Query Processing under GLAV Mappings for Relational and Graph Databases

Diego Calvanese
Free Univ. of Bolzano-Bozen
I-39100 Bolzano, Italy
calvanese@inf.unibz.it

Giuseppe De Giacomo
Maurizio Lenzerini
Sapienza Università di Roma
I-00185 Rome, Italy
lastname@dis.uniroma1.it

Moshe Y. Vardi
Rice University
Houston, TX 77005, U.S.A.
vardi@cs.rice.edu

ABSTRACT

Schema mappings establish a correspondence between data stored in two databases, called source and target respectively. Query processing under schema mappings has been investigated extensively in the two cases where each target atom is mapped to a query over the source (called GAV, global-as-view), and where each source atom is mapped to a query over the target (called LAV, local-as-view). The general case, called GLAV, in which queries over the source are mapped to queries over the target, has attracted a lot of attention recently, especially for data exchange. However, query processing for GLAV mappings has been considered only for the basic service of query answering, and mainly in the context of conjunctive queries (CQs) in relational databases. In this paper we study query processing for GLAV mappings in a wider sense, considering not only query answering, but also query rewriting, perfectness (the property of a rewriting to compute exactly the certain answers), and query containment relative to a mapping. We deal both with the relational case, and with graph databases, where the basic querying mechanism is that of regular path queries. Query answering in GLAV can be smoothly reduced to a combination of the LAV and GAV cases, and for CQs this reduction can be exploited also for the remaining query processing tasks. In contrast, as we show, GLAV query processing for graph databases is non-trivial and requires new insights and techniques. We obtain upper bounds for answering, rewriting, and perfectness, and show decidability of relative containment.

1. INTRODUCTION

A schema mapping is a declarative, formal specification of how data structured according to a source schema relate to data conforming to a target schema. Schema mappings are at the basis of several data management scenarios. In data integration [44], they are established between the source database and a so-called mediated schema, i.e., the unified view to be presented to the clients for querying multiple sources in a transparent way. In data exchange [10,43], schema mappings specify how the data at sources should be transformed and packed so as to transfer the desired information from the sources to the target database. Model management [16], and schema and ontology matching [31] are other examples of scenarios where schema mappings play a central role.

The interest of database research on schema mappings has been constantly growing in the last years, starting from the pioneering work in the context of data integration (see, e.g., [46, 52]). Such research work can be discussed under three coordinates: *(i)* the data model in which the schemas are expressed, *(ii)* the mapping-based tasks that have been studied, and *(iii)* the type of mechanisms used to express schema mappings.

Data models. Most of the early research results on schema mappings refer to the relational model, where the basic classes of queries used to express schema mappings are positive fragments of first-order logic, such as conjunctive queries (CQs) and unions thereof. More recently, schema mappings have been studied also in the context of semistructured data, both XML [10] and graph databases [5, 28]. Methods for extracting information from semistructured data incorporate special querying mechanisms that are not common in traditional database systems. Perhaps, the basic mechanism is that of *regular-path queries* (RPQs), which retrieve all pairs of nodes in the graph connected by a path conforming to a regular expression [6, 18].

Mapping-based tasks. Different operations and tasks concerning schema mappings have been studied in the literature. For example, in the context of data exchange, one of the basic tasks is the computation of a solution of a schema mapping M with respect to a given source database D_s, i.e., a target database D_t such that the pair (D_s, D_t) satisfies the specification M. In model management, schema mapping operations such as composition, inversion, and merge have been investigated in detail, especially with the aim of formalizing the typical operations to be carried out in the context of schema evolution. Perhaps, the most interesting class of mapping-based tasks, and the one we concentrate on in this paper, is query processing under schema mapping.

Generally speaking, *query processing under schema mapping* amounts to processing a query expressed over the target schema based on both the data in the source database, and the mapping from the source to the target. There are two basic forms of query processing under schema mappings.

Proceedings of the VLDB Endowment, Vol. 6, No. 2

The first one, originating with [3], is *query-answering*, where the goal is to compute the so-called *certain answers* to a query, i.e., the answers to the query in all target databases satisfying the schema mapping with respect to the source database. The second form, originating with [45], is *query-rewriting*, which is based on the idea of first using the mapping to reformulate the query in terms of the source alphabet, and then evaluating the resulting query, called the rewriting, over the source database. Note that, in order for the rewritten query to be evaluated at the source, it should be expressed in a specific query language, namely, the one suited for the query evaluation engine we can use for the source database. For this reason, in the rewriting approach, we fix in advance the language (or, the class of queries) used to express the rewriting, and target our rewriting technique toward such language. Obviously, in the case where different rewritings exist in the chosen class, we should also aim at computing the (or, a) "best" one. Typically, in the literature [41,44] "best" has been intended as "maximal", i.e., the rewriting ensuring the maximal set of answers among those in the class. This is in line with the idea of guaranteeing that the answers returned by the rewriting are themselves certain answers, though not all of them in general. Notice that also "minimally containing rewritings" have been considered in the literature [7,39], but not much used, especially in data integration and data exchange, since the returned answers are not guaranteed to be certain answers.

The relationship between query rewriting and answering is discussed in [27], where it is observed that, apart from seeing query rewriting as a means to perform query answering [45], also query answering can be seen under the lens of query rewriting: indeed, given a schema mapping and a source database, any algorithm for computing the certain answers to a target query based on such a mapping can be considered as a function of the source data, or, in other words, as a rewriting of the target query in terms of the source alphabet. Since such a rewriting computes exaclty the certain answers, it is often called the *perfect rewriting*. However, we have to take into account that, in general, the answering algorithm is a computable function, and it is questionable whether we can express such a function in terms of a given query language. It is thus of interest, given a maximal rewriting in a class of queries, to be able to check whether such a rewriting is *perfect*, i.e., it always computes the certain answers [27]. So, perfectness is another relevant task in query processing under schema mappings.

The problem of query containment is fundamental to many aspects of database systems, including query optimization, determining independence of queries from updates, and rewriting queries using views. However, as argued in [50], in data integration, the standard notion of query containment does not suffice. This is why in [50] the authors define *relative containment*, which formalizes the notion of query containment relative to the sources available to the data-integration system. Relative contanment is indeed a further relevant task concerning schema mappings.

Types of mappings. From a syntactic point of view, a schema mapping is constituted by a set of mapping assertions, each one specifying a relationship between the source and the target. In its most general form, an assertion specifies a correspondence between a view over the source and a view over the target. In the so-called *sound* schema mappings, which are the ones we deal with in this paper, the correspondence is containment: the answers to the view on the source are a subset of the answers to the view on the target. Obviously, the expressive power of this mechanism depends on the expressivity of the query language used in the view definitions. Since the early work on schema mappings in data integration [44], various restricted forms of such assertions have been considered, in particular GAV and LAV. The most obvious form of mappings are GAV (*Global-As-Views*) mappings, which associate with each element of the (global) target schema one view over the source schema, in a way analogous to how views are defined in standard databases. Indeed, processing queries under GAV mappings amounts essentially to substituting (global) view symbols with their definition in terms of the sources. With a surprising stance, [45] reversed this conception, by proposing to describe the content of sources in terms of queries over a (virtual) target schema, giving rise to so-called LAV (*Local-As-Views*) mappings. In this way, precise semantics is assigned to sources. The ability of easily processing LAV mappings for relational data gave new impetus to data integration in the '90's, on the basis of virtualizing the target schema, while leaving the sources untouched. In the years, it became clear that a pure LAV approach, while compelling from a theoretical point of view, is not practically realizable, because schemas of legacy sources are typically too dirty for allowing a clean description in terms of a query over the target. This is one of the reasons why recent work on data integration and virtually all work on data exchange focuses on *GLAV mappings*, which generalize both GAV and LAV by mapping queries over the source schema to queries over the target schema. GLAV mappings overcome the limitations of LAV by allowing the use of queries over the sources to cleanse source data to be related to the target schema.

The importance of GLAV mappings in modern information systems is also emphasized by the work on ontology-based data access [51], where the domain of interest is formalized by means of an ontology, while the data remain at the sources. In such a setting, the correct relationship between the sources and the concepts in the ontology often can only be expressed by means of a view-to-view mechanism.

Contribution. Most of the research on query processing under schema mapping in data integration has concentrated on GAV mappings, and on LAV mappings under the label of view-based query processing (see, for instance the survey in [41]). In data integration, GLAV mappings have been specifically considered in [19, 38, 42], but only for the case of relational databases, and only for query answering and query rewriting. GLAV schema mappings have been mainly investigated in data exchange: in [9, 32, 33, 47] the emphasis is on providing foundations for data exchange systems based on schema mappings, whereas in [11–14, 34–36, 48] the goal is to study operators on schema mappings relevant to model management, notably, composition, merge, and inverse. For XML data, data exchange based on GLAV mappings has been studied in [8, 10].

In this paper we thoroughly investigate query processing under GLAV mappings. We start by reviewing and extending the work on relational databases, where the basic querying mechanism used both in mappings and user queries are CQs. In this case, it is known that query answering can be done by splitting the GLAV mapping into a GAV mapping followed by a LAV mapping over an intermediate alphabet, as shown in [38]. Using this observation, together with

the possibility of doing query answering through maximal rewriting, allows us to easily reconstruct a complete picture of query processing under GLAV mappings for CQs in the relational case, including, apart from query answering and query rewriting, also perfectness and relative containment.

We then turn to graph databases. Here, the basic querying mechanism is that of regular path queries (RPQs), which allow for a controlled use of recursion necessary to navigate graphs of unbounded size [2,22]. Notice that query processing in the presence of recursion has also been studied in the context of Datalog [15, 20, 29, 30]. However, when we consider these works in the setting of query processing under mappings, the severe restrictions on the use of recursion that are imposed to gain decidability, make them inapplicable to the case of RPQ mappings.

Query processing under RPQ LAV and GAV mappings has been studied extensively [25,27,40]. However, while for CQs the results on LAV and GAV can be readily adapted to deal also with GLAV mappings, interestingly, for RPQs this is not the case. In this paper we obtain tight upper bounds for answering, rewriting, and perfectness, and show decidability of relative containment, but such results are non-trivial and require new insights and techniques.

The paper is organized as follows. In Section 2 we present our framework for query processing under schema mappings, and illustrate the basic techniques developed for the case of LAV mappings. In Section 3, we show that for the relational case these results can be extended easily to GLAV CQ mappings. Then we present our main contributions, which consist in novel techniques for the case of graph databases for the various mapping-based tasks, namely query answering and rewriting (Section 4), perfectness (Section 5), and relative containment (Section 6). Finally, Section 7 concludes the paper.

2. BACKGROUND

In this section, after some preliminary notions, we introduce the forms of query processing under schema mappings that we investigate in this paper, and we summarize the techniques and results for the known case of LAV mappings.

2.1 Databases and queries

We consider databases as finite relational structures. A *(relational) schema* is a set of relation symbols, each with an associated *arity*. Let Σ be an alphabet of relation symbols, each with an associated arity. A *finite relational structure* (or simply *structure*) D over Σ is a pair $(\Delta^D, \cdot^D)$, where Δ^D is a finite domain and $\cdot^D$ is a function that assigns to each relation symbol in $a \in \Sigma$ a relation a^D, also denoted by $a(D)$, of the appropriate arity over Δ^D. Given a *query* q over Σ, we denote by $q(D)$ the result of evaluating q over D.

We consider the case of *conjunctive queries* (CQs) over relational databases. In particular, we deal with standard CQs without equalities and without constants. A CQ is a conjunction of atoms whose variables are either free (called *distinguished variables*) or existentially quantified. They correspond to the well-known select-project-join fragment of relational algebra. If we add union to CQs, we obtain *unions of conjunctive queries* (UCQs).

We also consider the case of regular path queries (RPQs) over graph structured data. In such a setting, a database consists of binary relations only, and can be interpreted as a finite graph whose nodes represent objects and whose edges are labeled by elements from the alphabet Σ of binary relation symbols [1,17].

A *regular-path query* (RPQ) over an alphabet Σ of binary relation symbols is expressed as a regular expression or a deterministic or non-deterministic finite-state automaton over Σ (DFA and NFA, respectively). When evaluated on a database D over Σ, an RPQ q computes the set of pairs of objects connected in D by some path labeled with a word in the regular language $\mathcal{L}(q)$ defined by q.

A query q over Σ is *empty* if $q(D) = \emptyset$ for every database D over Σ. Given two queries q_1 and q_2 over Σ, we say that q_1 is *contained in* q_2, denoted $q_1 \sqsubseteq q_2$, if $q_1(D) \subseteq q_2(D)$, for every database D over Σ. Moreover, q_1 is *equivalent to* q_2, denoted $q_1 \equiv q_2$, if $q_1 \sqsubseteq q_2$ and $q_2 \sqsubseteq q_1$.

An *atomic* query is a query of the form $\{\vec{x} \mid r(\vec{x})\}$, where r is a relation symbol, and $\vec{x}$ is a tuple of pairwise distinct variables of the same arity as r. When clear from the context, we denote such an atomic query simply as r. We also use $r(\vec{x})$ when we need to make the distinguished variables $\vec{x}$ explicit.

2.2 Query processing under schema mappings

We consider databases over a fixed, countably infinite domain Δ and over two schemas, a source schema Σ and a target schema Γ. We call databases over Σ *source databases* and databases over Γ *target databases*. When not specified otherwise, D_s will denote a source database and D_t a target database. A *(GLAV) mapping* from Σ to Γ is a finite set M of *mapping assertions* of the form $e \rightsquigarrow f$, where e is a query over Σ and f is a query over Γ of the same arity as e.

A pair (D_s, D_t) of source and target databases *satisfies* a mapping assertion $e \rightsquigarrow f$, if $e(D_s) \subseteq f(D_t)$, and satisfies a mapping M, denoted $(D_s, D_t) \models M$, if it satisfies every mapping assertion in M.

We consider also two special forms of mappings, in which either the source queries or the target queries are atomic. More precisely:

- a *LAV mapping* M_L is a mapping all of whose assertions have the form $a \rightsquigarrow f$, where a is an atomic query, and for each source symbol a there is exactly one mapping assertion involving a. For each source symbol a, $M_L[a]$ denotes the target query that M_L associates with a, i.e., $a \rightsquigarrow M_L[a] \in M_L$. Moreover, for a target database D_t, $M_L(D_t)$ denotes the source database in which the extension of each source symbol a is given by $M_L[a](D_t)$.

- a *GAV mapping* M_G is a mapping all of whose assertions have the form $e \rightsquigarrow b$, where b is an atomic query, and for each target symbol b there is exactly one mapping assertion involving b. For each target symbol b, $M_G[b]$ denotes the source query that M_G associates with b, i.e., $M_G[b] \rightsquigarrow b \in M_G$. Moreover, for a source database D_s, $M_G(D_s)$ denotes the target database in which the extension of each target symbol b is given by $M_G[b](D_s)$.

Given a mapping M, a source database D_s, and a query q_t over Γ, the set of *certain answers* (under sound mappings) to q_t w.r.t. M and D_s is the set of tuples that are in $q_t(D_t)$, for every target database D_t such that $(D_s, D_t) \models M$. Given a mapping M and a target query q_t, the *perfect rewriting* of q_t

w.r.t. M, denoted by $cert[q_t, M]$, is the (source) query that, for every source database D_s, returns the set of certain answers to q_t w.r.t. M and D_s.[1] Hence, $cert[q_t, M](D_s)$, considered as the evaluation of the perfect rewriting $cert[q_t, M]$ over the source database D_s, is the set of certain answers to q_t w.r.t. M and D_s.

DEFINITION 1. ANS *is the following decision problem: given a source database D_s, a mapping M, a target query q_t, and a tuple $\vec{c}$ of values, decide whether $\vec{c} \in cert[q_t, M](D_s)$.*

Mapping-based containment allows us to compare two queries, each one either over the source or over the target schema, taking into account a (GLAV) mapping between source and target.

DEFINITION 2. *Let M be a mapping, q_s^1, q_s^2 two source queries, and q_t^1, q_t^2 two target queries.*

- $q_s^1 \sqsubseteq_M q_t^2$ *if for every source database D_s, we have that $q_s^1(D_s) \subseteq cert[q_t^2, M](D_s)$.*
- $q_t^1 \sqsubseteq_M q_s^2$ *if for every source database D_s, we have that $cert[q_t^1, M](D_s) \subseteq q_s^2(D_s)$.*
- $q_t^1 \sqsubseteq_M q_t^2$ *if for every source database D_s, we have that $cert[q_t^1, M](D_s) \subseteq cert[q_t^2, M](D_s)$.*

In all three cases[2], we say that the left-hand side query is M-contained *in the right-hand side query.*

We observe that both CQs and RPQs are positive queries, and that in the above definition we have assumed to deal only with such kinds of queries. Indeed, for arbitrary queries, we would need to restrict the quantification only to those source databases D_s such that there exists a target database D_t with $(D_s, D_t) \models M$.

Notice that the last case of the above definition, where two target queries are compared considering their certain answers w.r.t. a given mapping, is called *relative containment*, and has been studied in the context of LAV mappings [49]. The corresponding decision problem is defined as follows:

DEFINITION 3. RCON *is the following decision problem: given two target queries q_t^1 and q_t^2, and a mapping M, decide whether $q_t^1 \sqsubseteq_M q_t^2$.*

We now turn our attention to query rewriting, and define it in terms of mapping-based containment.

DEFINITION 4. *A source query q_s is a* (sound) rewriting *of a target query q_t w.r.t. a mapping M if $q_s \sqsubseteq_M q_t$.*

The fact that the rewriting is "sound" is reflected in the fact that we require q_s to be M-contained in q_t, rather than equivalent (modulo M). In this paper we are interested mainly in sound rewritings, which we call simply rewritings in the following.

A fundamental problem we are interested in is establishing the existence of a non-empty rewriting.

[1]The perfect rewriting is not to be confused with an *exact* rewriting, which, when it exists, returns for every source database exactly the answers of the original query (and not only the certain answers) [27].

[2]We do not consider the fourth case, in which we compare two source queries, since in this case the mapping is irrelevant, and M-containment corresponds to ordinary containment.

DEFINITION 5. REW *is the following decision problem: given a target query q_t and a mapping M, decide whether there exists a non-empty rewriting of q_t w.r.t. M.*

In query rewriting, we are also interested in computing rewritings that capture the original query at best. Let $\mathcal{C}$ be a query class. A source query q_s in $\mathcal{C}$ is a *$\mathcal{C}$-maximally contained rewriting* of a target query q_t in $\mathcal{C}$ w.r.t. a mapping M if *(i)* q_s is a rewriting of q_t w.r.t. M, and *(ii)* there is no source query q_s' in $\mathcal{C}$ that is a rewriting of q_t w.r.t. M and such that $q_s \sqsubseteq q_s'$ and not $q_s \equiv q_s'$.

Notice that in the above definition we do not necessarily require that the mapping M is expressed by means of queries that are in $\mathcal{C}$ or that the input query is in $\mathcal{C}$. For example, we might consider $\mathcal{C}$ to be UCQs, while M might consist of mapping assertions $e \rightsquigarrow f$ in which e and f have to be CQs.

Perfectness is the problem of checking whether a given rewriting of a target query w.r.t. a mapping, is equivalent to the perfect rewriting. We can rephrase the definition of perfect rewriting in terms of mapping-based containment.

DEFINITION 6. *A source query q_s is a* perfect rewriting *of a target query q_t w.r.t. a mapping M if both $q_s \sqsubseteq_M q_t$ (i.e., q_s is a rewriting of q_t), and $q_t \sqsubseteq_M q_s$.*

The following definition formalizes in terms of mapping-based containment the problem of checking whether a given rewriting is indeed perfect.

DEFINITION 7. PERF *is the following decision problem: given a mapping M, a target query q_t, and a source query q_s, which is assumed to be a rewriting of q_t w.r.t. M, decide whether $q_t \sqsubseteq_M q_s$.*

In the following, we study the complexity of the above problems both under *data complexity* [53], where we assume that the input in only the source database, while query and mappings are fixed, and under *combined complexity*, where the input is the source database, the query, and the mappings.

EXAMPLE 1. *We illustrate the above definitions with a simple example. Consider a source schema $\Sigma = \{a_1, a_2, a_3\}$ and a target schema $\Gamma = \{b_1, b_2\}$, where all source and target symbols are binary relations. Consider the GLAV RPQ mapping M with mapping assertions*

$$\begin{array}{rcl} a_1 \cdot a_2^* & \rightsquigarrow & b_1 \cdot b_1^* \cdot b_2 \\ a_3 & \rightsquigarrow & b_2 \end{array}$$

and the target queries

$$\begin{array}{rcl} q_t^1 & = & b_1^* \cdot b_2^* \cdot b_2 \\ q_t^2 & = & b_1 \cdot b_1^* \cdot b_2^*. \end{array}$$

Given the source database

$$D_s = \{a_1(1,2),\ a_2(2,3),\ a_2(3,4),\ a_3(4,4)\},$$

it is possible to see that the certain answers to q_t^1 and to q_t^2 w.r.t. M are

$$\begin{array}{rcl} cert[q_t^1, M](D_s) & = & \{(1,4),(4,4)\} \\ cert[q_t^2, M](D_s) & = & \{(1,4)\}. \end{array}$$

Turning to rewriting, the RPQ-maximally contained rewriting of q_t^1 and of q_t^2 w.r.t. M are respectively

$$\begin{array}{rcl} r_s^1 & = & a_1 \cdot a_2^* \cdot a_3^* + a_3^* \cdot a_3 \\ r_s^2 & = & a_1 \cdot a_2^* \cdot a_3^*. \end{array}$$

It can be shown that both of them are perfect rewritings. Finally, turning to mapping-based containment, while q_t^1 and q_t^2 are incomparable as queries over the target schema, when we take into account the mapping, we get that

$$q_t^2 \sqsubseteq_M q_t^1.$$

Note that, in this easy case, due to the perfectness of the RPQ-rewriting, all query processing tasks can be carried out by directly comparing and evaluating RPQs. As we will show later, this does not hold in general.

2.3 The case of LAV mappings

For the case of (U)CQs under LAV mappings, a fundamental result [27, 45] establishes that the perfect rewriting $cert[q_t, M_L]$ of a (U)CQ q_t w.r.t. a LAV mapping M_L can be effectively computed as a UCQ, and hence coincides with the UCQ-maximally contained rewriting of q_t. Such UCQ consists of a set of CQs, each of polynomial length. The number of such CQs is worst-case exponential in the size of the largest CQ in q_t and of the LAV mapping. Hence, query answering can be simply done by evaluating the perfect rewriting over the source database. We get that ANS is in AC^0 in data complexity, which is the data complexity of evaluating UCQs over a database [4]. Checking whether there exists a non-empty rewriting corresponds to verifying whether the perfect rewriting is non-empty. Analogously, checking perfectness and the various forms of mapping-based containment can be done directly through standard UCQ containment using the computed perfect rewriting.

As shown in [23, 26, 27], for LAV RPQ mappings certain answers to an RPQ can be characterized in terms of constraint satisfaction. We recall here this relationship, since we build on it for our technical development for graph databases.

A *constraint-satisfaction problem (CSP)* is traditionally defined in terms of a set of variables, a set of values, and a set of constraints, and asks whether there is an assignment of values to the variables that satisfies the constraints. A characterization of CSP can be given in terms of homomorphisms between relational structures [37]. Here we consider relational structures whose relations are of arbitrary arity.

A *homomorphism* $h : A \to B$ between two relational structures A and B over the same alphabet is a mapping $h : \Delta^A \to \Delta^B$ such that, if $(c_1, \ldots, c_n) \in r(A)$, then $(h(c_1), \ldots, h(c_n)) \in r(B)$, for every relation symbol r in the alphabet. Let $\mathcal{A}$ and $\mathcal{B}$ be two classes of structures. The *(uniform) constraint-satisfaction problem* $CSP(\mathcal{A}, \mathcal{B})$ is the following decision problem: given a structure $A \in \mathcal{A}$ and a structure $B \in \mathcal{B}$ over the same alphabet, is there a homomorphism $h : A \to B$? When $\mathcal{B}$ consists of a single structure B and $\mathcal{A}$ is the set of all structures over the alphabet of B, we get the so-called *non-uniform* constraint-satisfaction problem, denoted by $CSP(B)$, where B is fixed and the input is just a structure $A \in \mathcal{A}$. As usual, we use $CSP(B)$ also to denote the set of structures A such that there is a homomorphism from A to B. From the very definition of CSP it follows directly that every $CSP(\mathcal{A}, \mathcal{B})$ problem is in NP.

The relationship between non-uniform CSP and answering RPQs for LAV mappings is based on the notions of constraint template, associated with the query and LAV mapping, and constraints instance, associated with the source database. Formally, given a source schema Σ and a target schema Γ consisting of binary relation symbols, an RPQ q over Γ, and a LAV RPQ mapping M_L between Σ and Γ, the *constraint template* $CT(q, M_L)$ of q with respect to M_L is the relational structure C defined as follows.

- The alphabet of C is $\Sigma \cup \{u_i, u_f\}$, where u_i and u_f are unary relation symbols.
- Let $A_q = (\Gamma, S_q, S_q^0, \varrho_q, F_q)$ be an NFA for q, where Γ is the alphabet, S_q the set of states, S_q^0 the set of initial states, ϱ_q the transition relation, and F_q the set of final states. The structure $C = (\Delta^C, \cdot^C)$ is given by:
  - $\Delta^C = 2^{S_q}$;
  - $\sigma \in u_i(C)$ iff $S_q^0 \subseteq \sigma$;
  - $\sigma \in u_f(C)$ iff $\sigma \cap F_q = \emptyset$;
  - for each source symbol $a \in \Sigma$, we have that $(\sigma_1, \sigma_2) \in a(C)$ iff there exists a word $w \in \mathcal{L}(M_L[a])$ such that $\varrho_q(\sigma_1, w) \subseteq \sigma_2$ (we consider ϱ_q extended to sets of states in the usual way).

Notice that the latter condition requires that $(\sigma_1, \sigma_2) \in a^C$ iff there exists a word $w = t_1 \cdots t_k \in \mathcal{L}(M_L[a])$ and a sequence $S_0, \ldots, S_k$ of subsets of S_q such that the following hold:

1. $S_0 = \sigma_1$ and $S_k = \sigma_2$, and
2. if $s \in S_i$ and $(s, t_{i+1}, s') \in \varrho_q$ then $s' \in S_{i+1}$, for $0 \leq i < k$.

Intuitively, the constraint template represents for each source symbol a, how the states of the NFA A_q for q change when we follow database edges according to (paths specified by) words in $\mathcal{L}(M_L[a])$. Specifically, the last condition above corresponds to saying that a pair of sets of states (σ_1, σ_2) is in $a(C)$ if and only if there is some word w in $\mathcal{L}(M_L[a])$ such that if A_q performs transitions according to w starting from a state in σ_1, it will always end in a state in σ_2. Moreover, the sets of states in $u_i(C)$ contain all initial states of A_q, while the sets of states in $u_f(C)$ do not contain any final state of A_q. This takes into account that we aim at characterizing counterexamples to query answering, and hence we are interested in not getting to a final state of A_q, regardless of the initial state from which we start and how we follow transitions.

The existence of a word $w \in \mathcal{L}(M_L[a])$ such that $\varrho_q(\sigma_1, w) \subseteq \sigma_2$ can be verified in polynomial space in the size of q, and in fact in nondeterministic logarithmic space in the size of $M_L[a]$ [23, 26, 27].

Given a source database D_s with domain Δ^{D_s} and a pair of objects c, d, the *constraint instance* (D_s, c, d) is the structure $I = (\Delta^I, \cdot^I)$ over the alphabet $\Sigma \cup \{u_i, u_f\}$ defined as follows:

- $\Delta^I = \Delta^{D_s} \cup \{c, d\}$;
- $a(I) = a(D_s)$, for each $a \in \Sigma$;
- $u_i(I) = \{c\}$, and $u_f(I) = \{d\}$.

The following theorem provides the characterization of answering RPQs w.r.t. LAV mappings in terms of CSP.

THEOREM 1 ([23]). *Let q be a target RPQ, M_L a LAV RPQ mapping, D_s a source database, and c, d a pair of objects. Then, $(c, d) \notin cert[q, M_L](D_s)$ if and only if there is a homomorphism from (D_s, c, d) to $CT(q, M_L)$.*

From this result it is immediate to derive that ANS is in coNP in data complexity and in PSPACE in combined complexity [23]. It turns out that these bounds are tight [21].

The characterization in terms of CSP can also be exploited for the various forms of mapping-based containment, and hence for rewriting, perfectness, and relative containment.

For query rewriting [27], we observe that a word $w_s = a_1 \cdots a_n$ (considered as a source query) is *not* a rewriting of a target query q_t w.r.t. a LAV mapping M_L iff $w_s \not\sqsubseteq_M q_t$ iff there is a homomorphism from the structure (D_{w_s}, c_0, c_n) to $CT(q_t, M_L)$, where D_{w_s} is a source database of the form $\{a_1(c_0, c_1), a_2(c_1, c_2), \ldots, a_n(c_{n-1}, c_n)\}$, for some constants $c_0, c_1, \ldots, c_n$. We can construct a "counterexample" NFA A_{q_t, M_L} that guesses such a homomorphism on the fly and checks it. So A_{q_t, M_L} accepts word w_s iff w_s is *not* a rewriting of q_t, and $A^c_{q_t, M_L}$, the complement of A_{q_t, M_L}, accepts all words w_s that are rewritings of q_t. Thus, $A^c_{q_t, M_L}$ describes the maximal rewriting, and it is nonempty iff a rewriting exists. An algorithms based on checking non-emptiness of $A^c_{q_t, M_L}$, gives us the same tight EXPSPACE upper bound for REW originally established in [24].

For perfectness, let M_L be a LAV RPQ mapping, q_s a source RPQ, and q_t a target RPQ. To check perfectness, i.e., whether $cert[q_t, M_L] \sqsubseteq q_s$, we define two constraint templates: $CT(q_s, I)$ for q_s, based on the identity mapping I, and $CT(q_t, M_L)$, for q_t. We know that $(c, d) \notin cert[q_t, M_L](D_s)$, for some source database D_s, iff there is a homomorphism from the constraint instance (D_s, c, d) to $CT(q_t, M_L)$. Also, since $q_s(D_s) = cert[q_s, I](D_s)$, we have that $(c, d) \notin cert[q_s, I](D_s)$ iff there is a homomorphism from (D_s, c, d) to $CT(q_s, I)$. We want to check that for every D_s, $(c, d) \notin cert[q_s, I](D_s)$ implies $(c, d) \notin cert[q_t, M_L](D_s)$. As shown in [26], we can consider $CT(q_s, I)$ as a source database, so the condition holds iff there is a homomorphism from $CT'(q_s, I)$ to $CT(q_t, M_L)$ for every structure $CT'(q_s, I)$ that is identical to $CT(q_s, I)$, except that the unary relations u_i and u_f are restricted to a singleton. From this, we obtain a NEXPTIME upper bound for checking perfectness, and this bound turns out to be tight [26].

For relative containment between two target queries q_t^1 and q_t^2, we can argue similarly as for perfectness, except that we have to consider for both queries the constraint template associated to the LAV mapping. Specifically, we want to test whether $cert[q_t^1, M_L]$ is contained is $cert[q_t^2, M_L]$, that is, for every source database D_s, if $(c, d) \in cert[q_t^1, M_L](D_s)$, then $(c, d) \in cert[q_t^2, M_L](D_s)$. Equivalently, if $(c, d) \notin cert[q_t^2, M_L](D_s)$, then $(c, d) \notin cert[q_t^1, M_L](D_s)$. Equivalently, if there is a homomorphism from (D_s, c, d) to $CT(q_t^2, M_L)$, then there is a homomorphism from (D_s, c, d) to $CT(q_t^1, M_L)$. As shown in [26], the last condition holds iff there is a homomorphism from each $CT'(q_t^2, M_L)$ to $CT(q_t^1, M_L)$, where again $CT'(q_t^2, M_L)$ is obtained from $CT(q_t^2, M_L)$ by selecting specific constants to represent the unary relations u_i and u_f. Also in this case, we obtain a tight NEXPTIME upper bound.

3. THE CASE OF GLAV MAPPINGS FOR RELATIONAL DATABASES

Notice that, in general, $cert[q_t, M]$ is not a query that can be expressed in the same query language as q_t and M. However, given a mapping M, a source database D_s, and a target query q_t, we can directly compute the certain answers $cert[q_t, M](D_s)$ w.r.t. a GLAV mapping M by reducing this problem to the problem of "materializing" intermediate views and computing the certain answers w.r.t. a LAV mapping and the data in the materialized views. We proceed as follows:

1. Introduce for each mapping assertion $e \rightsquigarrow f$ of M a fresh (intermediate view) symbol v, of the same arity as e and f. Let $\mathcal{V}$ be the resulting (view) schema.
2. Split each mapping assertion $e \rightsquigarrow f$ of M into a GAV assertion $e \rightsquigarrow v$ and a LAV assertion $v \rightsquigarrow f$, where v is the view symbol introduced for $e \rightsquigarrow f$. Let M_G and M_L respectively denote the resulting sets of GAV and LAV mapping assertions[3].
3. For each GAV assertion $e \rightsquigarrow v$ in M_G, materialize the intermediate view v with the result of $e(D_s)$. The resulting database $M_G(D_s)$ is expressed over $\mathcal{V}$.
4. Compute $cert[q_t, M_L](M_G(D_s))$, i.e., the certain answers of q_t w.r.t. the (LAV) mapping M_L and $M_G(D_s)$.

The correctness of the above algorithm has been shown in [38] for CQs. More generally, it is an immediate consequence of the following result.

THEOREM 2. *Let M be a GLAV mapping split into a GAV mapping M_G and a LAV mapping M_L, and let q_t be a target query. Then, for each source database D_s, we have that $cert[q_t, M](D_s) = cert[q_t, M_L](M_G(D_s))$.*

PROOF. We show both inclusions separately. Let D_v denote $M_G(D_s)$, which is a database over the set $\mathcal{V}$ of intermediate view symbols.

"$\supseteq$" Consider a tuple $\vec{c} \in cert[q_t, M_L](D_v)$, and a target database D_t such that $(D_s, D_t) \models M$. We have to show that $\vec{c} \in q_t(D_t)$. Since $\vec{c} \in cert[q_t, M_L](D_v)$, it suffices to show that $(D_v, D_t) \models M_L$. For a mapping assertion $v \rightsquigarrow f$ in M_L, let $e \rightsquigarrow f$ be the corresponding assertion in M, and $e \rightsquigarrow v$ the corresponding assertion in M_G. Then, $v(D_v) = e(D_s)$ and since $e(D_s) \subseteq f(D_t)$, we also have that $v(D_v) \subseteq f(D_t)$. It follows that $(D_v, D_t) \models M_L$.

"$\subseteq$" Consider a tuple $\vec{c} \in cert[q_t, M](D_s)$, and a target database D_t such that $(D_v, D_t) \models M_L$. We have to show that $\vec{c} \in q_t(D_t)$. Since $\vec{c} \in cert[q_t, M](D_s)$, it suffices to show that $(D_s, D_t) \models M$. For a mapping assertion $e \rightsquigarrow f$ in M, let $e \rightsquigarrow v$ be the corresponding assertion in M_G, and $v \rightsquigarrow f$ the corresponding assertion in M_L. Then, $v(D_v) = e(D_s)$ and since $v(D_v) \subseteq f(D_t)$, we also have that $e(D_s) \subseteq f(D_t)$. It follows that $(D_s, D_t) \models M$. □

Theorem 2 implies that we can indeed compute the certain answers w.r.t. a GLAV mapping as indicated above, by materializing intermediate views and computing the certain answers w.r.t. a LAV mapping and the data in the materialized views.

In the case of GLAV CQ mappings, the perfect rewriting of a (U)CQ w.r.t. a LAV mapping can itself be expressed as a UCQ. Hence, we are able to deal separately with the GAV and the LAV parts of a GLAV mapping not only for query answering, but also for rewriting, perfectness, and relative containment. It follows that addressing all these tasks for GLAV CQ mappings and (U)CQ target queries is essentially straightforward, as we show in the rest of this section. For graph databases, instead, the perfect rewriting of an RPQ

[3]Note that in M_G the view symbols play the role of target symbols, while in M_L they play the role of source symbols.

w.r.t. an RPQ LAV mapping cannot be directly represented as a query expression. This makes the situation significantly more complicated, as we will show in the rest of the paper.

3.1 Query answering

Theorem 2 gives us a PTIME upper bound in data complexity for query answering for (U)CQs, since the materialization of the intermediate views requires polynomial time in the size of the source database. However, this bound can be refined by avoiding this materialization. This is done by computing explicitly the perfect rewriting and directly evaluating it over the source database. Indeed, as we show next, the perfect rewriting of a source (U)CQ w.r.t. a GLAV CQ mapping is a UCQ. This is an immediate consequence of the fact that the perfect rewriting of a (U)CQ w.r.t. a LAV mapping is a UCQ, and that we can make use of the standard notion of query unfolding w.r.t. a GAV mapping.

For a (U)CQ q_v expressed over $\mathcal{V}$, let $unfold[q_v, M_G]$ denote the *unfolding* of q_v w.r.t. a GAV CQ mapping M_G, i.e., the UCQ obtained from q_v by replacing each occurrence in q_v of a view symbol v with the query $M_G[v]$ [4]. Note that such an unfolding in general requires the introduction of fresh variables for each occurrence of v in q_v. The following result is immediate.

LEMMA 3. *Let q_v be a (U)CQ over the alphabet $\mathcal{V}$ of intermediate views. Then for each source database D_s we have that $q_v(M_G(D_s)) = unfold[q_v, M_G](D_s)$.*

The next theorem shows that, if q_v is the perfect rewriting of a target query q_t w.r.t. M_L, then by unfolding it w.r.t. M_G, we get the perfect rewriting of q_t w.r.t. M.

THEOREM 4. *Let M be a GLAV CQ mapping split into a GAV mapping M_G and a LAV mapping M_L, and let q_t be a target (U)CQ. Then $cert[q_t, M] \equiv unfold[cert[q_t, M_L], M_G]$.*

PROOF. We have to show that $cert[q_t, M](D_s) = unfold[cert[q_t, M_L], M_G](D_s)$, for every source database D_s. But this is an immediate consequence of Theorem 2 and Lemma 3, by observing that $cert[q_t, M_L]$ is a query expressed over the view alphabet $\mathcal{V}$. □

We observe that, although we stated the previous result only for CQ mappings and UCQ target queries, it actually holds whenever it is possible to explicitly compute the perfect rewriting of the target query w.r.t. the LAV mapping as a query expression that can then be unfolded w.r.t. the GAV mapping.

By exploiting that the perfect rewriting $cert[q_t, M_L]$ of a (U)CQ q_t w.r.t. the LAV mapping M_L is a UCQ that can be effectively computed (see Section 2.3), we obtain an optimal upper bound for query answering.

THEOREM 5. *For GLAV CQ mappings and (U)CQ queries,* ANS *is in AC^0 in data complexity and* NP*-complete in combined complexity.*

PROOF. We first observe that the unfolding of a UCQ w.r.t. a CQ GAV mapping is still a UCQ. Hence, for a target query q_t and a GLAV mapping M, by Theorem 4, we have that $cert[q_t, M]$ is a UCQ. For data complexity, $cert[q_t, M]$ is a fixed query, so its size does not matter. Then, the AC^0 upper bound in data complexity is an immediate consequence of the fact that UCQs can be evaluated over a source database in AC^0 in data complexity [4].

For the NP upper bound in combined complexity, we observe that to check whether for a tuple $\vec{c}$ of constants we have that $\vec{c} \in cert[q_t, M](D_s)$, it suffices to guess a CQ q (of polynomial size) in $cert[q_t, M]$ and a homomorphism h from q to D_s, and check that h maps the tuple $\vec{x}$ of distinguished variables of q to $\vec{c}$. The NP lower bound in combined complexity is an immediate consequence of the same lower bound for plain CQ evaluation over a database. □

We observe that the above upper bound in data complexity can also be derived by applying the forward chaining technique in [15].

3.2 Query rewriting

For (U)CQ queries and LAV CQ mappings, as observed in Section 2.3, the UCQ-maximally contained rewriting coincides with the perfect rewriting, and can be effectively computed with the complexity bounds mentioned above. Hence, as follows from Theorem 4 for the perfect rewriting, we can obtain a UCQ-maximally contained rewriting of a CQ q_t w.r.t. a GLAV CQ mapping M as follows: *(i)* split M into a GAV part M_G and a LAV part M_L; *(ii)* compute the UCQ-maximally contained rewriting r_v of q_t w.r.t. the LAV mapping M_L, which is expressed over the intermediate view symbols $\mathcal{V}$; *(iii)* unfold r_v w.r.t. the GAV mapping M_G, to obtain a source query r_s. From this we can derive a tight upper bound for checking non-emptiness of the rewriting.

THEOREM 6. *For GLAV CQ mappings and (U)CQ queries,* REW *is* NP*-complete.*

PROOF. Since the rewriting r_s computed by the above algorithm is obtained as the unfolding of the query r_v expressed over the intermediate view symbols $\mathcal{V}$, it is non-empty if and only if r_v is non-empty. The query r_v is a UCQ consisting of a set of CQs, each of length bounded by the length of the maximum CQ in q_t. We can check in NP whether r_v is non-empty by simply guessing a CQ over $\mathcal{V}$ within the length bound, and checking whether its unfolding w.r.t. the LAV mapping M_L is contained in q_t, which can also be done in NP. □

3.3 Perfectness

We can check perfectness for GLAV CQ mappings and (U)CQ queries, by exploiting Theorem 4, which states that the perfect rewriting of a target (U)CQ is a source UCQ that can be effectively computed, since it coincides with the UCQ-maximal rewriting. Hence, given a GLAV CQ mapping M, a source UCQ q_s, and a target UCQ q_t, to check perfectness, we compute the UCQ-maximal rewriting r_s of q_t with respect to M, which is a source UCQ, and check whether $r_s \sqsubseteq q_s$. Hence, we get the following upper bound, which matches the one for the LAV case [26].

THEOREM 7. *For GLAV CQ mappings and (U)CQ queries,* PERF *is in Π_2^p.*

PROOF. We do not need to actually compute the whole rewriting r_s of q_t, which in the worst-case is an UCQ consisting of an exponential number of CQs. Instead, we can check whether $r \sqsubseteq q_s$, for each CQ r in r_s, by a coNP procedure that uses an NP oracle for checking CQ containment, which gives us the claim. □

[4] Recall that when $e \rightsquigarrow v$ is the (unique) mapping assertion in M_G involving v, then $M_G[v] = e$.

3.4 Relative containment

We can check relative containment by exploiting again Theorem 4. Hence, given a GLAV CQ mapping M and two target (U)CQs q_1 and q_2, to check whether $q_1 \sqsubseteq_M q_2$, i.e., whether $cert[q_1, M]$ is contained is $cert[q_2, M]$, it suffices to compute the maximal rewritings r_1 of q_1 and r_2 of q_2, and check whether $r_1 \sqsubseteq r_2$. Hence, we get the following upper bound, which matches the one for the LAV case [26].

THEOREM 8. *For GLAV CQ mappings and (U)CQ queries,* RCON *is* Π_2^p*-complete.*

PROOF. The hardness already holds for the LAV case [49]. For the upper bound, again, we do not need to actually compute the whole rewritings r_1 and r_2. Instead, we can check whether $r \sqsubseteq r_2$, for each CQ r in r_1 by a coNP procedure that uses an NP oracle to guess both a CQ r' in r_2 and a homomorphism from r' to r. □

4. QUERY ANSWERING AND REWRITING FOR GRAPH DATABASES

For query answering under GLAV mappings in the case of RPQs, we can proceed as for UCQs, and make use of Theorem 2 to compute the certain answers to a target RPQ w.r.t. a GLAV RPQ mapping.

THEOREM 9. *For GLAV RPQ mappings and RPQ queries,* ANS *is* coNP*-complete in data complexity and* PSPACE*-complete in combined complexity.*

PROOF. The lower bounds follow directly from the same lower bounds for LAV mappings. For the upper bounds, it suffices to observe that we can materialize the intermediate views resulting from splitting the GLAV mapping in PTIME in data complexity and combined complexity. The claim then follows from Theorem 2 and from the upper bounds for the LAV case (see Section 2.3). □

We now turn to query rewriting for GLAV RPQ mappings. The following example shows that in this case rewriting cannot be achieved by splitting the GLAV mapping into a GAV and a LAV mapping, and treating them separately.

EXAMPLE 2. *Consider a source schema* $\Sigma = \{a_1, a_2\}$ *and a target schema* $\Gamma = \{b_1, b_2, b_3, b_4\}$*, where all source and target symbols are binary relations. Consider the GLAV RPQ mapping* M *with mapping assertions*

$$a_1 \rightsquigarrow b_1, \qquad a_1 \rightsquigarrow b_2, \qquad a_2 \rightsquigarrow b_3 + b_4.$$

By splitting M*, we obtain the LAV mapping* M_L

$$v_1 \rightsquigarrow b_1, \qquad v_2 \rightsquigarrow b_2, \qquad v_3 \rightsquigarrow b_3 + b_4$$

and the GAV mapping

$$a_1 \rightsquigarrow v_1, \qquad a_1 \rightsquigarrow v_2, \qquad a_2 \rightsquigarrow v_3.$$

Given the target RPQ $q_t = b_1 \cdot b_3 + b_2 \cdot b_4$*, the RPQ-maximal rewriting of* q_t *w.r.t.* M_L *is empty (notice that RPQs are not closed under conjunction). On the other hand, it is easy to verify that a non-empty rewriting of* q_t *with respect to* M *is* $a_1 \cdot a_2$*.*[5]

[5]Note that this example is based on Example 4.1 in [27], where it is shown that the RPQ rewriting of q_t w.r.t. M_L is empty, but that a non-empty rewriting can be expressed as a conjunctive RPQ [23].

To address emptiness of RPQ rewritings in the case of GLAV RPQ mappings, following the approach adopted for LAV mappings (see Section 2.3), we aim at constructing an NFA for the RPQ-maximal rewriting and then checking non-emptiness of such an NFA. The straightforward extension of such an approach to the GLAV case is as follows. Let $w_s = a_1 \cdots a_n$ be a word over the source alphabet Σ and D_{w_s} a source database of the form $\{a_1(c_0, c_1), a_2(c_1, c_2), \ldots, a_n(c_{n-1}, c_n)\}$, for some constants $c_0, c_1, \ldots, c_n$. Then, w_s (considered as a source query) is *not* a rewriting of a target query q_t w.r.t. a GLAV mapping M iff w_s is not M-contained in q_t iff there is a homomorphism from the structure $(M_G(D_{w_s}), c_0, c_n)$ to $CT(q_t, M_L)$, where M_G and M_L are respectively the GAV and LAV mappings obtained by splitting M. Again we can construct an NFA $A_{q_t,M}$ that recognizes bad rewritings. This NFA guesses the homomorphism from $(M_G(D_{w_s}), c_0, c_n)$ to $CT(q_t, M_L)$. The difficulty with this approach, however, is that the homomorphism condition now is not only between adjacent letters in w_s. Rather, we need to check that the homomorphism condition holds for every edge in $M_G(D_{w_s})$. Thus, the automaton $A_{q_t,M}$, which tests for the negation of the homomorphism condition, is already of doubly exponential size, so its complement $A^c_{q_t,M}$ is of triply exponential size. As shown next, we can retain the same upper bound as for the LAV case by following a direct approach in the construction of the automaton for the rewriting.

We first describe a direct approach for the case of LAV mappings and then show how to modify it for GLAV mappings. Let the target RPQ q_t be given in terms of an NFA A_{q_t} over the target alphabet Γ, let $\Sigma = \{a_1, \ldots, a_m\}$ be the source alphabet, and let $M_L = \bigcup_{1 \le i \le m} \{a_i \rightsquigarrow f_i\}$ be a LAV RPQ mapping. We proceed as follows:

1. We first determinize A_{q_t} to get an equivalent DFA $A_d = (\Gamma, S, s_0, \varrho, F)$ over Γ.

2. We construct a universal automaton A_u over the state space S of A_d and over the source alphabet Σ as follows: $A_u = (\Sigma, S, s_0, \delta, F)$, where the transition function δ is defined as follows, where we assume that the RPQ f_i is represented as an NFA. For each $s \in S$ and $a_i \in \Sigma$:

$$\delta(s, a_i) = \{s' \mid \text{there exists a word } w \in \mathcal{L}(f_i) \text{ such that } \varrho(s, w) = s'\}.$$

   Intuitively, A_u simulates A_d over *all* target databases that are consistent with the input source database.

3. We now construct the automaton R_{q_t,M_L} as the DFA equivalent to the universal automaton A_u: $R_{q_t,M_L} = (\Sigma, 2^S, \{s_0\}, \alpha, T_F)$, where $T_F = \{P \in 2^S \mid P \subseteq F\}$, and $\alpha(T, a) = \delta(T, a)$ (again, we consider δ extended to sets of states in the usual way). Intuitively, when R_{q_t,M_L} runs on a word w of the source alphabet Σ, it simulates all possible runs over possible unfoldings of w w.r.t. M_L.

The construction of A_d is exponential, as is the construction of A_u, which is why the construction of R_{q_t,M_L} is doubly exponential.

Note that a run of R_{q_t,M_L} over $w = a_1 \cdots a_n$ is a sequence $T_0 \cdots T_n$ of subsets of S such that:

(a) $\delta(T_{i-1}, a_i) = T_i$, for $i \in \{1, \ldots, n\}$, and

(b) $T_n \subseteq F$.

In fact, because of Requirement (b), we could relax Requirement (a) and rephrase it as:

(a') $\delta(T_{i-1}, a_i) \subseteq T_i$, for $i \in \{1, \ldots, n\}$.

Now we can see how this construction can be modified for a GLAV mapping M. Again, we assume that the target RPQ q_t is given in terms of an NFA A_{q_t} over the target alphabet Γ, and let $M = \bigcup_{1 \leq i \leq m} \{e_i \rightsquigarrow f_i\}$ be a GLAV RPQ mapping. We proceed as follows:

1. We first determinize A_{q_t} to get an equivalent DFA $A_d = (\Gamma, S, s_0, \varrho, F)$ over Γ.
2. For each target query f_i in the right-hand side of a mapping assertion of M, we construct an NFA A_i over the target alphabet Γ.
3. We construct an NFA A_n over the state space S of A_d and over the intermediate alphabet $\mathcal{V} = \{v_1, \ldots, v_m\}$ of view symbols, as follows: $A_n = (\mathcal{V}, S, s_0, \delta, F)$, where δ is defined as follows:
$$\delta(s, v_i) = \{s' \mid \text{there exists a word } u \in \mathcal{L}(A_i) \text{ such that } \varrho(s, u) = s'\}.$$
4. We now need to characterize rewritings of q_t w.r.t. M.

Proposition 10. *A word $w = a_1 \cdots a_n$ over the source alphabet Σ is a rewriting of q_t w.r.t. M if there exists a sequence $T_0 \cdots T_n$ of subsets of S such that the following hold:*
(a) If a subword $w[i, j] = a_i \cdots a_{j-1}$, for $1 \leq i < j \leq n + 1$ of w is in $\mathcal{L}(e_k)$, then $\delta(T_{i-1}, v_k) \subseteq T_j$.
(b) $T_n \subseteq F$.

We call $T_0 \cdots T_n$ a *witnessing sequence* for w.

5. The idea behind the construction of the rewriting $R_{q_t,M}$ of q_t w.r.t. M is to guess a candidate witnessing sequence and check that is satisfies the conditions of Proposition 10. It is not obvious, however, how to check Condition (a), where we quantified over all subwords $w[i, j]$ of w. So we take an indirect approach.

Lemma 11. *Let $w = a_1 \cdots a_n$ be a word over the source alphabet Σ and let $T_0 \cdots T_n$ be a sequence of subsets of S such that $T_n \subseteq F$. Then $T_0 \cdots T_n$ is not a witnessing sequence for w if there exists a subword $w[i, j]$ of w that is in $\mathcal{L}(e_k)$ and $\delta(T_{i-1}, v_k) \not\subseteq T_j$.*

Thus, it is easy to construct an automaton A_{bad} that runs on an interleaving of a word $w = a_1 \cdots a_n$ and a candidate witnessing sequence $T_0 \cdots T_n$ and checks that $T_0 \cdots T_n$ is *not* a witnessing sequence: either Condition (b) of Proposition 10 does not hold, or we can guess a subword $w[i, j]$ and check that the condition of Lemma 11 holds. Note that the state set S of A_d is exponential in q_t, so the T_i's are of exponential size. Still, A_{bad} does not need to remember any set T_i, only elements of S, so the size of the state space of A_{bad} is exponential. By complementing A_{bad} we get an automaton A_{good}, of doubly exponential size[6], that checks that $T_0 \cdots T_n$ is a witnessing sequence for w.

[6]Note that the alphabet of A_{bad} is already of double exponential size, however it does not contribute further to the exponential blowup in the complexity of determinization.

6. Finally, we construct an NFA $R_{q_t,M}$ that, given a word w over the source alphabet Σ, guesses a candidate witnessing sequence for w and simulates A_{good} to check that the candidate witnessing sequence is indeed a witnessing sequence.

Note that $R_{q_t,M}$ is doubly exponential. In contrast to the automaton we get in the case of LAV RPQ mapping, which is a deterministic automaton, the automaton here is nondeterministic. It is this nondeterminism that allows us to overcome the potentially triply-exponential blow-up.

Theorem 12. *For GLAV RPQ mappings and RPQ queries,* Rew *is* ExpSpace*-complete.*

Proof. The claim follows from the above construction, by observing that we need not construct $R_{q_t,M}$ explicitly. Instead, we can check for non-emptiness while constructing $R_{q_t,M}$ on the fly, since all checks that it needs to make are local conditions. □

We observe that, by construction, $R_{q_t,M}$ is the RPQ-maximal rewriting. In fact, such a rewriting is the maximal one among all path-based rewritings, and it turns out that such a rewriting is actually regular, as in the LAV case.

5. PERFECTNESS FOR GRAPH DATABASES

Now, let us consider a GLAV RPQ mapping M, split into a GAV mapping M_G and a LAV mapping M_L. Given a target query q_t, by Theorem 2, we know that $cert[q_t, M](D_s) = cert[q_t, M_L](M_G(D_s))$, for every source database D_s. Let q_s be a source query. We want to test whether $cert[q_t, M] \sqsubseteq q_s$, that is, for every source database D_s and for every pair (c, d) of constants in D_s, if $(c, d) \in cert[q_t, M](D_s)$, then $(c, d) \in q_s(D_s)$. Equivalently, for all D_s, if $(c, d) \in cert[q_t, M_L](M_G(D_s))$, then $(c, d) \in q_s(D_s)$. Equivalently, if $(c, d) \notin q_s(D_s)$ then $(c, d) \notin cert[q_t, M_L](M_G(D_s))$. Equivalently, if there is a homomorphism from (D_s, c, d) to $CT(q_s, I)$, then there is a homomorphism from $(M_G(D_s), c, d)$ to $CT(q_t, M_L)$, where I is the identity mapping on Σ. As in the LAV case, we can show that it suffices to consider only those constraint instances (D_s, c, d) that are of the form $CT'(q_s, I)$, where $CT'(q_s, I)$ is identical to $CT(q_s, I)$, except that the unary relations u_i and u_f are restricted to a singleton. Thus, we obtain that checking perfectness amounts to checking that there is a homomorphism from each $M_G(CT'(q_s, I))$ to $CT(q_t, M_L)$, where we assume that M_G acts as the identity mapping on the unary relations u_i and u_f.

Theorem 13. *For GLAV RPQ mappings and RPQ queries,* Perf *is* NExpTime*-complete.*

Proof. The lower bound holds already for the case of LAV RPQ mappings [26]. For the upper bound, from the construction above, we have that the constraint template $CT(q_s, I)$ is of exponential size in the size of q_s, and there are exponentially many restrictions of $CT(q_s, I)$ in which the unary source and target relations are singletons. $CT(q_t, M_L)$ is also of exponential size. Given that checking for the existence of a homomorphism between two structures is NP-complete in the size of the structures, we get the NExpTime upper bound. □

Note that the algorithm in Theorem 13 has to consider $2^{|q_s|}$ many restrictions of $CT(q_s, I)$, and, for each such restriction it has to consider $2^{|q_t| \cdot 2^{|q_s|}}$ mappings into $CT(q_t, M_L)$. Thus, its deterministic complexity is exponential in $|q_t|$ and doubly exponential in $|q_s|$.

6. RELATIVE CONTAINMENT FOR GRAPH DATABASES

Given a GLAV RPQ mapping M and two target RPQs q_1 and q_2, we want to check whether $q_1 \sqsubseteq_M q_2$, i.e., whether $cert[q_1, M]$ is contained is $cert[q_2, M]$. That is, for every source database D_s, if $(c, d) \in cert[q_1, M](Ds)$, then $(c, d) \in cert[q_2, M](Ds)$. Equivalently, if $(c, d) \notin cert[q_2, M](Ds)$ then $(c, d) \notin cert[q_1, M](Ds)$. Equivelently, if there is a homomorphism from $(M_G(D_s), c, d)$ to $CT(q_2, M_L)$, then there is a homomorphism from $(M_G(D_s), c, d)$ to $CT(q_1, M_L)$, where M_G and M_L are respectively the GAV and LAV mappings obtained from M.

Thus, containment fails if there is a source database D_s such that there is a homomorphism h from $(M_G(D_s), c, d)$ to $CT(q_2, M_L)$, but there is *no* homomorphism from $(M_G(D_s), c, d)$ to $CT(q_1, M_L)$.

Let $\mathcal{V}$ be the schema of intermediate view symbols introduced by splitting M into M_G and M_L. Ideally, we'd like to be able to guess a database D_v over the intermediate view symbols $\mathcal{V}$, check that $D_v = M_G(D_s)$ for some source database D_s, and check that there is a homomorphism h from (D_v, c, d) to $CT(q_2, M_L)$, but there is no homomorphism from (D_v, c, d) to $CT(q_1, M_L)$.

There are two challenges:

1. We need to be able to solve the inverse problem: given D_v, is there some D_s such that $D_v = M_G(D_s)$.

2. We need to be able to bound the size of D_v.

We address first the inverse problem, and we do it so that we also get a bound on D_v. Let the elements of D_v be $x_1, \ldots, x_n$. For each edge $v_j(x_{j_1}, x_{j_2})$, D_s must contain a path in $\mathcal{L}(e_j)$ from x_{j_1} to x_{j_2}, where e_j is the source query in the mapping assertion $e_j \rightsquigarrow f_j$ corresponding to v_j. Some of the elements on these paths come from $x_1, \ldots, x_n$, but others do not. Let D'_s be the source database obtained by extracting from D_s all these paths and "linearizing" them.

This process duplicates elements, so an element x_i may have several copies in D'_s. Thus, it is fine if $M_G(D'_s)$ creates copies of edges in D_v, but it should not create new edges. We can represent D'_s as a set of k words, where k is the number of edges in D_v, and the words are over the alphabet $\{x_1, \ldots, x_n\} \cup \{x\} \cup \Sigma$, where x denotes an element of D'_s that is not in D_v. In this representation of paths, letters from $\{x_1, \ldots, x_n\} \cup \{x\}$ represent nodes, and letters from Σ represent edges. Note that two occurrences of some x_{i_j} represent the same node x_{i_j}, but we assume that all occurrences of x represent distinct nodes.

Suppose now that we are given D'_s represented as a set of k words. We want to check that $\mathcal{M}_G(D'_s) = D_v$. Given such a candidate source D'_s, for the j-th edge $v_j(x_{j_1}, x_{j_2})$ in D_v, we need to check, using an NFA A_j that the j-th word in D'_s, considering edges only, is a word in $\mathcal{L}(e_j)$ from x_{j_1} to x_{j_2}.

More difficult is to check that $M_G(D'_s)$ contains no spurious edges. There are two types of spurious edges: (1) edges that arise from a path corresponding to $\mathcal{L}(e_j)$ between y_1 and y_2, where either y_1 or y_2 is an x node; (2) edges corresponding to a path in $\mathcal{L}(e_j)$ between y_1 and y_2, which are not x nodes, where the edge $v_j(y_1, y_2)$ is not in D_v.

To check for spurious edges that include an x node, we label all nodes on the words of D'_s by sets of states of the NFAs corresponding to the e_j's, such that:

(a) all x nodes are labeled by the start states,

(b) if a word contains $y \cdot b \cdot z$, y's label contains a state s, and one of the NFAs has a transition from s to t upon reading b, then z's label contains t, and

(c) if the label of one node x_i contains a state s, then all nodes x_i contain s in their label.

No node can be labeled by an accepting state, since this would imply that $M_G(D'_s)$ contained an edge that includes an x node. We also need a similar labeling where non-x nodes are labeled by the start states, and no x node is labeled by an accepting state.

Instead of checking for spurious edges between non-x nodes, we check that there is a homomorphism from $M_G(D'_s)$ to $CT(q_2, M_L)$. That is we need a labeling of all nodes in D_v by elements of $CT(q_2, M_L)$ such that, if there is a path corresponding to $\mathcal{L}(e_j)$ between non-x nodes y_1 and y_2 in D'_s, labeled by elements p_1 and p_2 of $CT(q_2, M_L)$, then there is a v_j edge between p_1 and p_2.

To check this we need, for each element p_1 of $CT(q_2, M_L)$, a labeling of all nodes in the words of D'_s by sets of states of the NFAs corresponding to the e_j's such that:

(a) all nodes labeled by p_1 are labeled by the start states,

(b) if a word contained $y \cdot b \cdot z$, y's label contains a state s, and one of the NFA's has a transition from s to t upon reading b, then z's label contains t, and

(c) if the label of one node x_i contains a state s, then all nodes x_i contain s in their label.

Now a node labeled by an element p_2 of $CT(q_2, M_L)$ can contain an accepting state of the NFA of e_i in its label only if there is an edge $v_i(p_1, p_2)$ in D_v.

Altogether, if q_2 is of size m, then $CT(q_2, M_L)$ has at most 2^m nodes, and we need $2^m + 2$ labelings to ensure that we avoid spurious edges and violation of homomorphism. Since all x_i nodes are labeled identically, we can guess their labelings, and then guess the words of D'_s one letter at a time checking for existence of consistent labelings on the fly. This can be done in space $O(n \cdot (2^m + 2))$, where n is the size of the expressions e_i's.

It is now quite possible that $M_G(D'_s)$ properly contains D_v, but if there is no homomorphism from (D_v, c, d) to $CT(q_1, M_L)$, then also there is no homomorphism from $(M_G(D'_s), c, d)$ to $CT(q_1, M_L)$, and our labelings guarantee that there is a homomorphism from $(M_G(D'_s), c, d)$ to $CT(q_2, M_L)$. Hence, D_v is not necessarily the counter-example we are looking for, but $M_G(D'_s)$ is guaranteed to be so. So, we do not really guess a counter-example, but rather a witness of its existence.

Note that the number of labels is $2^{n \cdot (2^m + 2)}$. Suppose that two nodes x_{j_1} and x_{j_2} of D_v have the same labels. Then if we equate them, there is no need to change the labeling, so we will still have that there is a homomorphism

from $(M_G(D'_s), c, d)$ to $CT(q_2, M_L)$ but not to $CT(q_1, M_L)$. Thus, we can assume, without loss of generality, that the size of D_v is bounded by $2^{n\cdot(2^m+2)}$.

Thus, to check that $cert[q_1, M]$ is not contained in $cert[q_2, M]$, we guess an intermediate database D_v of size bounded by $2^{n\cdot(2^m+2)}$, check that there is no homomorphism from D_v to $cert[q_1, M]$, guess the labelings of the elements in D_v, and then for each edge $v_j(x_{j_1}, x_{j_2})$ in D_v, guess on the fly a word in $\mathcal{L}(e_j)$ and check that the labeling can be extended to the whole word consistently. We obtain the following result.

THEOREM 14. *For GLAV RPQ mappings and RPQ queries,* RCON *is decidable.*

7. CONCLUSIONS

We have carried out a thorough study on query processing under schema mappings, for the case of both relational and graph databases. We have shown that the common intuition that it is easy to extend the results from LAV to GLAV is actually false in the case of graph databases. Indeed, GLAV query processing for graph data requires new insights and non-trivial techniques. In the future, we plan to continue our investigation, by considering other relevant mapping-based tasks that were left out in this paper, such as checking a rewriting for exactness (i.e., equivalence to the query, modulo the mappings), or checking the mappings for losslessness w.r.t. a target query. Interestingly, the first problem to address about these tasks is to define a convincing semantics, since the semantics adopted for the case of LAV mappings does not extend easily to the case of GLAV.

Acknowledgements

Work partially supported by the EU under the projects ACSI (Artifact-Centric Service Interoperation), grant n. FP7-257593 and Optique (Scalable End-user Access to Big Data), grant n. FP7-318338.

8. REFERENCES

[1] S. Abiteboul. Querying semi-structured data. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, pages 1–18, 1997.

[2] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: from Relations to Semistructured Data and XML*. Morgan Kaufmann, 2000.

[3] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *Proc. of the 17th ACM Symp. on Principles of Database Systems (PODS'98)*, pages 254–265, 1998.

[4] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., 1995.

[5] S. Abiteboul, I. Manolescu, P. Rigaux, M.-C. Rousset, and P. Senellart. *Web Data Management*. Cambridge University Press, 2011.

[6] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener. The Lorel query language for semistructured data. *Int. J. on Digital Libraries*, 1(1):68–88, 1997.

[7] F. N. Afrati, M. Chandrachud, R. Chirkova, and P. Mitra. Approximate rewriting of queries using views. In *Proc. of the 13th East European Conference on Advances in Databases and Information Systems (ADBIS 2009)*, volume 5739 of *Lecture Notes in Computer Science*, pages 164–178. Springer, 2009.

[8] S. Amano, C. David, L. Libkin, and F. Murlak. On the tradeoff between mapping and querying power in XML data exchange. In *Proc. of the 13th Int. Conf. on Database Theory (ICDT 2010)*, pages 155–164, 2010.

[9] M. Arenas, P. Barcelo, R. Fagin, and L. Libkin. Locally consistent transformations and query answering in data exchange. In *Proc. of the 23rd ACM Symp. on Principles of Database Systems (PODS 2004)*, pages 229–240, 2004.

[10] M. Arenas, P. Barcelo, L. Libkin, and F. Murlak. *Relational and XML Data Exchange*, volume 8 of *Synthesis Lectures on Data Management*. Morgan & Claypool, 2010.

[11] M. Arenas, R. Fagin, and A. Nash. Composition with target constraints. In *Proc. of the 13th Int. Conf. on Database Theory (ICDT 2010)*, pages 129–142, 2010.

[12] M. Arenas, J. Pérez, J. L. Reutter, and C. Riveros. Foundations of schema mapping management. In *Proc. of the 29th ACM Symp. on Principles of Database Systems (PODS 2010)*, pages 227–238, 2010.

[13] M. Arenas, J. Pérez, and C. Riveros. The recovery of a schema mapping: Bringing exchanged data back. *ACM Trans. on Database Systems*, 34(4), 2009.

[14] P. C. Arocena, A. Fuxman, and R. J. Miller. Composing local-as-view mappings: Closure and applications. In *Proc. of the 13th Int. Conf. on Database Theory (ICDT 2010)*, pages 209–218, 2010.

[15] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. Extending decidable cases for rules with existential variables. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009)*, pages 677–682, 2009.

[16] P. A. Bernstein and H. Ho. Model management and schema mappings: Theory and practices. In *Proc. of the 33rd Int. Conf. on Very Large Data Bases (VLDB 2007)*, pages 1439–1440, 2007.

[17] P. Buneman. Semistructured data. In *Proc. of the 16th ACM Symp. on Principles of Database Systems (PODS'97)*, pages 117–121, 1997.

[18] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization technique for unstructured data. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 505–516, 1996.

[19] A. Calì. Query answering by rewriting in GLAV data integration systems under constraints. In *Proc. of the 2nd Int. Workshop on Semantic Web and Databases (SWDB 2004)*, volume 3372 of *Lecture Notes in Computer Science*, pages 167–184. Springer, 2004.

[20] A. Calì, G. Gottlob, and A. Pieris. Advanced processing for ontological queries. *Proc. of the VLDB Endowment*, 3(1):554–565, 2010.

[21] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Answering regular path queries using views. In *Proc. of the 16th IEEE Int. Conf. on Data Engineering (ICDE 2000)*, pages 389–398, 2000.

[22] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Containment of conjunctive regular path queries with inverse. In *Proc. of the 7th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 176–185, 2000.

[23] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query processing and constraint satisfaction. In *Proc. of the 15th IEEE Symp. on Logic in Computer Science (LICS 2000)*, pages 361–371, 2000.

[24] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Rewriting of regular expressions and regular path queries. *J. of Computer and System Sciences*, 64(3):443–465, 2002.

[25] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Reasoning on regular path queries. *SIGMOD Record*, 32(4):83–92, 2003.

[26] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query containment. In *Proc. of the 22nd ACM Symp. on Principles of Database Systems (PODS 2003)*, pages 56–67, 2003.

[27] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query processing: On the relationship between rewriting, answering and losslessness. *Theoretical Computer Science*, 371(3):169–182, 2007.

[28] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Simplifying schema mappings. In *Proc. of the 14th Int. Conf. on Database Theory (ICDT 2011)*, pages 114–125, 2011.

[29] S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer, Berlin (Germany), 1990.

[30] O. M. Duschka, M. R. Genesereth, and A. Y. Levy. Recursive query plans for data integration. *J. of Logic Programming*, 43(1):49–73, 2000.

[31] J. Euzenat and P. Schwaiko. *Ontology Matching*. Springer, 2007.

[32] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

[33] R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: Getting to the core. *ACM Trans. on Database Systems*, 30(1):174–210, 2005.

[34] R. Fagin, P. G. Kolaitis, L. Popa, and W.-C. Tan. Composing schema mappings: Second-order dependencies to the rescue. *ACM Trans. on Database Systems*, 30(4):994–1055, 2005.

[35] R. Fagin, P. G. Kolaitis, L. Popa, and W.-C. Tan. Quasi-inverses of schema mappings. *ACM Trans. on Database Systems*, 33(2):1–52, 2008.

[36] R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Reverse data exchange: Coping with nulls. In *Proc. of the 28th ACM Symp. on Principles of Database Systems (PODS 2009)*, pages 23–32, 2009.

[37] T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction. *SIAM J. on Computing*, 28:57–104, 1999.

[38] M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI'99)*, pages 67–73. AAAI Press, 1999.

[39] G. Grahne and A. O. Mendelzon. Tableau techniques for querying information sources through global schemas. In *Proc. of the 7th Int. Conf. on Database Theory (ICDT'99)*, volume 1540 of *Lecture Notes in Computer Science*, pages 332–347. Springer, 1999.

[40] G. Grahne and A. Thomo. Query containment and rewriting using views for regular path queries under constraints. In *Proc. of the 22nd ACM Symp. on Principles of Database Systems (PODS 2003)*, pages 111–122, 2003.

[41] A. Y. Halevy, A. Rajaraman, and J. Ordille. Data integration: The teenage years. In *Proc. of the 32nd Int. Conf. on Very Large Data Bases (VLDB 2006)*, pages 9–16, 2006.

[42] C. 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.

[43] P. G. Kolaitis. Schema mappings, data exchange, and metadata management. In *Proc. of the 24th ACM Symp. on Principles of Database Systems (PODS 2005)*, pages 61–75, 2005.

[44] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of the 21st ACM Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.

[45] A. Y. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *Proc. of the 14th ACM Symp. on Principles of Database Systems (PODS'95)*, pages 95–104, 1995.

[46] A. Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *J. of Intelligent Information Systems*, 5:121–143, 1995.

[47] L. Libkin and C. Sirangelo. Data exchange and schema mappings in open and closed worlds. In *Proc. of the 27th ACM Symp. on Principles of Database Systems (PODS 2008)*, pages 139–148, 2008.

[48] J. Madhavan and A. Y. Halevy. Composing mappings among data sources. In *Proc. of the 29th Int. Conf. on Very Large Data Bases (VLDB 2003)*, pages 572–583, 2003.

[49] T. D. Millstein, A. Y. Levy, and M. Friedman. Query containment for data integration systems. In *Proc. of the 19th ACM Symp. on Principles of Database Systems (PODS 2000)*, pages 67–75, 2000.

[50] T. D. Millstein, A. Y. Levy, and M. Friedman. Query containment for data integration systems. *J. of Computer and System Sciences*, 66(1):20–39, 2003.

[51] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.

[52] J. D. Ullman. Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210, 2000.

[53] M. Y. Vardi. The complexity of relational query languages. In *Proc. of the 14th ACM Symp. on Theory of Computing (STOC'82)*, pages 137–146, 1982.