Simplifying Schema Mappings

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

Giuseppe De Giacomo
Maurizio Lenzerini
Dip. di Inf. e Sist.
Univ. di Roma "La Sapienza"
I-00185 Roma, Italy
lastname@dis.uniroma1.it

Moshe Y. Vardi
Dep. of Computer Science
Rice University, P.O. Box 1892
Houston, TX 77251-1892,
U.S.A.
vardi@cs.rice.edu

ABSTRACT

A schema mapping is a formal specification of the relationship holding between the databases conforming to two given schemas, called source and target, respectively. While in the general case a schema mapping is specified in terms of assertions relating two queries in some given language, various simplified forms of mappings, in particular LAV and GAV, have been considered, based on desirable properties that these forms enjoy. Recent works propose methods for transforming schema mappings to logically equivalent ones of a simplified form. In many cases, this transformation is impossible, and one might be interested in finding simplifications based on a weaker notion, namely logical implication, rather than equivalence. More precisely, given a schema mapping M, find a simplified (LAV, or GAV) schema mapping M' such that M' logically implies M. In this paper we formally introduce this problem, and study it in a variety of cases, providing techniques and complexity bounds. The various cases we consider depend on three parameters: the simplified form to achieve (LAV, or GAV), the type of schema mapping considered (sound, or exact), and the query language used in the schema mapping specification (conjunctive queries and variants over relational databases, or regular path queries and variants over graph databases). Notably, this is the first work on comparing schema mappings for graph databases.

Categories and Subject Descriptors

D.2.12 [**Software Engineering**]: Interoperability—*data mapping*; H.2.3 [**Database Management**]: Languages—*query languages*

General Terms

Theory

Keywords

Schema mappings, GAV, LAV, graph databases, conjunctive queries

ICDT 2011, March 21–23, 2011, Uppsala, Sweden.

1. INTRODUCTION

A schema mapping is a formal specification of the relationship holding between the databases conforming to two given schemas. Many papers point out the importance of schema mappings in several data management tasks, especially those requiring inter-operability between different information systems, such as data integration [38, 36], data exchange [37, 14], and model management [17].

In data integration, schema mappings are established between the source schema and the global (mediated) schema, and are used to decide how to access the source data for answering queries posed in terms of the global schema. In data exchange, schema mappings are specified in terms of a source schema and a target schema, and determine how the source data should be transferred to the target in order to populate a database conforming to the target schema. Schema mappings are also the main objects of interest in model management, whose goal is to support the creation, compilation, reuse, evolution, and execution of mappings between schemas, expressed in a wide range of model.

A schema mapping is constituted by two schemas and a set of mapping assertions between the two. We follow the data exchange terminology, and call the two schema *source* and *target*, respectively. As usual, we assume that each assertion relates a query q_s over the source to a query q_t over the target, and specifies a correspondence between the tuples computed by q_s in source databases and those computed by q_t in target databases. In the following, we consider two types of schema mappings, called *sound* and *exact*, respectively. The correspondence specified by assertions in a sound mapping is *inclusion*, whereas the correspondence specified by assertions in an exact mapping is *equality*. Semantically, a schema mapping M is characterized by the set of pairs $(\mathcal{D}_s, \mathcal{D}_t)$ of databases such that $\mathcal{D}_s$ is a source database, $\mathcal{D}_t$ is a target database, and they satisfy the correspondences sanctioned by the assertions in M.

Since the pioneering work on schema mappings in data integration [46], various restricted forms of mappings have been considered, in particular LAV and GAV. In LAV (Local-As-Views) mappings, the source queries in the assertions are constituted by one atom, and exactly one assertion appears for each relation symbol in the source schema. In other words, a LAV mapping associates to each element of the source schema one view over the target schema. Conversely, a GAV (Global-As-Views) mapping associates to each element of the target schema one view over the source schema. Extending the above terminology, the term GLAV is often used to refer to unrestricted forms of schema mappings.

Schema mappings have been widely investigated in the last years. In [26, 28, 7, 40] the emphasis is on providing foundations for data exchange systems based on schema mappings. Other works deal with answering queries posed to the target schema on the basis of both the data at the sources, and a set of source-to-target mapping assertions (see, for instance, [3, 7] and the surveys in [46, 36]). A large body of work has been devoted to studying operators on schema mappings relevant to model management, notably, composition, merge, and inverse (see, for example [41, 30, 31, 29, 11, 12, 8, 10]).

Recently, there has been a growing interest in principles and tools for comparing both schema mapping languages, and schema mappings expressed in a certain language. Comparing schema mapping languages aims at characterizing such languages in terms of both expressive power, and complexity of mapping-based computational tasks [45, 6]. In particular, [45] studies various relational schema mapping languages with the goal of characterizing them in terms of structural properties possessed by the schema mappings specified in these languages.

Methods for comparing schema mappings have been recently proposed in [27, 34, 29, 8]. In [29, 8], schema mappings are compared with respect to their ability to transfer source data and avoid redundancy in the target databases, as well as their ability to cover target data. More relevant to the present paper is the work in [27], which introduces three notions of equivalence. The first one is the usual notion based on logic: two schema mappings are logically equivalent if they are indistinguishable by the semantics, i.e., if they are satisfied by the same set of database pairs. The other two notions, called data exchange and conjunctive query equivalence, respectively, are relaxations of logical equivalence, capturing indistinguishability for different purposes. In [34], schema mapping optimization is studied, based on logical equivalence. In particular, a set of optimality criteria are proposed for an important class of relational schema mappings, and rewriting rules for transforming a schema mapping into an equivalent optimal one are presented. Notably, LAV and GAV enjoy many of the optimality criteria mentioned in the paper. It follows that the proposed rewriting rules often lead to transforming an input schema mapping into one of the two simplified forms.

The above discussion shows that the work on optimization and simplification of schema mappings has concentrated so far on equivalence preserving transformations. However, there are cases where equivalence preserving simplification is not possible, as demonstrated for LAV by the following example.

Example 1. Consider the schema mapping M constituted by the following mapping assertion

$$\{(x,w,z) \mid r_1(x,w,y) \wedge r_2(y,z)\} \rightsquigarrow \{(x,w,z) \mid t_1(x,w,v) \wedge t_1(v,w,e) \wedge t_2(e,z)\}$$

Suppose that M is interpreted under the sound semantics. Now, let $\mathcal{D}_s$ be a database such that $r_1^{\mathcal{D}_s} \neq \emptyset$, and $r_2^{\mathcal{D}_s} = \emptyset$, and let $\mathcal{D}_t$ be the empty database. Clearly, $(\mathcal{D}_s, \mathcal{D}_t) \models M$. On the other hand, for any sound LAV mapping M' on the same alphabet as M, it holds that $(\mathcal{D}_s, \mathcal{D}_t) \not\models M'$, because $r_1^{\mathcal{D}_s} \neq \emptyset$, while the query $M'(r_1)$ that M' associates to r_1 is such that $M'(r_1)^{\mathcal{D}_t} = \emptyset$, and therefore the assertion $r_1 \rightsquigarrow M'(r_1)$ in M' cannot be satisfied by $(\mathcal{D}_s, \mathcal{D}_t)$. It follows that no LAV mapping M' exists such that $M \models M'$, and therefore equivalence preserving LAV simplification of M is impossible to achieve. ■

To address such cases, we argue that simplification should be based on a weaker notion, namely logical implication, rather than equivalence.

Example 2. Refer again to the schema mapping M of Example 1, and consider the sound LAV mapping M'' constituted by the following two mapping assertions:

$$\{(x,w,y) \mid r_1(x,w,y)\} \rightsquigarrow \{(x,w,y) \mid t_1(x,w,v) \wedge t_1(v,w,y)\}$$
$$\{(x,y) \mid r_2(x,y)\} \rightsquigarrow \{(x,y) \mid t_2(x,y)\}$$

It is not difficult to see that $M'' \models M$. Therefore, if we are happy with LAV simplification based on logical implication rather than logical equivalence, M'' represents an acceptable simplification of M. ■

Mapping simplification based on logical implication is the subject of this paper. The problem can be stated as follows: given a schema mapping M, check whether a simplified (LAV, or GAV) schema mapping M' exists such that M' logically implies M (and, if it exists, find one). We formally introduce this problem, and study it in a variety of cases, depending on three parameters:

1. the simplified form to achieve (LAV, or GAV),
2. the type of schema mapping considered (sound, or exact), and
3. the data model and the query language used in the schema mapping specification.

As for the first parameter, we essentially concentrate on LAV in this paper. We discuss GAV only briefly, pointing out that GAV simplification is an open problem in several cases.

As for the type of mapping, although the sound semantics is the more popular one in data exchange [37], the importance of considering exact schema mappings is widely recognized for both data exchange [33], and data integration [35].

As for the data model and the query language used in schema mappings, we consider both the relational data model with conjunctive queries and unions thereof, and the graph database model with regular path queries and their extensions. Note that, while schema mappings have been extensively studied for relational data, and, to some extent, for XML data [9], this is the first paper on comparing schema mappings for graph databases. Graph databases [25] were introduced in the '80s, and are regaining wide attention recently [2, 32, 15], for their relevance in areas such as semi-structured data, biological data management, social networks, and the semantic web.

The results we present in this paper can be summarized as follows. We first illustrate our ideas with relational mappings, where the results follow fairly easily from the characterization of containment for conjunctive queries and unions thereof. We show that LAV simplification is NP-complete in the case of both sound and exact schema mappings based on conjunctive queries. In the case of unions of conjunctive queries, the problem is still in NP for sound mappings, while it is in Π_2^p for exact ones.

For graph database schema mappings based on regular path queries, we prove that LAV simplification is PSPACE-complete under the sound semantics, and in EXPSPACE in the case of exact schema mappings. By exploiting a language-theoretic characterization for containment of regular path queries with inverse (called two-way regular path queries) provided in [22], we also extend the results to the case where queries in schema mappings are two-way regular path queries, as well as conjunctive two-way regular path queries, and unions of such queries.

Note that a regular path query returns the set of node pairs in the graph database connected by a path conforming to the query, and therefore can be seen as the regular language constituted by all the words labeling the paths denoted by the query. Indeed, the simplification problem addressed in this paper has a language theoretic interpretation in terms of language equations. In general, solving systems of equations of the form $e = e'$, where e and e' are regular expressions over an alphabet of constants and variables is undecidable, because it is easy to express the universality problem for Context Free Grammars in this way. In [13], the authors study linear equations of the form

$$e_0 + e_1 \cdot x_1 + \cdots + e_n \cdot x_n = e'_0 + e'_1 \cdot x_1 + \cdots + e'_n \cdot x_n$$

where $e_0, e'_0, \ldots, e_n, e'_n$ are regular expressions, and they prove that solving these equations is EXPTIME-complete. In contrast, we prove the solvability of another class of problems, namely, systems of language constraints of the form $e_1 \sqsubseteq e_2$ and $e_1 = e_2$ where e_2 has no variables. The key idea of our approach is that we can prove that solutions of the above equations are closed under congruence, which enables us to represent languages as graphs over the finite-state automaton for e_2. Taking into account the language-theoretic view, our work has also connections with [4, 42], which also study language constraints of the forms $e_1 \sqsubseteq e_2$, and $e_1 = e_2$. However, in these works, e_1 is restricted to be a single word on both the source and the target alphabets.

The paper is organized as follows. In Section 2, we recall some preliminary notions. In Section 3, we formally define the problem of schema mapping simplification based on logical implication. In Section 4, we study the problem in the case where queries and views are conjunctive queries, and unions thereof. In Section 5 we illustrate the techniques for the case of RPQs over graph databases, and in Section 6 we extend them to two-way RPQs, and to (unions of) conjunctive two-way RPQs. Section 7 briefly discusses GAV simplification, and Section 8 concludes the paper.

2. PRELIMINARIES

In this work we deal with two data models, the standard relational model [5], and the graph database model [21].

Given a (relational) alphabet Σ, a database $\mathcal{D}$ over Σ is a finite structure over Σ. For a query q over Σ, we denote with $q^{\mathcal{D}}$ the set of tuples resulting from evaluating q in $\mathcal{D}$. A query q over Σ is *empty* if for each database $\mathcal{D}$ over Σ we have $q^{\mathcal{D}} = \emptyset$. 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 $q_1^{\mathcal{D}} \subseteq q_2^{\mathcal{D}}$ for every database $\mathcal{D}$ over Σ. The queries q_1 and q_2 are *equivalent*, denoted $q_1 \equiv q_2$, if both $q_1 \sqsubseteq q_2$ and $q_2 \sqsubseteq q_1$.

We assume familiarity with (unions of) conjunctive queries, (U)CQs, over a relational database. In particular, we consider such queries as interpreted under the active domain semantics [5]. The relational alphabets we are interested in, always include two unary predicates true and false, returning respectively the active domain and the empty set. Hence, for every n, we can express what we call the *universal query* and the *empty query* of arity n, denoted true/n, respectively false/n, as the CQ that consist of one atom $\mathsf{true}(x)$, respectively $\mathsf{false}(x)$, for every distinguished variable x. We may omit n when it is clear from the context. We recall that containment between (U)CQs can be characterized in terms of homomorphisms (also called *containment mappings*) [24]: For two CQs q_1 and q_2, we have that $q_1 \sqsubseteq q_2$ iff there is a homomorphisms from q_2 to q_1, i.e., a mapping h from the variables and constants of q_2 to those of q_1 that is the identity on distinguished variables and constants and such that, if $r(x_1, \ldots, x_k)$ is an atom of q_2 with $r \neq \mathsf{true}$, then $r(h(x_1), \ldots, h(x_k))$ is an atom of q_1. For two UCQ q_1 and q_2, we have that $q_1 \sqsubseteq q_2$ iff for each CQ q_1^i in q_1 there is a CQ q_2^j in q_2 such that $q_1^i \sqsubseteq q_2^j$ [44].

We recall the basic notions regarding graph databases and regular path queries. A *graph database* is a finite graph whose nodes represent objects and whose edges are labeled by elements from an alphabet of binary relational symbols [25, 18, 1, 22]. An edge (o_1, r, o_2) from object o_1 to object o_2 labeled by r represents the fact that relation r holds between o_1 and o_2. A *regular-path query* (RPQ) over an alphabet Σ of binary relation symbols is expressed as a regular expression or a *nondeterministic finite state automaton (1NFA)* over Σ. When evaluated on a graph database $\mathcal{D}$ over Σ, an RPQ q computes the set $q^{\mathcal{D}}$ of pairs of objects connected in $\mathcal{D}$ by a path in the regular language $\mathcal{L}(q)$ defined by q.

We consider also *two-way regular-path queries* (2RPQs) [20, 22], which extend RPQs with the *inverse* operator. Formally, let $\Sigma^{\pm} = \Sigma \cup \{r^- \mid r \in \Sigma\}$ be the alphabet including a new symbol r^- for each r in Σ. Intuitively, r^- denotes the inverse of the binary relation r. If $p \in \Sigma^{\pm}$, then we use p^- to mean the *inverse* of p, i.e., if p is r, then p^- is r^-, and if p is r^-, then p^- is r. 2RPQs are expressed by means of a 1NFA over $\Sigma^{\pm}$. When evaluated on a database $\mathcal{D}$ over Σ, a 2RPQ q computes the set $q^{\mathcal{D}}$ of pairs of objects connected in $\mathcal{D}$ by a semipath that conforms to the regular language $\mathcal{L}(q)$. A *semipath* in $\mathcal{D}$ from x to y (labeled with $p_1 \cdots p_n$) is a sequence of the form $(y_0, p_1, y_1, \ldots, y_{n-1}, p_n, y_n)$, where $n \geq 0$, $y_0 = x$, $y_n = y$, and for each y_{i-1}, p_i, y_i, we have that $p_i \in \Sigma^{\pm}$, and, if $p_i = r$ then $(y_{i-1}, y_i) \in r^{\mathcal{D}}$, and if $p_i = r^-$ then $(y_i, y_{i-1}) \in r^{\mathcal{D}}$. We say that a semipath $(y_0, p_1, \ldots, p_n, y_n)$ *conforms to* q if $p_1 \cdots p_n \in \mathcal{L}(q)$.

Finally, we consider conjunctions of 2RPQs and their unions, abbreviated (U)C2RPQs [19], which are (unions of) conjunctive queries constituted only by binary atoms whose predicate is a 2RPQ. Specifically, a C2RPQ q of arity n is written in the form

$$\{\ (x_1, \ldots, x_n) \ \mid\ q_1(y_1, y_2) \wedge \cdots \wedge q_m(y_{2m-1}, y_{2m})\ \}$$

where $x_1, \ldots, x_n, y_1, \ldots, y_{2m}$ range over a set $\{z_1, \ldots, z_k\}$ of variables, $\{x_1, \ldots, x_n\} \subseteq \{y_1, \ldots, y_{2m}\}$, and each q_j is a 2RPQ. When evaluated over a database $\mathcal{D}$ over Σ, the C2RPQ q computes the set of tuples $(o_1, \ldots, o_n)$ of objects such that there is a total mapping φ from $\{z_1, \ldots, z_k\}$ to the objects in $\mathcal{D}$ with $\varphi(x_i) = o_i$, for $i \in \{1, \ldots, n\}$, and $(\varphi(y_{2j-1}), \varphi(y_{2j})) \in q_j^{\mathcal{D}}$, for $j \in \{1, \ldots, m\}$.

We conclude by observing that (U)CQs, RPQs, 2RPQs, and (U)C2RPQs are *monotone*, where a query q is monotone if, whenever $\mathcal{D}_1 \subseteq \mathcal{D}_2$ (i.e., $r^{\mathcal{D}_1} \subseteq r^{\mathcal{D}_2}$ for each relation r) we have that $q^{\mathcal{D}_1} \subseteq q^{\mathcal{D}_2}$.

3. SCHEMA MAPPING SIMPLIFICATION

We refer to a scenario with one source schema, one target schema, and a schema mapping between the two. To model the source and the target schemas we refer to two finite alphabets, the *source alphabet* Σ_s and the *target alphabet* Σ_t, and to specify the mapping, we use a correspondence between queries expressed in a given query language.

DEFINITION 1. *Given a query language $\mathcal{Q}$, a $\mathcal{Q}$-*based (schema) mapping assertion from Σ_s to Σ_t *is a statement of the form $q_s \rightsquigarrow q_t$, where q_s and q_t are two queries in $\mathcal{Q}$ with the same arity, respectively over Σ_s and over Σ_t. A $\mathcal{Q}$-*based (schema) mapping from Σ_s to Σ_t *is a set of mapping assertions from Σ_s to Σ_t.*

In the following, we specify explicitly $\mathcal{Q}$, Σ_s, and Σ_t only when they are required or not clear from the context.

We consider two types of schema mappings, called *sound* and *exact*. Intuitively, in a sound mapping, the correspondence between the tuples computed by q_s and those computed by q_t is set containment, while in an exact mapping the correspondence is set equality. Formally, given a source database $\mathcal{D}_s$ and a target database $\mathcal{D}_t$, we say that a sound mapping M is *satisfied* by $(\mathcal{D}_s, \mathcal{D}_t)$, denoted $(\mathcal{D}_s, \mathcal{D}_t) \models M$, if for each mapping assertion $q_s \rightsquigarrow q_t$ in M, we have that $q_s^{\mathcal{D}_s} \subseteq q_t^{\mathcal{D}_t}$. Similarly, an exact mapping M is *satisfied* by $(\mathcal{D}_s, \mathcal{D}_t)$ if for each $q_s \rightsquigarrow q_t$ in M, we have that $q_s^{\mathcal{D}_s} = q_t^{\mathcal{D}_t}$.

A fundamental notion in our setting is that of logical implication between mappings.

DEFINITION 2. *A mapping M_1* logically implies *a mapping M_2, denoted $M_1 \models M_2$, if for every pair $(\mathcal{D}_s, \mathcal{D}_t)$ such that $(\mathcal{D}_s, \mathcal{D}_t) \models M_1$, we also have that $(\mathcal{D}_s, \mathcal{D}_t) \models M_2$.*

We consider two simplified forms of mappings called LAV (local-as-view) and GAV (global-as-view), respectively. A LAV *assertion* is an assertion $q_s \rightsquigarrow q_t$, where q_s is constituted simply by an atom whose predicate symbol belongs to Σ_s, while q_t is an arbitrary query[1]. Conversely, in a GAV *assertion*, q_t is constituted simply by an atom (whose predicate symbol belongs to Σ_t), while q_s is an arbitrary query. A LAV *mapping* is a set of LAV assertions with one assertion for each symbol in Σ_s. If M_L is a LAV mapping and $a \in \Sigma_s$, we denote with $M_L(a)$ the target query to which a is mapped by M_L. Conversely, a GAV *mapping* is a set of GAV assertions with one assertion for each symbol in Σ_t. Analogously to the case of LAV mappings, $M_G(a)$ denotes the source query to which the symbol $a \in \Sigma_t$ is mapped by the GAV mapping M_G. Note that some of the queries in a LAV (resp., GAV) mapping may be the empty query.

The problem we consider aims at checking whether a simplified mapping exists that logically implies a given mapping M. We would like to rule out simplified mappings that trivially imply M by making the query on the left-hand (resp., right-hand) side of some mapping assertion of M evaluate to $\emptyset$ (resp., the active domain).

[1]For RPQs and 2RPQs, where query variables are not represented explicitly, we consider an atom to be simply a binary predicate symbol.

DEFINITION 3. *A* LAV *(resp.,* GAV*) mapping M' is said to* trivially imply *a mapping M, denoted $M' \models_{triv} M$, if there is a mapping assertion $q_s \rightsquigarrow q_t \in M$ such that for each pair $(\mathcal{D}_s, \mathcal{D}_t)$ with $(\mathcal{D}_s, \mathcal{D}_t) \models M'$, we have that $q_s^{\mathcal{D}_s} = \mathsf{false}^{\mathcal{D}_s}$ (resp., $q_t^{\mathcal{D}_t} = \mathsf{true}^{\mathcal{D}_t}$).*

DEFINITION 4. *Let t be one of* SOUND *or* EXACT*, f one of* LAV *or* GAV*, and $\mathcal{Q}_1$ and $\mathcal{Q}_2$ two query languages.* Mapping simplification*, denoted*

$$\text{MSIMP}[t, f, \mathcal{Q}_1, \mathcal{Q}_2],$$

is the following decision problem: given a $\mathcal{Q}_1$-based schema mapping M of type t, check whether there exists a $\mathcal{Q}_2$-based schema mapping M' of type t and form f such that $M' \models M$, and $M' \not\models_{triv} M$.

If a simplified mapping exists, we are also interested in actually computing one. Therefore, we consider the corresponding synthesis problem.

DEFINITION 5. *Let t be one of* SOUND *or* EXACT*, f one of* LAV *or* GAV*, and $\mathcal{Q}_1$ and $\mathcal{Q}_2$ two query languages.* Mapping synthesis*, denoted,*

$$\text{MSYNT}[t, f, \mathcal{Q}_1, \mathcal{Q}_2],$$

is the following problem: given a $\mathcal{Q}_1$-based schema mapping M of type t, find a $\mathcal{Q}_2$-based schema mapping M' of type t and form f such that $M' \models M$, and $M' \not\models_{triv} M$.

To rule out (uninteresting) cases where all LAV (or GAV) mappings that imply a given mapping M do so trivially, in the following we require that for each mapping assertion $q_s \rightsquigarrow q_t \in M$, both q_s and q_t are different from false.

In general we are interested in the tightest simplification of a mapping M, i.e., the simplification that better approximates M. Hence, we also consider the *maximal mapping synthesis* problem, $\text{MAXMSYNT}[t, f, \mathcal{Q}_1, \mathcal{Q}_2]$, where, given a $\mathcal{Q}_1$-based mapping M of type t, we aim at computing a $\mathcal{Q}_2$-based mapping M' of type t and form f such that $M' \models M$ and there is no $\mathcal{Q}_2$-based mapping M'' of type t and form f such that $M'' \models M$, $M' \models M''$, and $M'' \not\models M'$.

In this paper we study the above problems for a variety of cases, where $\mathcal{Q}_1$ and $\mathcal{Q}_2$ range over (U)CQs and variants of queries over graph databases.

We start by observing that we can characterize mapping implication, and hence mapping simplification, in terms of query unfolding wrt a set of mappings. We make use of such a characterization in the technical development in the subsequent sections. The notion of query unfolding is formally defined as follows: let q_s be a source query and M_L a LAV mapping. The *unfolding of q_s wrt M_L*, denoted $q_s[M_L]$, is the target query obtained by replacing each atom α in q_s whose predicate symbol is a with $M_L(a)$. An analogous definition holds for the unfolding $q_t[M_G]$ of a target query q_t wrt a GAV mapping M_G.

THEOREM 6. *Let $\mathcal{Q}$ be a monotone query language.*

(1) Let M_L be a LAV *mapping and M a mapping, both $\mathcal{Q}$-based and of type* SOUND *(resp.,* EXACT*). Then $M_L \models M$ iff for each assertion $q_s \rightsquigarrow q_t$ in M, we have that $q_s[M_L] \sqsubseteq q_t$ (resp., $q_s[M_L] \equiv q_t$).*

(2) Let M_G be a GAV *mapping and M a mapping, both $\mathcal{Q}$-based and of type* SOUND *(resp.,* EXACT*). Then $M_G \models M$ iff for each assertion $q_s \rightsquigarrow q_t$ in M, we have that $q_s \sqsubseteq q_t[M_G]$ (resp., $q_s \equiv q_t[M_G]$).*

PROOF (SKETCH). We provide the proof for (1), in particular for the case of sound mappings. The other cases can be proved analogously.

We start with the following observation, which is easy to prove: if M_L is a LAV mapping, and $\mathcal{D}_s$ is the source database obtained from a target database $\mathcal{D}_t$ by letting $r^{\mathcal{D}_s} = M_L(r)^{\mathcal{D}_t}$ for every $r \in \Sigma_s$, then for every source query q, we have that $q^{\mathcal{D}_s} = q[M_L]^{\mathcal{D}_t}$.

"$\Rightarrow$" Now, assume that there is an assertion $q_s \rightsquigarrow q_t$ in M such that $q_s[M_L] \not\sqsubseteq q_t$, and let $\mathcal{D}_t$ be such that for some tuple $\mathbf{d}$ we have $\mathbf{d} \in q_s[M_L]^{\mathcal{D}_t}$, and $\mathbf{d} \notin q_t^{\mathcal{D}_t}$. Let $\mathcal{D}_s$ be the source database obtained from $\mathcal{D}_t$ by letting $r^{\mathcal{D}_s} = M_L(r)^{\mathcal{D}_t}$ for every $r \in \Sigma_s$. Clearly, $(\mathcal{D}_s, \mathcal{D}_t) \models M_L$. By the above observation, we have that $q_s^{\mathcal{D}_s} = q_s[M_L]^{\mathcal{D}_t}$, and therefore $\mathbf{d} \in q_s^{\mathcal{D}_s}$. It follows that $(\mathcal{D}_s, \mathcal{D}_t) \not\models q_s \rightsquigarrow q_t$, and $M_L \not\models M$.

"$\Leftarrow$" Assume that $M_L \not\models M$, i.e., there is an assertion $q_s \rightsquigarrow q_t$ in M, and a pair $(\mathcal{D}_s, \mathcal{D}_t)$ such that $(\mathcal{D}_s, \mathcal{D}_t) \models M_L$, and $q_s^{\mathcal{D}_s} \not\subseteq q_t^{\mathcal{D}_t}$, which means that there is $\mathbf{d}$ such that $\mathbf{d} \in q_s^{\mathcal{D}_s}$ but $\mathbf{d} \notin q_t^{\mathcal{D}_t}$. Let $\mathcal{D}'_s$ be such that $r^{\mathcal{D}'_s} = M_L(r)^{\mathcal{D}_t}$ for every $r \in \Sigma_s$. Clearly, $\mathcal{D}'_s \supseteq \mathcal{D}_s$, and $(\mathcal{D}'_s, \mathcal{D}_t) \models M_L$. Since q_s is monotone, we have that $\mathbf{d} \in q_s^{\mathcal{D}'_s} = q_s[M_L]^{\mathcal{D}_t}$, and therefore $q_s[M_L] \not\sqsubseteq q_t$. $\square$

THEOREM 7. *Let M be a mapping and M' a* LAV *(resp.,* GAV*) mapping. Then $M' \models_{triv} M$ iff for some mapping assertion $q_s \rightsquigarrow q_t \in M$ we have that $q_s[M'] \sqsubseteq$ false (resp., true $\sqsubseteq q_t[M']$).*

PROOF SKETCH. Similar to that of Theorem 6. $\square$

For many of the results in the next sections, we make use of the above characterization, without further mentioning Theorems 6 and 7.

4. LAV SIMPLIFICATION FOR (U)CQs

In this section, we consider the case of mappings based on conjunctive queries (CQs) and their unions (UCQs), and study the problem of simplifying a given mapping M in terms of a LAV mapping. The techniques we adopt for establishing our upper bounds are based on determining a polynomial bound on the length of the queries to consider when searching for the LAV mapping logically implying M, and are reminiscent of those in [39].

In the following, when we refer to a LAV mapping logically implying a given mapping, we implicitly assume that implication is non-trivial. We start with the problem of simplifying a CQ-based mapping in terms of a CQ-based LAV mapping.

THEOREM 8. *Both* MSIMP[SOUND,LAV,CQ,CQ] *and* MSIMP[EXACT,LAV,CQ,CQ] *are in NP.*

PROOF. Consider a sound CQ-based mapping consisting of a single assertion $q_s \rightsquigarrow q_t$, where q_t contains ℓ_{q_t} atoms, and a sound CQ-based LAV mapping M_L such that $q_s[M_L] \sqsubseteq q_t$. Then, there exists a homomorphism from q_t to $q_s[M_L]$, and at most ℓ_{q_t} atoms of $q_s[M_L]$ are in the image of this homomorphism. Hence, for each symbol $a \in \Sigma_s$ occurring in q_s, only at most ℓ_{q_t} atoms in query $M_L(a)$ are needed for the homomorphism. In the general case where the mapping M consists of several assertions, for each $a \in \Sigma_s$ we need at most $\ell_M = \sum_{q_s \rightsquigarrow q_t \in M} \ell_{q_t}$ atoms in the query $M_L(a)$, in order to guarantee the existence of the homomorphisms for all the assertions in M. Hence, in order to check for the existence of an appropriate LAV mapping M_L, it suffices to guess, for each symbol $a \in \Sigma_s$ appearing in one of the mapping assertions in M, a CQ $M_L(a)$ over Σ_t of size at most ℓ_M, and check that $q_s[M_L] \sqsubseteq q_t$, for each $q_s \rightsquigarrow q_t \in M$. In doing so, we rule out the guess of mappings that trivially imply M. This gives us immediately an NP upper bound for MSIMP[SOUND,LAV,CQ,CQ].

For MSIMP[EXACT,LAV,CQ,CQ], in addition to checking that $q_s[M_L] \sqsubseteq q_t$, we need also to check that $q_t \sqsubseteq q_s[M_L]$. We observe that the bound on the number of atoms in $M_L(a)$ derived for the sound case is still valid, since if $q_t \sqsubseteq q_s[M_L]$ for a LAV mapping M_L, then also $q_t \sqsubseteq q_s[M'_L]$ for every LAV mapping M'_L such that $M'_L(a)$ is constituted by a subset of the atoms of $M_L(a)$. Therefore, the overall complexity does not change. $\square$

For the case where M is UCQ-based, and the LAV mapping M_L is still CQ-based, we can generalize the above argument by considering containment between UCQs instead of containment between CQs.

THEOREM 9. *Both* MSIMP[SOUND,LAV,UCQ,CQ] *and* MSIMP[EXACT,LAV,UCQ,CQ] *are in NP.*

The last case we consider is the one where both M and the LAV mapping M_L are UCQ-based. In the sound case, we show that simplification to a UCQ-based LAV mapping is equivalent to simplification to a CQ-based LAV mapping.

LEMMA 10. MSIMP[SOUND,LAV,UCQ,UCQ] *admits a solution for a mapping M iff* MSIMP[SOUND,LAV,UCQ,CQ] *admits a solution for M.*

PROOF. Let M be a sound UCQ-based mapping and M_L a sound UCQ-based LAV mapping such that $M_L \models M$. Consider the CQ-based LAV mapping M'_L obtained from M_L by choosing, for each symbol a in Σ_s, as $M'_L(a)$ one of the CQs in $M_L(a)$. We show that $M'_L \not\models_{triv} M$, and that $M'_L \models M$. Consider one assertion $q_s \rightsquigarrow q_t \in M$ such that $q_s[M_L]$ is a non-empty positive query. Such a mapping assertion exists, since $M_L \not\models_{triv} M$. Then, $q_s[M'_L]$ is a non-empty UCQ, and hence $M'_L \not\models_{triv} M$. To show that $M'_L \models M$, it is sufficient to observe that, for each assertion $q_s \rightsquigarrow q_t \in M$, each CQ in $q_s[M'_L]$ is contained in $q_s[M_L]$, and hence in q_t. $\square$

By the above lemma, we trivially get:

THEOREM 11. MSIMP[SOUND,LAV,UCQ,UCQ] *is in NP.*

In the exact case, if we allow for UCQ-based LAV mappings, we get a higher upper-bound.

THEOREM 12. MSIMP[EXACT,LAV,UCQ,UCQ] *is in Π_2^p.*

We now show that the upper bounds for the sound cases established in Theorems 8, 9, and 11 are tight.

THEOREM 13. MSIMP[SOUND,LAV,CQ,CQ] *is NP-hard.*

PROOF. The proof is by a reduction from 3-colorability. Consider a graph $G = (N, E)$, with $N = \{n_1, \ldots, n_k\}$.

Let $\Sigma_s = \{t/2, a_s/2, a_f/2\}$, $\Sigma_t = \{e/2, b_s/2, b_f/2\}$,

$$\begin{aligned} q_T = \{(s, f) \mid\ & a_s(s, r), a_s(s, g), a_s(s, b), \\ & t(r, g), t(g, r), t(r, b), t(b, r), t(g, b), t(b, g), \\ & a_f(r, f), a_f(g, f), a_f(b, f)\ \} \end{aligned}$$

$$\begin{aligned} q_G = \{(s, f) \mid\ & b_s(s, x_1), \ldots, b_s(s, x_k), \\ & \textstyle\bigwedge_{(n_i, n_j) \in E} \{e(x_i, x_j), e(x_j, x_i)\}, \\ & b_f(x_1, f), \ldots, b_f(x_k, f)\ \} \end{aligned}$$

and define the following mapping M:

$$q_T \rightsquigarrow q_G \quad (1)$$
$$\{(x,y) \mid t(x,y)\} \rightsquigarrow \{(x,y) \mid e(x,y)\} \quad (2)$$
$$\{(x,y) \mid a_s(x,y)\} \rightsquigarrow \{(x,y) \mid b_s(x,y)\} \quad (3)$$
$$\{(x,y) \mid a_f(x,y)\} \rightsquigarrow \{(x,y) \mid b_f(x,y)\} \quad (4)$$

Intuitively, assertion (1) maps a triangle, whose three vertexes are connected by a_s and a_f to the distinguished variables s and f respectively, to the graph G, whose nodes are connected by b_s and b_f to the distinguished variables s and f respectively.

One can show that G is 3-colorable iff MSimp[sound,lav,cq,cq] with input M admits a solution. □

Corollary 14. *MSimp[sound,lav,ucq,cq] and MSimp[sound,lav,ucq,ucq] are NP-hard.*

Proof. From Theorem 13 we trivially get the result for MSimp[sound,lav,ucq,cq], and by considering Lemma 10, we get the result also for MSimp[sound,lav,ucq,ucq]. □

We conjecture that the upper bounds we provided for the case where schema mappings are of type exact are tight.

5. LAV SIMPLIFICATION FOR RPQs

In this section, we consider the case of RPQs over graph databases, and study the problem of simplifying an rpq-based mapping in terms of an rpq-based lav mapping. For our results, we exploit a straightforward language theoretic characterization of containment between RPQs.

Theorem 15 ([21]). *Let q_1, q_2 be two RPQs, and $\mathcal{L}(q_1)$, $\mathcal{L}(q_2)$ the corresponding regular languages. Then $q_1 \sqsubseteq q_2$ iff $\mathcal{L}(q_1) \subseteq \mathcal{L}(q_2)$.*

In the following, we identify an RPQ q over an alphabet Σ with the language over Σ accepted by the regular expression (RE) or 1NFA representing q. Considering the language-theoretic characterization above, it follows from Theorems 6 and 7 that, if M_L is a lav mapping and M a mapping, both of type sound (resp., exact), then $M_L \models M$ and $M_L \not\models_{triv} M$ iff for each assertion $q_s \rightsquigarrow q_t$ in M, we have that $q_s[M_L] \subseteq q_t$ (resp., $q_s[M_L] = q_t$) and $q_s[M_L] \neq \emptyset$. Here, the unfolding $q_s[M_L]$ of q_s wrt M_L denotes the *language* over Σ_t obtained from q_s by expanding in each word in q_s each symbol $a \in \Sigma_s$ with the language $M_L(a)$.

We start by showing that we can characterize mapping implication $M_L \models M$ between a lav mapping M_L and a mapping M in terms of a single language containment (for sound mappings) or language equality (for exact mappings). For this, we extend the notion of unfolding of q_s wrt M_L to the case where q_s may contain additional symbols wrt those in Σ_s. In particular, the additional symbols are left unchanged by the unfolding.

Proposition 16. *Let M be an rpq-based mapping of type sound (resp., exact) from Σ_s to Σ_t, and let $\#$ be a symbol not in $\Sigma_s \cup \Sigma_t$. Then there are RPQs $q_{M,s}$ over $\Sigma_s \cup \{\#\}$ and $q_{M,t}$ over $\Sigma_t \cup \{\#\}$, both of size linear in M, such that an rpq-based lav mapping M_L of type sound (resp., exact) is a solution to MSynt[sound,lav,rpq,rpq] (resp., MSynt[exact,lav,rpq,rpq]) with input M iff $q_{M,s}[M_L] \subseteq q_{M,t}$ (resp., $q_{M,s}[M_L] = q_{M,t}$) and $q_{M,s}[M_L] \neq \emptyset$.*

Proof. Let $M = \{q_{1,s} \rightsquigarrow q_{1,t}, \ldots, q_{k,s} \rightsquigarrow q_{k,t}\}$. We set $q_{M,s} = q_{1,s}\cdot\#\cdots\#\cdot q_{k,s}$ and $q_{M,t} = q_{1,t}\cdot\#\cdots\#\cdot q_{k,t}$. Intuitively, the fresh symbol $\#$ acts as a separator for the different parts of $q_{M,s}$ and $q_{M,t}$. It is easy to verify that, for every lav mapping M_L, we have that $q_{i,s}[M_L] \subseteq q_{i,t}$ (resp., $q_{i,s}[M_L] = q_{i,t}$) for $i \in \{1,\ldots,k\}$ iff $q_{M,s}[M_L] \subseteq q_{M,t}$ (resp., $q_{M,s}[M_L] = q_{M,t}$), and that $q_{i,s}[M_L] \neq \emptyset$ for $i \in \{1,\ldots,k\}$ iff $q_{M,s}[M_L] \neq \emptyset$. □

In the following, let Σ_n be an alphabet of new symbols disjoint from Σ_s and Σ_t, and let $\Sigma'_s = \Sigma_s \cup \Sigma_n$ and $\Sigma'_t = \Sigma_t \cup \Sigma_n$. By Proposition 16, the problem MSimp[sound,lav,rpq,rpq] (or MSimp[exact,lav,rpq,rpq]) can be polynomially reduced to the problem of checking, whether for languages q_s over Σ'_s and q_t over Σ'_t there is a lav mapping M_L such that $q_s[M_L] \subseteq q_t$ (resp., $q_s[M_L] = q_t$) and $q_s[M_L] \neq \emptyset$.

Our technique for mapping simplification exploits a characterization of regular languages by means of congruence classes [43, 23, 42]. Let $A_t = (\Sigma'_t, S_t, s^0_t, \delta_t, F_t)$ be a 1NFA for q_t. Then A_t defines a set of congruence classes partitioning Σ'^*_t. Each congruence class is characterized by a binary relation $G \subseteq S_t \times S_t$ (i.e., a directed graph over S_t), and we define the congruence class associated with G as

$$\mathcal{L}(G) = \{w \in \Sigma'^*_t \mid \text{for all } s_1, s_2 \in S_t, \; s_2 \in \delta_t(s_1, w) \text{ iff } (s_1, s_2) \in G\}.$$

Intuitively, each word $w \in \mathcal{L}(G)$ connects s_1 to s_2 in A_t, for each pair $(s_1, s_2) \in G$. For a word $w \in \Sigma'^*_t$, we denote with $[w]_{A_t}$ the congruence class to which w belongs.

It follows immediately from the characterization of congruence classes in terms of binary relations over the states of A_t that the set of congruence classes is closed under concatenation. Specifically, for two congruence classes $\mathcal{L}(G_1)$ and $\mathcal{L}(G_2)$, respectively with associated relations G_1 and G_2, the binary relation associated with $\mathcal{L}(G_1) \cdot \mathcal{L}(G_2)$ is $G_1 \circ G_2$.[2]

We introduce some notation that we use here, and later in this section:

- $\mathcal{G}_t = 2^{S_t \times S_t}$ denotes the set of binary relations associated with the congruence classes for A_t,
- $G^\varepsilon_t = \{(s,s) \mid s \in S_t\}$, and
- $G^b_t = \{(s_1, s_2) \mid s_2 \in \delta_t(s_1, b)\}$, for each $b \in \Sigma'_t$.

Then, for each $G \in \mathcal{G}_t$, we can characterize the congruence class $\mathcal{L}(G)$ associated with G in terms of a DFA.

Lemma 17 ([43]). *The language $\mathcal{L}(G)$ is accepted by the DFA $A_G = (\Sigma'_t, \mathcal{G}_t, G^\varepsilon_t, \delta_{A_t}, F_G)$, where $F_G = \{G\}$ and $\delta_{A_t}(R, b) = R \circ G^b_t$, for each $R \subseteq S_t \times S_t$ and $b \in \Sigma'_t$.*

Notice that, if A_t has m states, then A_G has 2^{m^2} states.

We observe next that we need to allow for the presence of empty queries in the lav mapping we are looking for. Consider, e.g., $q_s = (a_1 + a_3) \cdot (a_2 + a_3)$ and $q_t = b_1 \cdot b_2$. It is easy to see that for all lav mappings M_L such that $q_s[M_L] \subseteq q_t$, we have that $M_L(a_3) = \emptyset$. One such lav mapping M_L is

$$M_L(a_1) = b_1, \qquad M_L(a_2) = b_2, \qquad M_L(a_3) = \emptyset.$$

Observe also that $[b_1]_{A_t} = \{b_1\}$ and $[b_2]_{A_t} = \{b_2\}$, where A_t is the obvious 1NFA for $b_1 \cdot b_2$.

[2]We use $L_1 \cdot L_2$ to denote concatenation between languages, and $G_1 \circ G_2$ to denote composition of binary relations.

5.1 Upper bounds for sound mappings

We first deal with the case of sound mappings, and prove two preliminary results. The first lemma states that w.l.o.g. we can restrict the attention to LAV mappings in which the queries are *singletons*, i.e., queries that are either empty or constituted by a single word.

LEMMA 18. *Let q_s be an RPQ over Σ'_s, and q_t an RPQ over Σ'_t. If there exists an* RPQ*-based* LAV *mapping M_L such that $q_s[M_L] \subseteq q_t$, then there exists an* RPQ*-based* LAV *mapping M'_L such that $q_s[M'_L] \subseteq q_t$ in which each query is either a single word over Σ_t or empty.*

The next lemma shows that one can close queries in LAV mappings under congruence.

LEMMA 19. *Let q_s be an RPQ over Σ'_s, q_t an RPQ over Σ'_t expressed through a 1NFA A_t, and M_L a singleton mapping such that $q_s[M_L] \subseteq q_t$. Then, for M'_L defined such that*

$$M'_L(a) = \begin{cases} [w^a]_{A_t}, & \text{if } M_L(a) = w^a \\ \emptyset, & \text{if } M_L(a) = \emptyset, \end{cases}$$

we have that $q_s[M'_L] \subseteq q_t$.

From these two lemmas we get that, when searching for a LAV mapping M_L satisfying $q_s[M_L] \subseteq q_t$, we can restrict the attention to queries that are congruence classes for A_t.

LEMMA 20. *Let q_s be an RPQ over Σ'_s, and q_t an RPQ over Σ'_t expressed through a 1NFA A_t. If there exists an* RPQ*-based* LAV *mapping M_L such that $q_s[M_L] \subseteq q_t$, then there exists an* RPQ*-based* LAV *mapping M'_L such that $q_s[M'_L] \subseteq q_t$, and such that $M'_L(a)$ is a congruence class for A_t, for each $a \in \Sigma_s$.*

PROOF. If there exist an RPQ-based LAV mapping M_L such that $q_s[M_L] \subseteq q_t$, then by Lemma 18, w.l.o.g., we can assume that M_L consists of singleton queries. Then, the claim follows from Lemma 19. □

We derive now a procedure that, given an RPQ q_s over Σ'_s, and an RPQ q_t over Σ'_t expressed respectively through 1NFAs $A_s = (\Sigma'_s, S_s, s^0_s, \delta_s, F_s)$ and $A_t = (\Sigma'_t, S_t, s^0_t, \delta_t, F_t)$, checks for the existence of a sound RPQ-based LAV mapping M_L such that
(1) $q_s[M_L] \neq \emptyset$, and
(2) $q_s[M_L] \subseteq q_t$.
Specifically, by Lemma 20, it is sufficient to consider LAV mappings in which each query is constituted by a single congruence class, which can be represented by a binary relation over the state set S_t of A_t. Hence, for each $a \in \Sigma_s$, we guess such a binary relation G_a and verify that for the LAV mapping M_L defined by $M_L(a) = \mathcal{L}(G_a)$, conditions (1) and (2) are satisfied. In doing so, we exploit Lemma 17, which provides a characterization of $\mathcal{L}(G_a)$ in terms of a DFA A_{G_a}.

To check condition (1), we proceed as follows:

1. for each $a \in \Sigma_s$, we check whether a is a *bad symbol*, i.e., whether $\mathcal{L}(G_a) = \emptyset$;
2. we delete from A_s each transition labeled by a bad symbol; and
3. we check whether the resulting 1NFA accepts a nonempty language.

To check condition (2), we proceed as follows:

1. we construct an 1NFA A_{M_L} accepting $q_s[M_L]$;
2. we construct the 1NFA $A_{\not\subseteq} = A_{M_L} \times \bar{A}_t$ as the product NFA of A_{M_L} and the 1NFA $\bar{A}_t$ accepting $\Sigma'^*_t \setminus q_t$;
3. we check $A_{\not\subseteq}$ for emptiness.

To construct A_{M_L}, we observe that a word w is in $q_s[M_L]$ if there is a word $a_1 \cdots a_n \in q_s$, and for $i \in \{1, \ldots, n\}$, words $w_i \in \mathcal{L}(G_{a_i})$ such that $w = w_1 \cdots w_n$. Hence, A_{M_L} simulates A_s while accepting words in Σ'_t that are concatenations of words in the various languages $\mathcal{L}(G_{a_i})$. Specifically, $A_{M_L} = (\Sigma'_t, S_{M_L}, s^0_{M_L}, \delta_{M_L}, F_{M_L})$, where

- $S_{M_L} = \Sigma'_s \times S_s \times \mathcal{G}$;
- $s^0_{M_L} = \Sigma'_s \times \{s^0_s\} \times G^\varepsilon_t$;
- $F_{M_L} = \{(a, s, G_a) \mid s \in F_s, a \in \Sigma_s\} \cup \{(b, s, G^b_t) \mid s \in F_s, b \in \Sigma_n\}$;
- and for each $a \in \Sigma'_s$, $s \in S_s$, $R \in \mathcal{G}$, and $b \in \Sigma'_t$,

$$\delta_{M_L}((a, s, R), b) = \begin{cases} \{(a, s, R \circ G^b_t)\}, & \text{if } a \in \Sigma_s, R \neq G_a; \\ \{(a, s, R \circ G^b_t)\} \cup & \\ \bigcup_{a' \in \Sigma'_s, s' \in \delta_s(s,a)} \{(a', s', G^b_t)\}, & \text{if } a \in \Sigma_s, R = G_a; \\ \bigcup_{a' \in \Sigma'_s, s' \in \delta_s(s,a)} \{(a', s', G^\varepsilon_t)\}, & \text{if } a \in \Sigma_n, a = b; \\ \emptyset, & \text{otherwise.} \end{cases}$$

THEOREM 21. MSIMP[SOUND,LAV,RPQ,RPQ] *is in* PSPACE.

PROOF. By Lemma 20, to check whether MSIMP[SOUND,LAV,RPQ,RPQ] admits a solution, it suffices guess for each symbol $a \in \Sigma_s$ a binary relation G_a over the state set S_t of A_t, and check whether for the resulting RPQ-based LAV mapping M_L conditions (1) and (2) hold. When checking condition (1), the emptiness test in item (1) can be done for each $a \in \Sigma_s$ in NLOGSPACE in $|A_{G_a}|$, and since the number of states of A_{G_a} is exponential in $|A_t|$, in PSPACE in $|A_t|$. The non-emptiness test in item (3) can be done in NLOGSPACE in $|A_s|$. When checking condition (2), we do not need to construct A_{M_L}, $\bar{A}_t$, and $A_{\not\subseteq}$ explicitly, but can check the nonemptiness of $A_{\not\subseteq}$ on the fly while constructing A_{M_L} and complementing A_t. Hence, since the number of states of A_{M_L} is linear in $|A_s|$ and exponential in $|A_t|$, we get that condition (2) can be checked in PSPACE in $|A_t|$ and in NLOGSPACE in $|A_s|$. □

5.2 Upper bounds for exact mappings

The method based on congruence classes can be adapted to address also LAV simplification for exact mappings. The difference wrt sound mappings is that in this case we need to consider also LAV mappings in which the queries are unions of congruence classes. Indeed, congruence classes (and hence solutions to the LAV mapping synthesis problem) are not closed under union, as shown by the following example.

Let $q_s = a_1 \cdot a_2$ and $q_t = 00 + 01 + 10$. Then the following two incomparable mappings are solutions to MAXMSYNT[SOUND,LAV,RPQ,RPQ] when the input mapping is $\{q_s \rightsquigarrow q_t\}$:

$$M^1_L(a_1) = 0, \qquad M^2_L(a_1) = 0 + 1,$$
$$M^1_L(a_2) = 0 + 1. \qquad M^2_L(a_2) = 0.$$

Notice that the mapping M_L, where $M_L(a_i) = M^1_L(a_i) + M^2_L(a_i)$, for $i \in \{1, 2\}$, is not a solution, since $q_s[M_L]$ includes 11.

On the other hand, we can show that considering mapping in which the queries are unions of congruence classes is sufficient to obtain maximal unfoldings. We first generalize Lemma 19 to non-singleton queries.

LEMMA 22. *Let q_s be an RPQ over Σ'_s, q_t an RPQ over Σ'_t expressed through an 1NFA A_t, and M_L a* LAV *mapping such that $q_s[M_L] \subseteq q_t$. Then for M'_L with*

$$M'_L(a) = \begin{cases} \bigcup_{w \in M_L(a)} [w]_{A_t}, & \text{if } M_L(a) \neq \emptyset \\ \emptyset, & \text{if } M_L(a) = \emptyset. \end{cases}$$

we have that $q_s[M'_L] \subseteq q_t$.

PROOF. Consider a word $a_1 \cdots a_h \in q_s$. If there is one of the a_i such that $M_L(a_i) = \emptyset$, then $M_L(a_1) \cdots M_L(a_h) = \emptyset \subseteq q_t$. Otherwise, we have that, for $i \in \{1, \ldots, h\}$, for some $w^{a_i} \in M_L(a_i)$, the word $w^{a_1} \cdots w^{a_h} \in q_s[M_L] \subseteq \mathcal{L}(A_t)$. We show that, for each $i \in \{1, \ldots, h\}$, we also have that $w^{a_1} \cdots w^{a_{i-1}} \cdot w' \cdot w^{a_{i+1}} \cdots w^{a_h} \in \mathcal{L}(A_t)$, for each $w' \in \bigcup_{w \in M_L(a_i)} [w]_{A_t}$. First, since $q_s[M_L] \subseteq q_t$, if $w^{a_1} \cdots w^{a_h} \in q_s[M_L] \subseteq \mathcal{L}(A_t)$, then, for each $w \in M_L(a_i)$, we also have that $w^{a_1} \cdots w^{a_{i-1}} \cdot w \cdot w^{a_{i+1}} \cdots w^{a_h} \in q_s[M_L] \subseteq \mathcal{L}(A_t)$. Then there is a sequence $s_0, s_1, \ldots, s_h$ of states of A_t such that $s_0 = s_t^0$, $s_h \in F_t$, $s_j \in \delta_t(s_{j-1}, w^{a_j})$, for $j \in \{1, \ldots, i{-}1, i{+}1, \ldots, h\}$, and $s_i \in \delta_t(s_{i-1}, w)$. Then, by the definition of congruence classes, for each word $w' \in [w]_{A_t}$, we have that $s_i \in \delta_t(s_{i-1}, w'_i)$, and hence $w^{a_1} \cdots w^{a_{i-1}} \cdot w' \cdot w^{a_{i+1}} \cdots w^{a_h} \in \mathcal{L}(A_t)$. □

The above lemma implies that, when searching for maximal LAV mappings that imply a given mapping, we can restrict the attention to queries that are unions of congruence classes.

LEMMA 23. *Given a mapping $M = \{q_s \rightsquigarrow q_t\}$, where q_t is defined by a 1NFA A_t, every solution M_L to* MAXMSYNT[SOUND,LAV,RPQ,RPQ] *with input M is such that each query in M_L is a union of congruence classes for A_t.*

PROOF. Consider a solution M_L to MAXMSYNT[SOUND,LAV,RPQ,RPQ] with input M, and assume that for some $a \in \Sigma_s$, $M_L(a)$ is not a union of congruence classes for A_t. Then there is some word $w \in M_L(a)$ and some word $w' \in [w]_{A_t}$ such that $w' \notin M_L(a)$. By Lemma 22, the mapping M'_L with $M'_L(a) = M_L(a) \cup \{w'\}$ is also a solution to MAXMSYNT[SOUND,LAV,RPQ,RPQ] with input M, thus contradicting the maximality of M_L. □

We get the following upper bound for the LAV simplification in the exact case.

THEOREM 24. MSIMP[EXACT,LAV,RPQ,RPQ] *is in* EXPSPACE.

PROOF. By Lemma 23, we can nondeterministically choose mappings M_L in which the queries are unions of congruence classes and then test whether $q_t = q_s[M_L]$. To do so, we build a 1NFA A_{s,M_L} accepting $q_s[M_L]$ as follows. We start by observing that for each union U of congruence classes, we can build the automaton $A_U = (\mathcal{G}, s_t, R_\epsilon, \delta_{A_t}, U)$ accepting the words in U, which incidentally, is deterministic. Hence, by substituting each a-transition in the 1NFA A_s for q_s with the 1NFA A_{U_a}, where $M_L(a) = U_a$, we obtain a 1NFA A_{s,M_L}. Note that, even when A_s is deterministic, A_{s,M_L} may be nondeterministic.

To test $q_s[M_L] \subseteq q_t$, we complement A_t, obtaining the 1NFA $\overline{A_t}$, and check the 1NFA $A_{s,M_L} \times \overline{A_t}$ for emptiness. The size of $A_{s,M_L} \times \overline{A_t}$ is polynomial in the size of A_s and exponential in the size of A_t. Checking for emptiness can be done in exponential time, and considering the initial nondeterministic guess, we get a NEXPTIME upper bound.

To test $q_t \subseteq q_s[M_L]$, we complement A_{s,M_L}, obtaining the 1NFA $\overline{A_{s,M_L}}$, and check $A_t \times \overline{A_{s,M_L}}$ for emptiness. Since A_{s,M_L} is nondeterministic, complementation is exponential. However, we observe again that such a complementation can be done on the fly in EXPSPACE, while checking for emptiness and intersecting with A_t. As a consequence, considering the initial nondeteministic guess, MSIMP[EXACT,LAV,RPQ,RPQ] can be decided in NEXPSPACE, which is equivalent to EXPSPACE. □

Note that the proofs of Theorems 21 and 24 imply that, wrt LAV mapping simplification, considering queries that are RPQs (as opposed to general, possibly non-regular, path languages) is not a restriction, since the existence of general LAV mappings implies the existence of regular ones. This is also in line with a similar observation holding for the existence of rewritings of RPQs wrt RPQ views [21].

Finally, we observe that using the machinery based on unions of congruence classes, we can also solve the maximal mapping synthesis problem. We guess a mapping and check that it is a solution to mapping synthesis. To check that it is a maximal solution, we generate all other mappings and check that they are contained in our candidate solution.

THEOREM 25. *A solution to* MAXMSYNT[LAV,SOUND,RPQ,RPQ] *and to* MAXMSYNT[LAV,EXACT,RPQ,RPQ] *can be computed in* EXPSPACE.

5.3 Lower bounds

It turns out that the upper bound established for the sound case is tight:

THEOREM 26. MSIMP[SOUND,LAV,RPQ,RPQ] *is* PSPACE-*hard.*

PROOF. The proof is by a reduction from the universality problem for REs. Given an RE e over the alphabet $\Sigma_t = \{b_1, \ldots, b_n\}$, let $\Sigma_s = \{a_1, \ldots, a_n\}$, and let M_e be the mapping constituted by the following assertions:

$$\Sigma_s^* \rightsquigarrow e \quad (5)$$

$$a_1 \rightsquigarrow b_1 \qquad \cdots \qquad a_n \rightsquigarrow b_n \quad (6)$$

We show that e is universal iff MSIMP[SOUND,LAV,RPQ,RPQ] with input M_e admits a solution. For the "only-if" part, assume that e is universal and consider the LAV mapping M_L consisting of the mapping assertions (6). We have that $\Sigma_s^*[M_L] = \Sigma_t^*$, and since e is universal, also $\Sigma_s^*[M_L] \subseteq e$. For the "if-part", consider a LAV mapping M_L such that $q_s[M_L] \subseteq q_t$ and $q_s[M_L] \neq \emptyset$, for each mapping assertion $q_s \rightsquigarrow q_t$ in M_e. By the mapping assertions (6), we have that $M_L(a_i) \neq \emptyset$ (since a_i is the left-hand side of a mapping assertion) and that $M_L(a_i)$ must include b_i, for $i \in \{1, \ldots, n\}$, and hence $\Sigma_s^*[M_L] = \Sigma_t^*$. Since $\Sigma_s^*[M_L] \subseteq e$, we have that e is universal. □

It is easy to see that the above proof shows also PSPACE-hardness of simplification for exact mappings.

COROLLARY 27. MSIMP[EXACT,LAV,RPQ,RPQ] *is* PSPACE-*hard.*

However, we can get a tight lower bound for a generalization of MSIMP[EXACT,LAV,RPQ,RPQ] in which we allow the input mapping to contain both sound and exact mapping assertions. We do so by showing an EXPSPACE lower bound for a problem that is closely related to the mapping simplication problem.

Consider a finite alphabet Σ and a finite set $\mathcal{V}$ of variables. A *language constraint* is a statement of the form $e_1 \sqsubseteq e_2$, where e_1 and e_2 are regular expressions over $\Sigma \cup \mathcal{V}$. A *language-constraint problem* P is a finite set of language constraints. A solution to P is an assignment $\sigma : \mathcal{V} \to 2^{\Sigma^*}$, assigning a language over Σ to each variable in $\mathcal{V}$ such that $\mathcal{L}(e_1[\sigma]) = \mathcal{L}(e_2[\sigma])$. It is easy to express the universality problem for context-free grammars as a language-constraint problem, which implies that the latter is undecidable. Here we consider *left-handed* language constraint problems, where we allow constraints of the form $e_1 \sqsubseteq e_2$ and $e_1 = e_2$, but require that variables appear only in the left-hand side of the constraint. The technique for the exact version of the LAV mapping simplification problem can be used to show an EXPSPACE upper bound for solving left-handed language constraint problems. We now show a matching lower bound.

To prove the result we exploit a reduction from tiling problems [47, 16]. A *tile* is a unit square of one of several types and the *tiling problem* we consider is specified by means of a finite set Δ of tile types, two binary relations H and P over Δ, representing horizontal and vertical adjacency relations, respectively, and two distinguished tile types $t_S, t_F \in \Delta$. The tiling problem consists in determining whether, for a given number n in unary, a region of the integer plane of size $2^n \times k$, for some $k > 0$, can be tiled consistently with the adjacency relations H and P, and with the left bottom tile of the region of type t_S and the right upper tile of type t_F. We also require that the last tile of a row and the first tile of the next row are consistent with H. Using a reduction from acceptance of EXPSPACE Turing machines analogous to the one in [47], it can be shown that this tiling problem is EXPSPACE-complete.

THEOREM 28. *Solving left-handed language constraints is* EXPSPACE-*complete.*

PROOF SKETCH. We sketch the lower-bound argument.

Let $T = (\Delta, H, P, t_S, t_F)$ be an instance of the EXPSPACE-complete tiling problem above and n a number in unary. The alphabet is $\Sigma = \Delta \cup \{0,1\}^3 \cup \{\#\}$. Intuitively, the letters in Δ denote tiles, symbols in $\{0,1\}^3$ denote address bits, and $\#$ denotes a separation marker. The idea is to encode each tiled cell by a word of length $n+2$ of the form $\# \cdot (\{0,1\}^3)^n \cdot \Delta$, consisting of a marker, an n-bit address, and a tile symbol. We use an element in $\{0,1\}^3$ for each address bit to make it easy to check that two n-bit addresses are consecutive; we use n-bits for the current address, n bits for the carry, and n bits for the next address. Thus, each tiling can be described by a word in $(\# \cdot (\{0,1\}^3)^n \cdot \Delta)^*$, obtained by encoding each cell as described above, and then concatenating the symbols, first column by column and then row by row.

Consider a word $w \in \Sigma^*$. Such a word does not describe a proper tiling if one of the following *errors* can be found in the word:

1. The symbol $\#$ does not occur precisely in positions $(n+2)i$, for $i = 0, 1, \ldots$.
2. The symbols in Δ do not occur precisely in positions $n+1+(n+2)i$, for $i = 0, 1, \ldots$.
3. The first address is not 0^n.
4. The last address is not 1^n.
5. There is a pair of adjacent but not successive addresses.
6. The first tile is not t_S.
7. The last tile is not t_F.
8. There is a pair of adjacent blocks with tiles that violate the relation H.
9. There is a pair of *vertically adjacent* blocks with tiles that violate the relation P.

We do need to define the notion of vertical adjacency. Two blocks are vertically adjacent if their addresses agree and either both addresses are 0^n and there is no occurrence of 0^n between them, or both addresses are not 0^n and there is precisely one ocurrence of 0^n between them.

If the tiling problem has no solution, then every word in Σ^* must contain an error. We now define a constraint of the form $e_{error} = \Sigma^*$, where the "task" of e_{error} is to discover errors in candidate words. The expression e_{error} is the sum of several terms corresponding to the various errors. We now sketch how to "discover" these possible errors. In order to have the left-hand sides use only variables, we introduce a variable v_a for each letter $a \in \Sigma$, accompanied by the constraint $v_a = a$. We use $\mathcal{V}_\Sigma$ to abbreviate $\sum_{a \in \Sigma} v_a$.

Most of the errors can be discovered with a single regular expression. For example, the error where the symbol $\#$ does not occur precisely in positions $(n+2)i$, for $i = 0, 1, \ldots$, is described using the expression

$$(\mathcal{V}_\Sigma^{n+2})^* \cdot (\textstyle\sum_{1 \le i \le n+1} \mathcal{V}_\Sigma^i) \cdot \# \cdot \mathcal{V}_\Sigma^*$$

The one error that is challenging is where there is a pair of *vertically adjacent* blocks with tiles that violate the relation P. Discovering this error is more difficult and cannot be done by one regular expression; rather, several additional constraints are needed. For simplicity we ignore the fact that each address bit is encoded by three bits rather than one.

Let e_{nza} be a regular expression that describes nonzero addresses: $\sum_{0 \le i \le n-1} \{0,1\}^i \cdot 1 \cdot \{0,1\}^{n-i-1}$.

We add to e_{error} the following term, which discovers non-matching tiles at zero-addressed vertically adjacent tiles.

$$\textstyle\sum_{(t,t') \notin P} \left(\mathcal{V}_\Sigma^* \cdot \# \cdot 0^n \cdot t \cdot (\# \cdot e_{nza} \cdot \Delta)^* \cdot \# \cdot 0^n \cdot t' \cdot \mathcal{V}_\Sigma^*\right)$$

We need to deal with non-zero-addessed vertically adjacent blocks. For this we use several constraints. First:

$$v_{nzava} \sqsubseteq \textstyle\sum_{(t,t') \notin P} (\# \cdot e_{nza} \cdot t \cdot (\# \cdot e_{nza} \cdot \Delta)^* \cdot \# \cdot 0^n \cdot \Delta \cdot (\# \cdot e_{nza} \cdot \Delta)^* \cdot \# \cdot e_{nza} \cdot t')$$

This says that v_{nzava} describes sequences of blocks that start and end with a pair of non-matching non-zero-addressed blocks, with a single zero-addressed block in between. We still have to impose the constraint that the first and last block have equal addresses. We do this with n constraints, one for each bit of the address. That is for each i, $0 \le i \le n-1$, we add the constraint:

$$\begin{aligned} v_{nzava} \sqsubseteq\ & (\# \cdot \{0,1\}^i \cdot 0 \cdot \{0,1\}^{n-i-1} \cdot \Delta \cdot (\# \cdot \{0,1\}^n \cdot \Delta)^* \cdot \\ & \# \cdot \{0,1\}^i \cdot 0 \cdot \{0,1\}^{n-i-1} \cdot \Delta) + \\ & (\# \cdot \{0,1\}^i \cdot 1 \cdot \{0,1\}^{n-i-1} \cdot \Delta \cdot (\# \cdot \{0,1\}^n \cdot \Delta)^* \cdot \\ & \# \cdot \{0,1\}^i \cdot 1 \cdot \{0,1\}^{n-i-1} \cdot \Delta) \end{aligned}$$

This constraint says that the i-th bits of the first and last addresses are either both 0 or both 1.

Now we can add to e_{error} the term $\mathcal{V}_\Sigma^* \cdot v_{nzava} \cdot \mathcal{V}_\Sigma^*$, which discovers all errors due to not-matching, non-zero-addressed vertically adjacent blocks.

Note that the constraint system constructed is of size quadratic in the size of the tiling system. If the tiling problem has no solution, then every word in Σ^* contains an error and the contraint problem constructed is satisfiable. If the tiling problem has a solution, then a word describing a proper tiling has no error, and for no assignment $\sigma : \mathcal{V} \to 2^{\Sigma^*}$ we have $\mathcal{L}(e_{error}[\sigma]) = \Sigma^*$, since e_{error} captures only errors. □

Let MSIMP[MIXED,LAV,RPQ,RPQ], be the following decision problem: given an RPQ-based schema mapping M consisting both of SOUND and of EXACT mapping assertions, check whether there exists an RPQ-based LAV schema mapping M' of type EXACT such that $M' \models M$, and $M' \not\models_{triv} M$.

As a corollary of Theorem 28, we get the following result.

COROLLARY 29. MSIMP[MIXED,LAV,RPQ,RPQ] *is* EXPSPACE-*complete.*

6. EXTENSIONS

In this section we sketch the extension of the results of the previous section on simplification in terms of LAV mappings to more expressive classes of queries: 2RPQs, CRPQs, UCRPQs, and UC2RPQs.

6.1 2RPQs

Consider now simplification for mappings based on 2RPQs, expressed by means of 1NFAs over the alphabets $\Sigma_s^\pm$ and $\Sigma_t^\pm$.

A key concept for 2RPQs is that of *folding*, of a language [22], which intuitively denotes the set of words that are the result of repeatedly cancelling out adjacent occurrences of a symbol and its inverse. Let $u, v \in \Sigma^\pm$. We say that v *folds* onto u, denoted $v \rightsquigarrow u$, if v can be "folded" on u, e.g., $abb^-bc \rightsquigarrow abc$. Formally, we say that $v = v_1 \cdots v_m$ folds onto $u = u_1 \cdots u_n$ if there is a sequence $i_0, \ldots, i_m$ of positive integers between 0 and $|u|$ such that

- $i_0 = 0$ and $i_m = n$, and
- for $j \in \{0, \ldots, m\}$, either $i_{j+1} = i_j + 1$ and $v_{j+1} = u_{i_j+1}$, or $i_{j+1} = i_j - 1$ and $v_{j+1} = u^-_{i_{j+1}}$.

Let L be a language over $\Sigma^\pm$. We define $fold(L) = \{u \mid v \rightsquigarrow u, v \in L\}$.

A language-theoretic characterization for containment of 2RPQs was provided in [22]:

LEMMA 30. *Let q_1 and q_2 be 2RPQs. Then $q_1 \sqsubseteq q_2$ iff $\mathcal{L}(q_1) \subseteq fold(\mathcal{L}(q_2))$.*

Furthermore, it is shown in [22] that if A is an n-state 1NFA over $\Sigma^\pm$, then there is a 2NFA for $fold(\mathcal{L}(A))$ with $n \cdot (|\Sigma^\pm| + 1)$ states. (We use 2NFA to refer to two-way automata.)

In the mapping simplification problem, we are given queries q_s and q_t, expressed as 1NFAs A_s and A_t, respectively, and we are asked whether there exist a 2RPQ-based LAV mapping M_L such that $q_s[M_L] \sqsubseteq q_t$ or $q_s[M_L] = q_t$, and also $q_s[V] \not\equiv \emptyset$.

Here we can still use Lemma 23 for the congruence-class based solution. A simplistic approach would be to convert the 2NFA for $fold(\mathcal{L}(A_t))$ into a 1NFA, with an exponential blow-up, and proceed as in Section 5. To avoid this exponential blowup, we need an exponential bound on the number of congruence classes. For a 1NFA, we saw that each congruence class can be defined in terms of a binary relation over its set of states. It turns out that for a 2NFA A, a congruence class can be defined in terms of *four* binary relations over the set S_t of states of A:

1. R_{lr}: a pair $(s_1, s_2) \in R_{lr}$ means that there is a word w that leads A from s_1 to s_2, where w is entered on the left and exited on the right.
2. R_{rl}: a pair $(s_1, s_2) \in R_{rl}$ means that there is a word w that leads A from s_1 to s_2, where w is entered on the right and exited on the left.
3. R_{ll}: a pair $(s_1, s_2) \in R_{ll}$ means that there is a word w that leads A from s_1 to s_2, where w is entered on the left and exited on the left.
4. R_{rr}: a pair $(s_1, s_2) \in R_{rr}$ means that there is a word w that leads A from s_1 to s_2, where w is entered on the right and exited on the right.

Thus, the number of congruence classes when A has m states is 2^{4m^2} rather than 2^{m^2}, which is still an exponential. This enables us to adapt the technique of Section 5 with essentially the same complexity bounds.

THEOREM 31. MSIMP[SOUND,LAV,2RPQ,2RPQ] *is* PSPACE-*complete.* MSIMP[EXACT,LAV,2RPQ,2RPQ] *is in* EXPSPACE.

6.2 CRPQs and UC2RPQs

Consider now the mapping simplification problem for the case where the input mapping is expressed in terms of CRPQs, where the constituent RPQs are expressed by means of 1NFAs. Here the LAV mappings have to be in terms of RPQs, rather than CRPQs, since CRPQs are not closed under substitutions. The crux of our approach is to reduce containment of two CRPQs, q_1 and q_2 to containment of standard languages. This was done in [19]. Let q_h, for $h = \{1, 2\}$, be in the form

$$q_h = \{ (x_1, \ldots, x_n) \mid q_{h,1}(y_{h,1}, y_{h,2}) \land \cdots \land q_{h,m_h}(y_{h,2m_h-1}, y_{h,2m_h}) \}$$

and let $\mathcal{V}_1$, $\mathcal{V}_2$ be the sets of variables of q_1 and q_2 respectively. It is shown in [19] that the containment $q_1 \sqsubseteq q_2$ can be reduced to the containment $\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$ of two word automata A_1 and A_2. A_1 is a 1NFA, whose size is exponential in q_1 and it accepts certain words of the form

$$\$d_1 w_1 d_2 \$ d_3 w_2 d_4 \$ \cdots \$ d_{2m_1-1} w_{m_1} d_{2m_1} \$$$

where each d_i is a subset of $\mathcal{V}_1$ and the words w_i are over the alphabet of A_1. Such words constitute a linear representation of certain graph databases that are canonical for q_1 in some sense. A_2 is a 2NFA, whose size is exponential in the size of q_2, and it accepts words of the above form if the there is an appropriate mapping from q_2 to the database represented by these words. The reduction of the containment $q_1 \sqsubseteq q_2$ to $\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$ is shown in [19].

The ability to reduce containment of CRPQs to containment of word automata means that we can also apply the congruence-class technique of Section 5. Suppose that we have an RPQ-based LAV mapping M_L such that $\mathcal{L}(A_s[M_L]) \subseteq \mathcal{L}(A_t)$. Then we can again assume that the queries in the LAV mapping are closed with respect to the congruence classes of A_t. Thus, the techniques of Section 5 can be applied.

THEOREM 32. MSIMP[SOUND,LAV,CRPQ,RPQ] *is in* EXPSPACE. MSIMP[EXACT,LAV,CRPQ,RPQ] *is in* 2EXPSPACE.

Finally, we consider UC2RPQ queries, which combine UCQs and 2RPQs. This requires combining the techniques developed for RPQs, 2RPQs, and CRPQs. The key idea is the reduction of query containment to containment of word automata. The resulting upper bounds are identical to those we obtained for CRPQs.

7. GAV SIMPLIFICATION

In this section, we consider the case of simplifying mappings in terms of GAV mappings. Our results cover only a subset of the possible combination of the problem space, specifically GAV simplification is wide open for RPQ-based mappings. It should be noted that for exact mappings, the LAV case and GAV case coincide, so the results from Section 4 apply; we focus therefore on sound mappings.

We start by considering sound CQ-based mappings.

THEOREM 33. MSIMP[SOUND,GAV,CQ,CQ] *is in NP.*

PROOF. Consider a sound CQ-based mapping M consisting of a single assertion $q_s \rightsquigarrow q_t$ and assume there exists some sound CQ-based GAV mapping M_G such that $q_s \sqsubseteq q_t[M_G]$, witnessed by a homomorphism h, and $q_t[M_G] \neq \mathsf{true}^{\mathcal{D}_t}$. We show that there is a sound CQ-based GAV mapping M'_G of bounded size such that $q_s \sqsubseteq q_t[M'_G]$. Indeed, since $q_t[M_G] \neq \mathsf{true}^{\mathcal{D}_t}$, there must be one distinguished variable x and one symbol $b \in \Sigma_t$, such that x occur in an atomic formula with the symbol b in $M_G[b]$. We now obtain $M'_G(b)$ from $M_G(b)$ by selecting in $M_G(b)$ such an atom. For all other symbols $b' \in \Sigma_t$, we take $M'_G(b')$ to be true. By constructing M'_G in this way, we have that the atoms in $q_t[M'_G]$ are a subset of the atoms in $q_t[M_G]$, and hence the projection of h on such atoms is still a homomorphism to q_s.

In the general case where the mapping M consists of k assertions, we can apply the above argument for each of the assertions in M. This shows that, if there exists some sound GAV mapping M_G such that $M_G \models M$, then there is also a GAV mapping M'_G such that $M'_G(b)$ has at most k atoms and $M'_G \models M$. Hence, in order to check the existence of an appropriate GAV mapping M_G, it suffices to guess (avoiding trivial mappings), for each symbol $b \in \Sigma_t$ appearing in M, a CQ $M_G(b)$ over Σ_s of size at most k, and check that $q_s \sqsubseteq q_t[M_G]$, for each $q_s \rightsquigarrow q_t \in M$. □

This result extends immediately to UCQ-based mappings, by checking containment between UCQs, instead of CQs.

THEOREM 34. MSIMP[SOUND,GAV,UCQ,CQ] *is in NP.*

For UCQ-based mappings, we get the same upper bound, with a somewhat subtler argument.

THEOREM 35. MSIMP[SOUND,GAV,UCQ,UCQ] *is in NP.*

PROOF. We consider first the case of a sound UCQ-based mapping M consisting of a single assertion $q_s \rightsquigarrow q_t$. Assume there exists some sound UCQ-based GAV mapping M_G such that $q_s \sqsubseteq q_t[M_G]$ and $q_t[M_G] \neq \mathsf{true}^{\mathcal{D}_t}$. First note that $q_s \sqsubseteq q_t[M_G]$, if for each CQ q_s^i of q_s we have that $q_s^i \sqsubseteq q_t[M_G]$. This means that there is a CQ q_t^i of q_t and a CQ q_b^i for each symbol $b \in \Sigma_t$ such that there is a homomorphism from $q_t^i[M'_G]$ to q_s^i, where $M'_G(b) = q_b^i$. Thus, if q_s is a union of l CQs, then we can assume that each $M_G[b]$ for $b \in \Sigma_t$ has at most l CQs. What is left is to bound the size of these CQs.

Let m be the number of CQs in q_t. Since $q_t[M_G] \neq \mathsf{true}^{\mathcal{D}_t}$, for each CQ q'_t of q_t, there must be one distinguished variable x and one symbol $b \in \Sigma_t$ such that x occurs in some atomic formula with the symbol b in each CQ q'_b of $M_G[b]$. Define M'_G to keep all such atomic formulas, and only such atomic formulas. $M'_G[b]$ contains at most lm atomic formulas. If we have k mapping assertions, then $M'_G[b]$ needs to contain only klm atomic formulas. To check the existence of a simplifying GAV mapping it suffices to guess a mapping M'_G under such a size bound and check that $q_s \sqsubseteq q_t[M'_G]$. □

We conjecture that the above upper bounds are tight.

8. CONCLUSIONS

We have introduced the problem of simplifying schema mappings based on logical implication. The problem comes in different forms, depending on the type of simplification to achieve, on whether the mappings are sound or exact, and on the types of queries used in the mappings. We have provided a formalization of the problem, and we have presented techniques and complexity bounds for both relational and graph databases. As we said in the introduction, this is the first investigation on comparing schema mappings for graph databases.

In this paper we have concentrated on LAV simplification, and we have discussed the GAV case only for relational schema mappings. In the future, we plan to continue investigating schema mapping simplification along different directions. In particular, we aim at addressing GAV simplification for graph databases, and we plan to study schema mapping simplification for tree-based (e.g., XML) semistructured data.

Acknowledgements

Work partially supported by the EU under the project ACSI (Artifact-Centric Service Interoperation), grant n. FP7-257593, by Regione Lazio under the project "Integrazione semantica di dati e servizi per le aziende in rete", and by NSF grants CCF-0613889, CCF-0728882, and CNS 1049862.

9. REFERENCES

[1] S. Abiteboul. Querying semi-structured data. In *Proc. of ICDT'97*, pages 1–18.

[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 PODS'98*, pages 254–265.

[4] S. Abiteboul, G. Gottlob, and M. Manna. Distributed XML design. In *Proc. of PODS 2009*, pages 247–258.

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

[6] B. Alexe, P. Kolaitis, and W.-C. Tan. Characterizing schema mappings via data examples. In *Proc. of PODS 2010*, pages 261–271.

[7] M. Arenas, P. Barcelo, R. Fagin, and L. Libkin. Locally consistent transformations and query answering in data exchange. In *Proc. of PODS 2004*.

[8] M. Arenas, R. Fagin, and A. Nash. Composition with target constraints. In *Proc. of ICDT 2010*.

[9] M. Arenas and L. Libkin. XML data exchange: Consistency and query answering. In *Proc. of PODS 2005*, pages 13–24.

[10] M. Arenas, J. Pérez, J. L. Reutte, and C. Riveros. Foundations of schema mapping management. In *Proc. of PODS 2010*, pages 227–238.

[11] 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.

[12] P. C. Arocena, A. Fuxman, and R. J. Miller. Composing local-as-view mappings: Closure and applications. In *Proc. of ICDT 2010*, pages 209–218.

[13] F. Baader and R. Küsters. Unification in a description logic with transitive closure of roles. In R. Nieuwenhuis and A. Voronkov, editors, *Proc. of LPAR 2001*, volume 2250 of *LNCS*, pages 217–232. Springer.

[14] B. Barcelò. Logical foundations of relational data exchange. *SIGMOD Record*, 38(1):49–58, 2009.

[15] B. Barcelò, C. A. Hurtado, L. Libkin, and P. T. Wood. Expressive languages for path queries over graph-structured data. In *Proc. of PODS 2009*.

[16] R. Berger. The undecidability of the dominoe problem. *Mem. Amer. Math. Soc.*, 66:1–72, 1966.

[17] P. A. Bernstein and H. Ho. Model management and schema mappings: Theory and practices. In *Proc. of VLDB 2007*, pages 1439–1440.

[18] P. Buneman. Semistructured data. In *Proc. of PODS'97*, pages 117–121.

[19] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Containment of conjunctive regular path queries with inverse. In *Proc. of KR 2000*.

[20] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Query processing using views for regular path queries with inverse. In *Proc. of PODS 2000*.

[21] 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.

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

[23] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. View synthesis from schema mappings. CoRR Technical Report abs/1003.1179, arXiv.org e-Print archive, Mar. 2010. Available at `http://arxiv.org/abs/1003.1179`.

[24] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proc. of STOC'77*, pages 77–90.

[25] I. F. Cruz, A. O. Mendelzon, and P. T. Wood. A graphical query language supporting recursion. In *Proc. of ACM SIGMOD*, pages 323–330, 1987.

[26] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. *Theor. Comp. Sci.*, 336(1):89–124, 2005.

[27] R. Fagin, P. G. Kolaitis, A. Nash, and L. Popa. Towards a theory of schema-mapping optimization. In *Proc. of PODS 2008*, pages 33–42.

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

[29] R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Reverse data exchange: Coping with nulls. In *Proc. of PODS 2009*, pages 23–32.

[30] 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.

[31] 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.

[32] E. Franconi and S. Tessaris. The logic of RDF and SPARQL: a tutorial. In *Proc. of PODS 2006*.

[33] A. Fuxman, P. G. Kolaitis, R. J. Miller, and W. C. Tan. Peer data exchange. *ACM Trans. on Database Systems*, 31(4):1454–1498, 2005.

[34] G. Gottlob, R. Pichler, and V. Savenkov. Normalization and optimization of schema mappings. *PVLDB*, 2(1):1102–1113, 2009.

[35] G. Grahne and A. O. Mendelzon. Tableau techniques for querying information sources through global schemas. In *Proc. of ICDT'99*, volume 1540 of *LNCS*, pages 332–347. Springer.

[36] A. Y. Halevy, A. Rajaraman, and J. Ordille. Data integration: The teenage years. In *Proc. of VLDB 2006*, pages 9–16.

[37] P. G. Kolaitis. Schema mappings, data exchange, and metadata management. In *Proc. of PODS 2005*.

[38] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS 2002*, pages 233–246.

[39] A. Y. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *Proc. of PODS'95*, pages 95–104.

[40] L. Libkin and C. Sirangelo. Data exchange and schema mappings in open and closed worlds. In *Proc. of PODS 2008*, pages 139–148.

[41] J. Madhavan and A. Y. Halevy. Composing mappings among data sources. In *Proc. of VLDB 2003*.

[42] W. Martens, M. Niewerth, and T. Schwentick. Schema design for XML repositories: Complexity and tractability. In *Proc. of PODS 2010*, pages 239–250.

[43] J.-E. Pin. Syntactic semigroups. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Language Theory*, volume 1, chapter 10, pages 679–746. Springer, 1997.

[44] Y. Sagiv and M. Yannakakis. Equivalences among relational expressions with the union and difference operators. *J. of the ACM*, 27(4):633–655, 1980.

[45] B. ten Cate and P. G. Kolaitis. Structural characterizations of schema-mapping languages. In *Proc. of ICDT 2009*, pages 63–72.

[46] J. D. Ullman. Information integration using logical views. In *Proc. of ICDT'97*, volume 1186 of *LNCS*.

[47] P. van Emde Boas. The convenience of tilings. In *Complexity, Logic, and Recursion Theory*, volume 187 of *Lecture Notes in Pure and Applied Mathematics*, pages 331–363. Marcel Dekker Inc., 1997.

APPENDIX: PROOFS

Proof of Theorem 9.

Consider a sound UCQ-based mapping M consisting of a single assertion $q_s \rightsquigarrow q_t$ and a sound CQ-based LAV mapping M_L such that $q_s[M_L] \sqsubseteq q_t$. We remind that $q_s[M_L] \sqsubseteq q_t$ if for each CQ q_1 in the UCQ $q_s[M_L]$ there is a CQ q_2 in the UCQ q_t such that $q_1 \sqsubseteq q_2$. Then, for each CQ q_1 in $q_s[M_L]$ there exists a homomorphism from some CQ q_2 in q_t to q_1, and for each $a \in \Sigma_s$ at most ℓ_{q_t} atoms in the query $M_L(a)$ are needed for the homomorphism, where ℓ_{q_t} is the maximum number of atoms among the CQs in q_t. In the general case where the mapping M consists of several assertions, for each $a \in \Sigma_s$ at most $\ell_M = \sum_{q_s \rightsquigarrow q_t \in M} \ell_{q_t}$ atoms in the query $M_L(a)$ are needed for the homomorphisms for all the assertions in M. Thus, in order to check for the existence of an appropriate LAV mapping M_L, it suffices to guess (again avoiding trivial implication), for each symbol $a \in \Sigma_s$ appearing in M, a CQ $M_L(a)$ over Σ_t of size at most $\ell_M = \sum_{q_s \rightsquigarrow q_t \in M} \ell_{q_t}$, and check for each $q_s \rightsquigarrow q_t \in M$ that $q_s[M_L] \sqsubseteq q_t$ (and $q_t \sqsubseteq q_s[M_L]$ for the exact variant). ☐

Proof of Theorem 12.

Consider an exact UCQ-based mapping M consisting of a single assertion $q_s \rightsquigarrow q_t$ and an exact UCQ-based LAV mapping M_L such that $M_L \models M$. Let m_{q_t} be the number of CQs in q_t, and ℓ_{q_t} the maximum number of atoms among the CQs in q_t. Let q'_{s,M_L} be obtained from $q_s[M_L]$ by distributing, for each atom α of q_s, the unions in the UCQ $\alpha[M_L]$ over the conjunctions of each CQ of q_s. Note that q'_{s,M_L} is a UCQ, and is equivalent to $q_s[M_L]$. Since $q_s[M_L] \sqsubseteq q_t$, for each CQ in q'_{s,M_L}, there is a homomorphism from some CQ in q_t to it. Hence, for each symbol $a \in \Sigma_s$ occurring in q_s, for each CQ in the UCQ $M_L(a)$, we need at most ℓ_{q_t} atoms for the homomorphism from q_t. To derive a bound for the number of such CQs, we observe that the inclusion $q_t \sqsubseteq q_s[M_L]$ must also hold. Therefore, we need at most m_{q_t} CQs in $M_L(a)$ to satisfy the inclusion of all the CQs in q_t. It follows that, to check for the existence of the LAV mapping M_L and the corresponding homomorphism, it suffices to guess for each $a \in \Sigma_s$ a UCQ over Σ_t consisting of at most m_{q_t} CQs, each with at most ℓ_{q_t} atoms. In the general case where the mapping M consists of several assertions $q_s \rightsquigarrow q_t$, we can proceed analogously to the case above, using instead of m_{q_t} and ℓ_{q_t}, the sum of these parameters over all mapping assertions in M. To check whether $q_t \sqsubseteq q_s[M_L]$, it suffices to check for the existence of a homomorphism from $q_s[M_L]$ to each of the CQs in q_t, which can be done in NP in the size of q_t. To check whether $q_s[M_L] \sqsubseteq q_t$, we have to check whether for each CQ q' obtained by selecting one of the CQs q'' in q_s and then substituting each atom α in q'' with one of the CQs in $\alpha[M_L]$, there is a homomorphism from some CQ in q_t to q'. We can do so by a coNP computation that makes use of an NP oracle to check for the existence of a homomorphism. This gives us the Π_2^p upper bound. ☐

Proof of Theorem 13.

The proof is by a reduction from 3-colorability. Consider a graph $G = (N, E)$, with $N = \{n_1, \ldots, n_k\}$.

Let $\Sigma_s = \{t/2, a_s/2, a_f/2\}$, $\Sigma_t = \{e/2, b_s/2, b_f/2\}$,

$$
\begin{aligned}
q_T = \{(s,f) \mid\ & a_s(s,r), a_s(s,g), a_s(s,b),\\
& t(r,g), t(g,r), t(r,b), t(b,r), t(g,b), t(b,g),\\
& a_f(r,f), a_f(g,f), a_f(b,f)\ \}\\
q_G = \{(s,f) \mid\ & b_s(s,x_1), \ldots, b_s(s,x_k),\\
& \textstyle\bigwedge_{(n_i,n_j)\in E}\{e(x_i,x_j), e(x_j,x_i)\},\\
& b_f(x_1,f), \ldots, b_f(x_k,f)\ \}
\end{aligned}
$$

and define the following mapping M:

$$
\begin{aligned}
q_T &\rightsquigarrow q_G & (7)\\
\{(x,y) \mid t(x,y)\} &\rightsquigarrow \{(x,y) \mid e(x,y)\} & (8)\\
\{(x,y) \mid a_s(x,y)\} &\rightsquigarrow \{(x,y) \mid b_s(x,y)\} & (9)\\
\{(x,y) \mid a_f(x,y)\} &\rightsquigarrow \{(x,y) \mid b_f(x,y)\} & (10)
\end{aligned}
$$

Intuitively, assertion (7) maps a triangle, whose three vertexes are connected by a_s and a_f to the distinguished variables s and f respectively, to the graph G, whose nodes are connected by b_s and b_f to the distinguished variables s and f respectively.

We show that G is 3-colorable iff MSIMP[SOUND,LAV,CQ,CQ] with input M admits a solution. For the "only-if" part, consider the LAV mapping M_L consisting of the mapping assertions (8), (9), and (10). If G is 3-colorable, a coloring of the nodes of G with the three colors r, g, b gives us immediately a homomorphism from $q_G[M_L]$ to q_T in which each variable x_i of $q_G[M_L]$ is mapped to the variable of q_T corresponding to the color assigned to node n_i. Hence we have that $q_T \sqsubseteq q_G[M_L]$. For the "if-part", consider a LAV mapping M_L such that $q_s[M_L] \sqsubseteq q_t$ for each mapping assertion $q_s \rightsquigarrow q_t$ in M. By the mapping assertions (8), (9), and (10), we have that the queries $M_L(t)$, $M_L(a_s)$, and $M_L(a_f)$, which we assume to have (x,y) as distinguished variables, must include respectively the atoms $e(x,y)$, $b_s(x,y)$, and $b_f(x,y)$, plus possibly additional atoms containing existentially quantified variables. Note that these existential variables appear in the unfolding $q_T[M_L]$. Now, consider a homomorphism h from $q_G(s,f)$ to $q_T[M_L](s,f)$. Since s and f are distinguished variables, we have that $h(s) = s$ and $h(f) = f$. Suppose that for some variable $x_i \in \{x_1, \ldots, x_k\}$ of q_G, we have that $h(x_i)$ is an existential variable y in an additional atom in $q_T[M_L]$. Then, since q_G contains the atoms $b_s(s,x_i)$ and $b_f(x_i,f)$, we must have that $q_T[M_L]$ contains the atoms $b_s(s,y)$ and $b_f(y,f)$. This is impossible, since y is an existential variable introduced by the unfolding of q_T with M_L, and hence can appear in the unfolding of just one atom of q_T. But there is no atom of q_T that contains both s and f, and that could generate both $b_s(s,y)$ and $b_f(y,f)$. So, the only possibility for a homomorphism from $q_G[M_L]$ to q_T is to map each x_i of $q_G[M_L]$ to one of the variables r, g, or b. The existence of such a homomorphism implies that G is 3-colorable. ☐

Proof of Lemma 18.

If $q_t = \varepsilon$, we can simply set $M'_L(a) = \varepsilon$, for each $a \in \Sigma_s$. Otherwise, since $q_s[M_L] \neq \emptyset$ and $q_s[M_L] \subseteq q_t$, there exists a nonempty word $a_1 \cdots a_k \in q_s$ and a word $w_1 \cdots w_k \in q_s[M_L]$ and hence in q_t, where $w_j \in M_L(a_j)$. To define the new LAV mapping M'_L, we consider each $a \in \Sigma_s$ appearing in $a_1 \cdots a_k$. Notice that a might appear in $a_1 \cdots a_k$ multiple times, and suppose the occurrences of a are $a_{i_1}, \ldots, a_{i_\ell}$, corresponding to $w_{i_1}, \ldots, w_{i_\ell}$. We chose arbitrarily one w_{i_j} and set $M'_L(a) = w_{i_j}$. Instead, for each $a \in \Sigma_s$ not appearing in $a_1 \cdots a_k$, we set $M'_L(a) = \emptyset$. Now, $q_s[M'_L] \neq \emptyset$ by construction, and since $M'_L(a) \subseteq M_L(a)$ for every $a \in \Sigma_s$, we have that $q_s[M'_L] \subseteq q_s[M_L] \subseteq q_t$. □

Proof of Lemma 19.

Let $A_t = (\Sigma_t, S_t, s_t^0, \delta_t, F_t)$. Consider a word $a_1 \cdots a_h \in q_s$. If there is one of the a_i such that $M_L(a_i) = \emptyset$, then $M_L(a_1) \cdots M_L(a_h) = \emptyset \subseteq q_t$. Otherwise, we have that $M_L(a_i) = \{w^{a_i}\}$, for $i \in \{1, \ldots, h\}$, and since $w^{a_1} \cdots w^{a_h} \in q_s[M_L] \subseteq q_t$, there is a sequence $s_0, s_1, \ldots, s_h$ of states of A_t such that $s_0 = s_t^0$, $s_h \in F_t$, and $s_i \in \delta_t(s_{i-1}, w^{a_i})$, for $i \in \{1, \ldots, h\}$. Consider now, for each $i \in \{1, \ldots, h\}$, a word $w'_i \in M'_L(a_i) = [w^{a_i}]_{A_t}$. Making use of the characterization of $[w^{a_i}]_{A_t}$ in terms of a binary relation over S_t, we have for each word in $[w^{a_i}]_{A_t}$, and in particular for w'_i, that $s_i \in \delta_t(s_{i-1}, w'_i)$. Hence, $s_h \in \delta_t(s_o^t, w'_1 \cdots w'_h)$ and $w'_1 \cdots w'_h \in q_t$. □