the global schema, the construction of the retrieved global database cannot be done independently of the FDs; in fact, some of the violations of the FDs that occur during the construction of the RGD are not "real" violations; instead, they lead to the inference of equalities among newly introduced constants and constants already present in the part of the RGD constructed in previous steps. The following example illustrates this issue.

Example 7. Consider again Example 1, and suppose that the following FD is expressed over $\mathcal{G}$:

$$\mathsf{work}: \ 1 \rightarrow 2$$

Such FD imposes that a person can work at most on one project. Now, suppose to have a source database $\mathcal{D}$ with the following facts:

$$\mathsf{hasjob}(anne, maths) \leftarrow$$
$$\mathsf{teaches}(anne, databases) \leftarrow$$
$$\mathsf{infield}(databases, compScience) \leftarrow$$

According to $\mathcal{M}$, The RGD will contain the following facts:

$$\mathsf{work}(anne, p_1) \leftarrow$$
$$\mathsf{area}(p_1, maths) \leftarrow$$
$$\mathsf{work}(anne, p_2) \leftarrow$$
$$\mathsf{area}(p_2, compScience) \leftarrow$$

where p_1 and p_2 are fresh constants introduced in the construction. Note that the facts $\mathsf{work}(anne, p_1) \leftarrow$ and $\mathsf{work}(anne, p_2) \leftarrow$ violate the above FD. In this case, however, the violation is due to the two fresh constants p_1 and p_2: therefore, instead of concluding that $sem(\mathcal{I}, \mathcal{D}) = \emptyset$, we instead infer $p_1 = p_2$. Now suppose that also the following facts are in $\mathcal{D}$:

$$\mathsf{getgrant}(anne, eu123) \leftarrow$$
$$\mathsf{grantfor}(eu123, venus) \leftarrow$$

Now the RGD will have the additional fact

$$\mathsf{work}(anne, venus) \leftarrow$$

asserting that *anne* works on project *venus*; here we have another violation, but again we do not conclude that $sem(\mathcal{I}, \mathcal{D}) = \emptyset$: instead, the new facts make us infer that the project on which Anne is working is *venus*. Therefore, we have $p_1 = p_2 = venus$. All occurrencies of p_1 and p_2 in the part of $ret(\mathcal{I}, \mathcal{D})$ constructed so far need to be replaced by *venus*. ∎

Now we present a technique for constructing the RGD in the case of GLAV mapping, in the presence of FDs on the global schema.

Definition 6. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a GLAV data integration system, where a set Σ_F of FDs is defined on $\mathcal{G}$, and $\mathcal{D}$ a source database for $\mathcal{I}$. The* retrieved global database *$ret(\mathcal{I}, \mathcal{D})$ is defined constructively as follows. Consider a mapping assertion $\varphi_{\mathcal{S}}(\boldsymbol{X}, \boldsymbol{Y}) \rightarrow \varphi_{\mathcal{G}}(\boldsymbol{X}, \boldsymbol{Z})$. For each set H of facts of $\mathcal{D}$ such that there exists a substitution σ that sends the atoms of $\varphi_{\mathcal{S}}(\boldsymbol{X}, \boldsymbol{Y})$ to H: (i) we first define a substitution σ' such that $\sigma'(X_i) = \sigma(X_i)$ for each X_i in $\boldsymbol{X}$, and $\sigma'(Z_j) = z_j$ for each Z_j in $\boldsymbol{Z}$, where z_j is a fresh constant, not introduced before and not appearing in $\mathcal{D}$; (ii) we add to $ret(\mathcal{I}, \mathcal{D})$ the set of facts that are in $\sigma'(\varphi_{\mathcal{G}}(\boldsymbol{X}, \boldsymbol{Z}))$. Now, suppose that one of the added facts, say $R(t) \leftarrow$, violates a FD ϕ because of the presence of another fact $R(t_0) \leftarrow$ in the part of the RGD that has been constructed in the previous steps (t and t_0 are tuples of constants). Formally, being ϕ of the form*

$$R : \mathbf{A} \rightarrow B$$

we have $t[\mathbf{A}] = t_0[\mathbf{A}]$ and $t[B] \neq t_0[B]$. Let $t[B] = c$ and $t_0[B] = c_0$; there are different cases, that we enumerate as follows:

1. *c is a fresh constant, not appearing in $\mathcal{D}$: in this case we substitute c with c_0 (which can be either a fresh constant or a constant of $\mathcal{D}$) and proceed;*
2. *c is a constant of $\mathcal{D}$ and c_0 is a fresh constant: in this case we replace c_0 with c in all the part of the RGD that has been constructed in the previous steps and proceed.*
3. *c and c_0 are both constants appearing in $\mathcal{D}$: in this case we have $sem(\mathcal{I}, \mathcal{D}) = \emptyset$ and we stop the construction.*

Note that the construction of the RGD in this case can be done in time polynomial in the size of the source database $\mathcal{D}$: in fact, though the replacement can involve all the data in the RGD retrieved at a certain point, we have that the replacement of case 2 can be performed at most once on each constant.

The following theorem states the correctness and completeness of the above technique.

Theorem 4. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system; let Σ_T and Σ_F two sets of TGDs and FDs defined on $\mathcal{G}$ respectively, where all TGDs in Σ_T are cycle-harmless w.r.t. Σ_T and NFCTFDs w.r.t. Σ_F. If $ret(\mathcal{I}, \mathcal{D}) \models \Sigma_F$, then we have that $\mathsf{TGDrewrite}(\mathcal{G}, \Sigma_T, Q)^{ret(\mathcal{I}, \mathcal{D})} = cert(Q, \mathcal{I}, \mathcal{D})$.*

Proof (sketch). The result follows straightforwardly from Proposition 1; since $ret(\mathcal{I}, \mathcal{D}) \models \Sigma_F$, we can proceed by applying Theorem 1 as if $\Sigma_F = \emptyset$. ■‘

5 Discussion

In this paper we have addressed the problem of query answering in GLAV data integration systems, in the presence of tuple-generating dependencies and functional dependencies.

Several works in the literature address the problem of query answering under integrity constraints, both in a single database context [5, 2, 12] and in data integration [9, 20, 13, 17, 10, 6, 4]. In particular, [17] presents a technique, well supported by experimental results and theoretical foundations, for query rewriting under *conjunctive inclusion dependencies (CINDs)*; CINDs are analogous to TGDs, but in [17] a syntactic restriction on CINDs imposes that CINDs are *acyclic*, so that the problem of having a chase of infinite size (and therefore the problem of the termination of the rewriting algorithm) is not relevant. Another interesting paper about repair of database dependencies is [10]; this paper addresses the problem of integrity constraints in *data exchange*, so in this case data are to be materialised in a target schema, as well as their chase, and not accessed on-the-fly as in our virtual data integration approach. Due to the need of materialising the target schema, the class of ICs considered, namely the *weakly-acyclic* TGDs together with *equality-generating dependencies*, though certainly quite general, is such that the chase is always finite. Therefore, such class of constraints is not comparable with the one considered in our paper.

In this paper we have first addressed the problem of query answering in the presence of TGDs alone; we have recalled the algorithm TGDrewrite, introduced in [3], showing that it can be applied to a more general class of TGDs, namely the *cycle-harmless TGDs*. The result about the introduced class of TGDs allowed us to use TGDrewrite for rewriting global queries according not only to the TGDs, but also according to the mapping, that can be seen as a set of TGDs over a unified schema including both the global schema and the source schema. The rewriting algorithm can be used, of course, also in particular cases, e.g., when there are no constraints; in such a case, when the mapping is LAV instead of GLAV, our technique is similar to the algorithm MiniCon [21], which incorporates effective optimisation techniques not present in TGDrewrite; however, TGDrewrite is more general, being able to deal with GLAV mappings.

Then, we have introduced functional dependencies together with TGDs, defining a class of ICs for which the query answering problem is decidable, and providing a query answering technique based on the algorithm TGDrewrite. In this case the mapping cannot be dealt with at once, together with the ICs on the global schema; instead, the construction of the retrieved global database is required. After that, if the RGD satisfies the FDs, the form of the constraints is such that we can proceed as if there were no FDs.

As for the computational complexity, we focus our attention on the *data complexity*, i.e. the complexity w.r.t. the size of the data residing at the sources. This is the usual way of considering complexity in a database context, since the size of schemata and constraints is usually negligible with respect to the size of the data. In our case, since we solve the problem of query answering in an intensional fashion, the only phases where the data are involved are the evaluation

of the reformulated query over the source database, and the construction of the RGD (only in the presence of FDs). Both such operations can be done in time polynomial w.r.t. the size of the data.

Acknowledgements. This work was partially supported by the project MAIS, funded by the Italian Ministry of Education, University and Research. The author wishes to thank Diego Calvanese, Maurizio Lenzerini, Riccardo Rosati and Domenico Lembo for their insightful comments about this material.

References

1. Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., Reading, Massachussetts, 1995.
2. Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In *Proc. of the 18th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'99)*, pages 68–79.
3. Andrea Calì. Reasoning in data integration systems: why LAV and GAV are siblings. In *Proc. of the 14th Int. Symp. on Methodologies for Intelligent Systems (ISMIS 2003)*, pages 562–571, 2003.
4. Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Data integration under integrity constraints. *Information Systems*, 29:147–163, 2004.
5. Andrea Calì, Domenico Lembo, and Riccardo Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. of the 22nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2003)*, pages 260–271, 2003.
6. Andrea Calì, Domenico Lembo, and Riccardo Rosati. Query rewriting and answering under constraints in data integration systems. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*, pages 16–21, 2003.
7. Ashok K. Chandra and Moshe Y. Vardi. The implication problem for functional and inclusion dependencies is undecidable. *SIAM J. on Computing*, 14(3):671–677, 1985.
8. Isabel Cruz, Stefan Decker, Jérôme Euzenat, and Deborah McGuinness, editors. *The Emerging Semantic Web — Selected Papers from the First Semantic Web Working Symposium*. IOS Press, 2002.
9. Oliver M. Duschka and Michael R. Genesereth. Answering recursive queries using views. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'97)*, pages 109–116.
10. Ronald Fagin, Phokion Kolaitis, Renee J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. In *Proc. of the 9th Int. Conf. on Database Theory (ICDT 2003)*, pages 207–224.
11. Marc Friedman, Alon Levy, and Todd Millstein. Navigational plans for data integration. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI'99)*, pages 67–73, 1999.
12. Gianluigi Greco, Sergio Greco, and Ester Zumpano. A logic programming approach to the integration, repairing and querying of inconsistent databases. In *Proc. of the 17th Int. Conf. on Logic Programming (ICLP'01)*, volume 2237 of *Lecture Notes in Artificial Intelligence*, pages 348–364. Springer.

13. Jarek Gryz. Query rewriting using views in the presence of functional and inclusion dependencies. *Information Systems*, 24(7):597–612, 1999.
14. Alon Y. Halevy. Answering queries using views: A survey. *Very Large Database J.*, 10(4):270–294, 2001.
15. Jeff Heflin and James Hendler. A portrait of the semantic web in action. *IEEE Intelligent Systems*, 16(2):54–59, 2001.
16. David S. Johnson and Anthony C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. of Computer and System Sciences*, 28(1):167–189, 1984.
17. Christoph Koch. Query rewriting with symmetric constraints. In *Proc. of the 2nd Int. Symp. on Foundations of Information and Knowledge Systems (FoIKS 2002)*, volume 2284 of *Lecture Notes in Computer Science*, pages 130–147. Springer, 2002.
18. Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.
19. Maurizio Lenzerini, 2004. Personal communication.
20. Jinxin Lin and Alberto O. Mendelzon. Merging databases under constraints. *Int. J. of Cooperative Information Systems*, 7(1):55–76, 1998.
21. Rachel Pottinger and Alon Y. Levy. A scalable algorithm for answering queries using views. In *Proc. of the 26th Int. Conf. on Very Large Data Bases (VLDB 2000)*, pages 484–495, 2000.