Schema Mappings and Data Exchange for Graph Databases

Pablo Barceló
Department of Computer Science, Universidad de Chile
pbarcelo@dcc.uchile.cl

Jorge Pérez
Department of Computer Science, Universidad de Chile
perez@dcc.uchile.cl

Juan Reutter
School of Informatics, University of Edinburgh
juan.reutter@ed.ac.uk

ABSTRACT

Data exchange and schema mapping management have received little attention so far in the graph database scenario, and tools developed in this context for relational databases have significant drawbacks in the context of graph-structured data. In this paper we embark on the study of interoperability issues for graph databases, including schema mappings, data exchange and certain answers computation.

We start by analyzing different possibilities for specifying mappings in graph databases. Our mapping languages are based on the most typical graph databases queries, ranging from regular path queries to conjunctions of nested regular expressions. They subsume all previously considered mapping languages, and let one express many data exchange scenarios in the graph database context. We study the problems of materializing solutions and query answering, in particular, the problem of computing universal representatives and certain answers for various classes of mappings. We show that both problems are difficult with respect to combined complexity, and that for the latter problem, even data complexity is high for some very simple mappings and queries. We then identify relevant classes of mappings and queries for which the problems of materializing solutions and query answering can be solved efficiently.

Categories and Subject Descriptors

H.2.1 [**Logical Design**]: *Data Models*; H.2.5 [**Heterogeneous Databases**]: *Data translation*

General Terms

Theory, Languages, Algorithms

Keywords

Graph databases, Schema Mappings, Data Exchange.

1. INTRODUCTION

Graph-structured data has become pervasive in data centric applications. Social networks, bioinformatics, astronomic databases, digital libraries, Semantic Web, and linked government data, are only a few examples of applications in which structuring data as graphs is, simply, essential [19].

But while the fundamental problem of querying graph databases has received considerable attention in the literature [34, 1, 9, 24], interoperability issues among graph-structured data sources remain almost unexplored from a theoretical point of view. Tasks such as data exchange, data integration and, more generally, schema mapping management, have so far received very little attention in the graph database context compared to its relational and XML counterparts [3]. Some remarkable exceptions can be found in the work by Calvanese et al. regarding rewriting of views for graph databases [14], and more recently the study of schema mapping simplification in the same context [16].

In the present paper we embark on the theoretical study of schema mapping specification and data exchange for graph databases, the latter being currently unexplored. Recall that schema mappings are high-level specifications that permit to define relationships between two different schemas [12, 30, 3]. Schema mappings have received considerable attention in the context of relational databases [35, 23, 21]. Regrettably, the possibility of using relational mappings tools for solving interoperability tasks is not suitable for graph databases. We delve into this issue below.

Relational mappings are typically defined in terms of conjunctive queries, and hence they lack any form of recursion which is a crucial feature for querying graph databases [1, 37]. Therefore, additional features would need to be included in this class of mappings in order to specify more complex navigational properties. However, this would imply leaving relational data exchange tools behind, since the study of relational data exchange has been carried out mostly in terms of mappings defined by non-recursive queries [8]. The problem is that even when navigational properties can sometimes be expressed in SQL or in other extensions of relational calculus, such as DATALOG, it is not clear how to define schema mappings based on those languages. Moreover, given the high complexity cost associated with performing simple static analysis tasks for them, such as equivalence or containment, using mappings based in unrestricted SQL or DATALOG may leave us without practical algorithms for even the most simple data exchange tasks.

We thus require a class of mappings that is specifically tailored for graph databases. The backbone of our mapping languages are the most common reachability queries over graph databases, ranging from *regular path queries* (RPQs) [1, 37] to conjunctions of *nested regular expressions* (NREs)

[11], a class of queries that extends RPQs with the ability to traverse edges in both directions and to perform branching while navigating the data. Our mappings express interesting exchange properties in the graph database context. Notably, they can express not only usual exchange properties based on exporting tuples of elements, but also complex navigational properties, such as exporting entire paths satisfying some regular conditions.

We apply our mappings to study data exchange in the graph database context. Recall that the data exchange problem is the following: Given a mapping $\mathcal{M}$ from a *source* schema to a *target* schema and a source database D, compute a target database that better reflects the source data in D under $\mathcal{M}$ [22, 8, 3]. Such target database is said to be a *solution* for D under $\mathcal{M}$. In relational data exchange one normally computes a *universal* solution [22], which is a database with incomplete information that represents the entire space of solutions for D under $\mathcal{M}$. In graph data exchange we face the analog problem of computing a *universal representative*, which is an incomplete graph database, with missing information both at the data and at the structural level, with the same good properties. We show that universal representatives enjoy many of the good properties of universal solutions. In particular, the usual techniques for computing universal solutions in relational data exchange can be applied to computing universal representatives in graph data exchange. The procedure works in exponential time in combined complexity (that is, assuming mappings and databases to be part of the input), and in polynomial time in data complexity (that is, assuming mappings to be fixed).

Notice that traditional data exchange analysis has been carried out in terms of data complexity (save for a few exceptions [28, 4]). The rationality behind this decision is the usual one in database theory: Mappings and queries are often much smaller than the data, and hence a meaningful complexity analysis should not locate databases and specifications at the same level. While this seems to be a reasonable assumption for regular size databases, it is no longer a valid assumption for massive applications of graph databases (such as, for example, social networks or scientific databases). For instance, the procedure mentioned above for constructing universal representatives in graph data exchange works in time $|G|^{O(|\mathcal{M}|)}$, for a source graph database G and a mapping $\mathcal{M}$, which can be considered infeasible for big graph databases, even for small $\mathcal{M}$.

It is thus important for the study of graph data exchange to identify relevant classes of mappings for which basic computational tasks can be solved efficiently, not only in data but also in combined complexity. In the paper we introduce a class of mappings, called *NRE-restricted* mappings, for which the combined complexity of computing universal representatives is tractable, and even given by low-degree polynomials. These mappings allow to define views over the source based on single NREs. Despite their simplicity we show that they allow to express important graph data exchange properties.

Another important problem in data exchange is query answering. Typically, one is interested in computing the *certain* answers of a query [30, 8, 3], that is, those answers that hold in all possible solutions. Just as in relational data exchange [22, 6], we do this in a two-step fashion: First, we compute a universal representative, and then we evaluate the query over the representative. We show that the problem of computing certain answers in graph data exchange is inherently difficult in combined complexity, and that even in data complexity (that is, assuming queries and mappings to be fixed) we easily face intractability. This motivates the search for relevant classes of mappings and queries for which the problem of computing certain answers is tractable. We study this problem in terms of both combined and data complexity.

We start by showing that we need to further restrict the class of NRE-restricted mappings in order to obtain efficient query answering in combined complexity for queries defined by NREs. This restriction defines a new class of *rigid* NRE-restricted mappings, that disallows the use of disjunction and recursion for views over the target. The complexity of query evalution for this class is quadratic in the size of the data. We also show that the certain answers of NREs can be computed in linear time, if one restricts to mappings that can only define symbols over the target, the so-called NRE-GAV mappings. These good properties follow from query *rewriting* [30] results that are of independent interest. In particular, we use those results to show that the class of NRE-GAV mappings has good properties also in terms of *composition*, a typical schema mapping management task that has received considerable attention in the literature for relational and XML databases [31, 33, 23, 35, 7, 2].

Finally, we study the data complexity of the problem of computing certain answers. We start by showing that for mappings that disallow the use of disjunction and recursion when defining views over the target, we can compute certain answers of a very general class of queries in polynomial time. For mappings that define arbitrary views over the target the situation is more complicated, and we need to find structural restrictions not only on mappings and queries, but also on the interaction between universal representatives and queries, in order to obtain a tractable class. We also show that our restrictions are, in a sense, optimal, since lifting any one of them leads to intractability.

Organization. The rest of the paper is organized as follows. In Section 2 we define graph databases, and present a quick summary of usual query languages for graph databases. In Section 3 we introduce our class of graph mappings and show that they can express interesting navigational properties. In Section 4 we apply those mappings to graph data exchange, and study the problems of computing universal representatives and certain answers. Afterwards, in Sections 5 and 6, we present classes of mappings and queries for which universal representatives and certain answers can be computed efficiently in combined complexity. We take a brief detour in Section 7, and quickly analyze the good properties of a particular class of mappings with respect to composition. Finally, in Section 8 we study the problem of efficient query answering in data exchange in terms of data complexity, and in Section 9 we establish our concluding remarks.

2. GRAPH DATABASES AND QUERIES

Graph databases. Let $\mathbf{V}$ be a countably infinite set of *node ids*, and Σ a finite alphabet. A *graph database* G over Σ is a pair (V, E), where V is a finite set of node ids (that is V is a finite subset of $\mathbf{V}$) and $E \subseteq V \times \Sigma \times V$. That is, G is an *edge-labeled* directed graph, where the fact that (u, a, v) belongs to E means that there is an edge from node

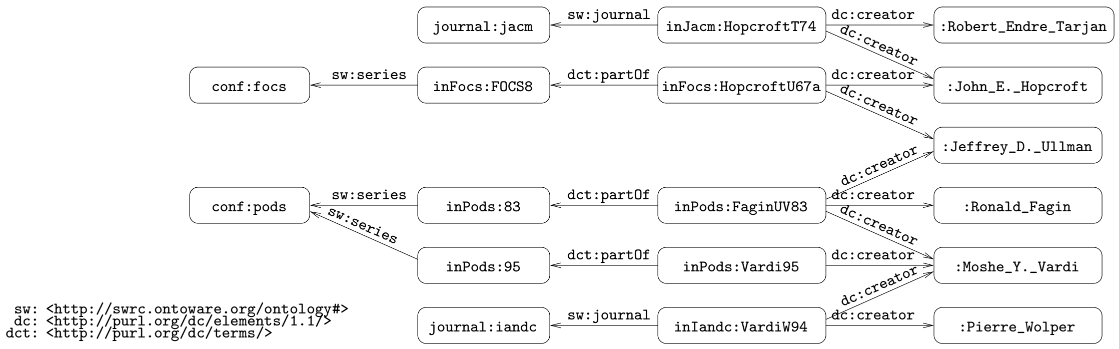


Figure 1: A fragment of the RDF Linked Data representation of DBLP [20] available at `http://dblp.l3s.de/d2r/`

u into node v labeled a. For a graph database $G = (V, E)$, we write $(u, a, v) \in G$ whenever $(u, a, v) \in E$.

Regular path queries. Queries over graph databases are typically navigational, in the sense that they allow to traverse the edges of the graph while checking for the existence of paths satisfying certain conditions [37]. For instance, one of the most basic querying mechanisms for graph databases is the class of *regular path queries* (RPQs) [34, 1, 37], that retrieve all pairs of nodes in a graph database that are linked by a path labeled with a string satisfying some regular expression. Formally, given a finite alphabet Σ, the class of RPQs r over Σ is defined by the grammar of regular expressions as follows:

$$r \quad := \quad \varepsilon \quad | \quad a \ (a \in \Sigma) \quad | \quad r + r \quad | \quad r \cdot r \quad | \quad r^*$$

We formalize the semantics of an RPQ r over a graph database G as a binary relation $[\![r]\!]_G$ defined as follows, where a is a symbol in Σ, r, r_1 and r_2 are arbitrary RPQs over Σ, and G is a graph database over Σ:

$$\begin{aligned}
[\![\varepsilon]\!]_G &= \{(u, u) \mid u \text{ is a node id in } G\} \\
[\![a]\!]_G &= \{(u, v) \mid (u, a, v) \in G\} \\
[\![r_1 + r_2]\!]_G &= [\![r_1]\!]_G \cup [\![r_2]\!]_G \\
[\![r_1 \cdot r_2]\!]_G &= [\![r_1]\!]_G \circ [\![r_2]\!]_G \\
[\![r^*]\!]_G &= [\![\varepsilon]\!]_G \cup [\![r]\!]_G \cup [\![r \cdot r]\!]_G \cup [\![r \cdot r \cdot r]\!]_G \cup \cdots
\end{aligned}$$

Here, the symbol $\circ$ denotes the usual composition of binary relations, that is, $[\![r_1]\!]_G \circ [\![r_2]\!]_G = \{(u, v) \mid \text{there exists } w \text{ s.t. } (u, w) \in [\![r_1]\!]_G \text{ and } (w, v) \in [\![r_2]\!]_G\}$. As it is customary, we use r^+ as a shortcut for $r \cdot r^*$.

RPQs are often enhanced with the ability to traverse edges *backwards*, i.e. with the atomic formula a^-, for each $a \in \Sigma$, such that $(u, v) \in [\![a^-]\!]_G$ iff $(v, a, u) \in G$, for every graph database G over Σ. The resulting class of queries is known as *two-way* RPQs, or simply 2RPQs [14].

Example 2.1. Let G be the graph database in Figure 1. This graph contains a fragment of the *RDF Linked Data* representation of DBLP [20]. The following is a simple 2RPQ that matches all pairs (x, y) such that x is an author that published a paper in conference y (for simplicity, we omit prefixes `dc:`, `dct:`, and `sw:` in edge labels):

$$r_1 = \texttt{creator}^- \cdot \texttt{partOf} \cdot \texttt{series}$$

For example, the pairs $(\texttt{:Jeffrey\_D.\_Ullman}, \texttt{conf:focs})$ and $(\texttt{:Ronald\_Fagin}, \texttt{conf:pods})$ are in $[\![r_1]\!]_G$. $\square$

Nested regular expressions. It is well-known that the expressive power of 2RPQs is limited in several ways. An obvious limitation is that 2RPQs allow only for *linear* navigation of the graph database, i.e. they are not capable of *branching* while traversing the data. In order to overcome this limitation, 2RPQs have been extended with an *existential test* operator $[(\cdot)]$, also known as *nesting* operator, *a la* XPath [25]. The extension gives rise to the class of *nested regular expressions* (NREs) [36, 11]. Formally, NREs extend 2RPQs with expressions of the form $[exp]$, for exp an NRE, such that for each graph database G it is the case that $[\![[exp]]\!]_G = \{(u, u) \mid (u, v) \in [\![exp]\!]_G, \text{ for some } v \in G\}$.

Example 2.2. The following NRE matches, in the graph database G shown in Figure 1, all pairs (x, y) such that x and y are connected by a coauthorship sequence that only considers conference papers:

$$exp = \left(\texttt{creator}^- \cdot \left[\texttt{partOf} \cdot \texttt{series}\right] \cdot \texttt{creator}\right)^+$$

Let us give the intuition of the evaluation of this expression. Assume that we start at node u. The (inverse) edge $\texttt{creator}^-$ forces us to navigate from u to a paper v created by u. Then the existential test $[\texttt{partOf} \cdot \texttt{series}]$ is used to check that from v we can navigate to a conference (and thus, v is a conference paper). Finally, we follow edge `creator` from v to an author w of v. The $(\cdot)^+$ over the expression allows us to repeat this sequence several times. For instance, $(\texttt{:John\_E.\_Hopcroft}, \texttt{:Moshe\_Y.\_Vardi})$ is in $[\![exp]\!]_G$, but $(\texttt{:John\_E.\_Hopcroft}, \texttt{:Pierre\_Wolper})$ is not in $[\![exp]\!]_G$. It can be proved that the use of nesting is essential for expressing this query. $\square$

Conjunctive 2RPQs. A second limitation of the class of 2RPQs is that it is not closed under conjunction and projection, and thus it is not able to accommodate the expressive power of usual conjunctive queries. This lead to the introduction of *conjunctive* 2RPQs [18, 1, 13, 37], defined as follows. Let $\bar{x} = (x_1, \ldots, x_n)$ and $\bar{y} = (y_1, \ldots, y_m)$ be (possibly empty) tuples of distinct variables. We denote by $\bar{x} \cup \bar{y}$ the set $\{x_1, \ldots, x_n, y_1, \ldots, y_m\}$. A conjunctive 2RPQ (C2RPQ), with free variables $\bar{x}$, is a formula $\varphi(\bar{x})$ of the form

$$\exists \bar{y} \Big((z_1, r_1, z_1') \wedge (z_2, r_2, z_2') \wedge \cdots \wedge (z_k, r_k, z_k') \Big) \qquad (1)$$

where r_i is a 2RPQ for every $1 \leq i \leq k$, $z_1, z_1', \ldots z_k, z_k'$ are not necessarily distinct variables, and $\{z_1, z_1', \ldots, z_k, z_k'\} = \bar{x} \cup \bar{y}$. The arity of a C2RPQ is the number of free variables in the formula (n in this case). A CRPQ is a C2RPQ of the form (1), in which each r_i is an RPQ.

Given a tuple $\bar{c} = (c_1, \ldots, c_n)$ and a graph database G, we say that G satisfies $\varphi(\bar{c})$, denoted by $G \models \varphi(\bar{c})$, if there exists a mapping h from $\bar{x} \cup \bar{y}$ to the node ids in G such that $h(x_j) = c_j$ for every $1 \leq j \leq n$, and $(h(z_i), h(z'_i)) \in [\![r_i]\!]_G$ for every $1 \leq i \leq k$. The evaluation of $\varphi(\bar{x})$ over G, denoted by $[\![\varphi(x)]\!]_G$, is the set of tuples $\{\bar{c} \mid G \models \varphi(\bar{c})\}$.

Example 2.3. Consider the binary C2RPQ

$$\varphi(x,y) \;=\; \exists u, v, w\big((x, \texttt{creator}^-, u) \wedge (u, \texttt{partOf}, v) \wedge (v, \texttt{series}, w) \wedge (u, \texttt{creator}, y)\big).$$

It is not hard to see that $\varphi(x, y)$ is *equivalent* to the NRE

$$\mathit{exp} \;=\; \texttt{creator}^- \cdot [\,\texttt{partOf} \cdot \texttt{series}] \cdot \texttt{creator}.$$

This means that for each graph database G over alphabet $\Sigma = \{\texttt{creator}, \texttt{partOf}, \texttt{series}\}$, it is the case that $[\![\varphi(x,y)]\!]_G = [\![\mathit{exp}]\!]_G$. □

Notice that by replacing the role of 2RPQs with NREs in the definition of C2RPQs, we obtain a new class of *conjunctive* NREs (CNREs), that will take a prominent role in our definition of mappings for graph databases. Formally, a CNRE is a formula of the form (1), in which each expression r_i is an NRE, for $1 \leq i \leq k$. The semantics of this CNRE is defined exactly as the semantics of a C2RPQ of the same form, with the slight difference that now we ask each pair $(h(z_i), h(z'_i))$ to belong to the interpretation of the NRE r_i in G. Despite its naturalness, CNREs have been completely overlooked in the literature until now, and we present them here for the first time.

Two queries $Q_1(\bar{x})$ and $Q_2(\bar{x})$ over the same alphabet Σ are *equivalent*, if $[\![Q_1(\bar{x})]\!]_G = [\![Q_2(\bar{x})]\!]_G$ for each graph database G over Σ. Sometimes we abuse notation, and identify the NRE *exp* with the CNRE $Q(x, y) = (x, \mathit{exp}, y)$. Notice that the two queries are equivalent.

Expressiveness and complexity. Interestingly, C2RPQs and NREs are incomparable in terms of its expressive power. That is, C2RPQs and NREs correspond to two orthogonal ways of extending the class of 2RPQs [11]. In particular, the NRE $\mathit{exp} = (\texttt{creator}^- \cdot [\,\texttt{partOf} \cdot \texttt{series}] \cdot \texttt{creator})^+$ in Example 2.2 is not equivalent to a C2RPQ. This immediately implies that the class of CNREs properly extends the class of C2RPQs.

With respect to complexity of query evaluation, NREs are not only *polynomial* in combined complexity (i.e. when both the database and the query are given as input), but they can be evaluated linearly in both the size of the database and the expression. Given a graph database G and an NRE *exp*, we use $|G|$ to denote the size of G (in terms of the number of egdes $(u, a, v) \in G$), and $|\mathit{exp}|$ to denote the size of *exp*.

Proposition 2.4. *[36] Checking, given a graph database G, a pair of nodes (u, v), and an NRE exp, whether $(u, v) \in [\![\mathit{exp}]\!]_G$, can be done in time $O(|G| \cdot |\mathit{exp}|)$.*

On the other hand, query evaluation for C2RPQs is NP-complete in combined complexity [17, 9]. In terms of *data* complexity, that is, assuming the query to be fixed, query evaluation for C2RPQs can be solved in NLOGSPACE [18]. It is not hard to prove that exactly the same complexity bounds hold for the CNREs.

3. SCHEMA MAPPINGS FOR GRAPH DATA

Schema mappings have been studied both in the relational [22] and the XML [5] scenario (see [3], for a recent general presentation of the area). In a very high level way, schema mappings are tuples of the form $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \mathcal{T})$, where $\mathbf{S}_1$ and $\mathbf{S}_2$ are appropriate schemas, and $\mathcal{T}$ is a finite set of rules of the form

$$\phi_{s_1}(\bar{x}) \;\rightarrow\; \psi_{s_2}(\bar{x}), \tag{2}$$

with $\phi_{s_1}(\bar{x})$ and $\psi_{s_2}(\bar{x})$ logical formulas over $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively, that specify the relationship between the two schemas.

Traditional rule specification has been carried out assuming that ϕ_{s_1} and ψ_{s_2} are suitable conjunctive queries for the data model at hand. This is because this class of rules allows to express how the schema $\mathbf{S}_2$ is defined in terms of the existence of certain patterns over the schema $\mathbf{S}_1$. For instance, in the relational case both ϕ_{s_1} and ψ_{s_2} correspond to usual conjunctive queries [22], while in the XML case they correspond to *tree pattern queries* [5], which are essentially acyclic conjunctive queries over XML trees including recursion at the atomic level.

In the same spirit, it seems completely natural to define mappings for graph databases by allowing rules of the form $\phi_{s_1}(\bar{x}) \rightarrow \psi_{s_2}(\bar{x})$, with $\phi_{s_1}(\bar{x})$ and $\psi_{s_2}(\bar{x})$ suitable conjunctive queries over graph alphabets Σ_1 and Σ_2, respectively. Among all the query languages introduced in Section 2, we have chosen the most general one, CNREs, as the default for specifying mappings. However, in the paper we consider all possibilities, from the most expressive CNREs to the least expressive RPQs.

Definition 3.1 (graph mapping). *Let Σ_1 and Σ_2 be finite alphabets. A graph mapping $\mathcal{M}$ (or, simply, mapping, from now on) from Σ_1 to Σ_2 is a tuple $(\Sigma_1, \Sigma_2, \mathcal{T})$, where $\mathcal{T}$ is a finite set of rules of the form $\phi_{s_1}(\bar{x}) \rightarrow \psi_{s_2}(\bar{x})$, with $\phi_{s_1}(\bar{x})$ and $\psi_{s_2}(\bar{x})$ CNREs over Σ_1 and Σ_2, respectively.*

If for each rule in $\mathcal{M}$ both ϕ_{s_1} and ψ_{s_2} are specified in some proper subclass $\mathcal{L}$ of the CNREs, such as the class of C2RPQs, CRPQs, NREs, 2RPQs, or RPQs, we say that $\mathcal{M}$ is an $\mathcal{L}$-mapping.

Example 3.2. Consider again the alphabet $\Sigma_1 = \{\texttt{creator}, \texttt{partOf}, \texttt{series}\}$, over which the graph database in Figure 1 is defined. Let Σ_2 be the alphabet $\{\texttt{makes}, \texttt{inConf}\}$. The mapping $\mathcal{M}_{12} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$, with $\mathcal{T}_{12}$ consisting of

$$(x, \texttt{creator}, y) \wedge (x, \texttt{partOf} \cdot \texttt{series}, w) \;\rightarrow\; (y, \texttt{makes}, x) \wedge (x, \texttt{inConf}, w),$$

is a CRPQ-mapping from Σ_1 to Σ_2. Consider now the alphabet $\Sigma_3 = \{\texttt{confConnected}\}$, and $\mathcal{T}_{23}$ defined by the rule

$$\big(x, (\,\texttt{makes} \cdot \texttt{makes}^-)^+, y\big) \;\;\rightarrow\;\; (x, \texttt{confConnected}, y).$$

Then $\mathcal{M}_{23} = (\Sigma_2, \Sigma_3, \mathcal{T}_{23})$ is a 2RPQ-mapping, this time from Σ_2 to Σ_3. Finally, $\mathcal{M}_{13} = (\Sigma_1, \Sigma_3, \mathcal{T}_{13})$, where $\mathcal{T}_{13}$ is given by the single rule

$$\big(x, (\texttt{creator}^- \cdot [\,\texttt{partOf} \cdot \texttt{series}] \cdot \texttt{creator})^+, y\big) \rightarrow (x, \texttt{confConnected}, y),$$

is an NRE-mapping from Σ_1 to Σ_3. □

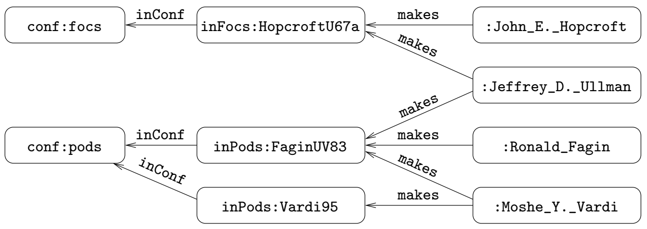


Figure 2: Result of exchanging the graph in Figure 1 under mapping $\mathcal{M}_{12}$

C2RPQ-mappings have been studied in contexts different from data exchange, such as query answering using views [15], and, more recently, in terms of its optimization properties [16], but so far none of these classes has been studied from a data exchange point of view. It is worth remarking that the use of (C)NREs in mappings is important, since (C)NRE-based mappings let one express relevant graph data exchange properties that mappings based on C2RPQs cannot express (see e.g. Example 3.3). Moreover, the results in the paper show that the increase in expressive power comes at no computational cost.

Solutions. If G_1 and G_2 are graph databases over Σ_1 and Σ_2, respectively, then the pair (G_1, G_2) satisfies the mapping $\mathcal{M}$, denoted $(G_1, G_2) \models \mathcal{M}$, if the following holds for each rule in $\mathcal{T}$ of the form $\phi_{s_1}(\bar{x}) \to \psi_{s_2}(\bar{x})$ and each tuple $\bar{u}$ of node ids in G_1 such that $|\bar{u}| = |\bar{x}|$:

$$\bar{u} \in [\![\phi_{s_1}(\bar{x})]\!]_{G_1} \implies \bar{u} \in [\![\psi_{s_2}(\bar{x})]\!]_{G_2}. \qquad (3)$$

Recall that $\phi_{s_1}(\bar{x})$ and $\psi_{s_2}(\bar{x})$ are CNREs, and hence they are of the form $\exists\bar{y}\alpha(\bar{x},\bar{y})$ and $\exists\bar{z}\beta(\bar{x},\bar{z})$, where $\alpha(\bar{x},\bar{y})$ and $\beta(\bar{x},\bar{z})$ are conjunctions of NREs over Σ_1 and Σ_2, respectively. Therefore, statement (3) means that whenever $(\bar{u},\bar{v}) \in [\![\alpha(\bar{x},\bar{y})]\!]_{G_1}$, for some tuple $\bar{v}$ of nodes in G_1 such that $|\bar{v}| = |\bar{y}|$, it is also the case that there is a tuple $\bar{w}$ of nodes in G_2 such that $|\bar{w}| = |\bar{z}|$ and $(\bar{u},\bar{w}) \in [\![\beta(\bar{x},\bar{z})]\!]_{G_2}$.

Following the usual data exchange terminology, we say that G_2 is a *solution* for G_1 under $\mathcal{M}$ (or simply a solution, if $\mathcal{M}$ is clear from the context) whenever $(G_1, G_2) \models \mathcal{M}$. The set of solutions for G_1 under $\mathcal{M}$, denoted $\mathrm{Sol}_{\mathcal{M}}(G_1)$, is $\{G_2 \mid (G_1, G_2) \models \mathcal{M}\}$. Finally, the semantics $[\![\mathcal{M}]\!]$ of mapping $\mathcal{M}$ is the set $\{(G_1, G_2) \mid (G_1, G_2) \models \mathcal{M}\}$. Two mappings $\mathcal{M}$ and $\mathcal{M}'$ are *equivalent* if $[\![\mathcal{M}]\!] = [\![\mathcal{M}']\!]$.

EXAMPLE 3.3. Consider again the mappings $\mathcal{M}_{12}$, $\mathcal{M}_{23}$ and $\mathcal{M}_{13}$ in Example 3.2. The graph database G_1 shown in Figure 2 is a solution for the graph database G shown in Figure 1 under $\mathcal{M}_{12}$, and the graph database G_2 shown in Figure 3 is a solution for G_1 under $\mathcal{M}_{23}$.

Solutions for G under $\mathcal{M}_{13}$ are graph databases that contain all triples of the form $(u, \texttt{confConnected}, v)$, such that (u, v) are in the coauthorship sequence of conference papers defined by NRE $(\texttt{creator}^- \cdot [\texttt{partOf} \cdot \texttt{series}] \cdot \texttt{creator})^+$ in G. Mapping $\mathcal{M}_{13}$ is not equivalent to a C2RPQ-mapping, which shows that CNRE-mappings can express interesting properties beyond the class of C2RPQ-mappings. □

In the following section we present an illustrative example of the kind of properties graph mappings can express.

3.1 Navigational exchange properties

The mappings we study in the present paper are designed for exchanging *tuples* of node ids from source to target. But

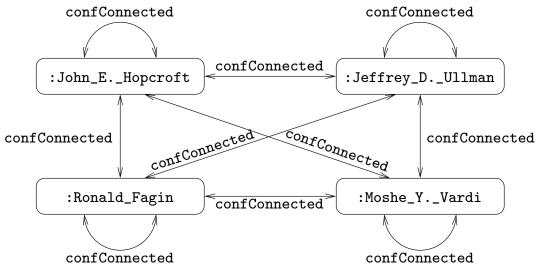


Figure 3: Result of exchanging the graph in Figure 2 under mapping $\mathcal{M}_{23}$.

what about more sophisticated *navigational* exchange tasks, that are relevant in the graph database context, such as exporting entire paths of data satisfying certain conditions on the source side? We show that graph mappings can express some interesting exchange properties of this type. We do it by means of an example, but the techniques that we use can be generalized to show that graph mappings are capable of expressing a broad class of exchange properties based on the idea of exporting entire *paths* in the source graph that satisfy certain regular conditions.

Assume that we have source and target alphabets $\Sigma_{\mathbf{S}} = \{a, b, c, d\}$ and $\Sigma_{\mathbf{T}} = \{a', b', c', d'\}$, respectively. We wish to exchange data according to the following intuitive rule: Copy each path from the source to the target that starts and ends with an edge labeled c, and has at least two consecutive edges labeled a or at least one edge labeled b. Clearly, the RPQ $r = c \cdot \Sigma^* \cdot (aa + b) \cdot \Sigma^* \cdot c$ extracts from the source the pairs of nodes that are linked by a path satisfying the regular condition mentioned above. But how can we express our desired copying rule as a graph mapping?

We start by posing the following question: Under which circumstances do we have to copy an edge labeled a from the source as an edge labeled a' in the target, while navigating the source data? This is the case if one of the following holds:

1. Two consecutive edges labeled a, or one edge labeled b, is yet to appear. That is, we have read $exp_1^1 = c \cdot \Sigma^*$ and have yet to read $exp_2^1 = \Sigma^* \cdot (aa + b) \cdot \Sigma^* \cdot c$.

2. We are reading the first of the consecutive edges labeled a. That is, we have read $exp_1^2 = c \cdot \Sigma^*$ and have to read $exp_2^2 = a \cdot \Sigma^* \cdot c$.

3. We are reading the second of the two consecutive edges labeled a. That is, we have read $exp_1^3 = c \cdot \Sigma^* \cdot a$ and need to read $exp_2^3 = \Sigma^* \cdot c$.

4. We already read two consecutive edges labeled a or an edge labeled b, but are waiting for the final edge labeled c. In this case we have read $exp_1^4 = c \cdot \Sigma^* \cdot (aa + b) \cdot \Sigma^*$ and are waiting to read $exp_2^4 = \Sigma^* \cdot c$.

The pairs (exp_1^i, exp_2^i), for $1 \le i \le 4$, have the following property: A word of the form $w_1 \cdot a \cdot w_2$, for $w_1, w_2 \in (\Sigma_{\mathbf{S}})^*$, belongs to the language defined by r if and only if it belongs to $L(exp_1^i) \cdot a \cdot L(exp_2^i)$, for some $1 \le i \le 4$. We say then that the set $\{(exp_1^i, exp_2^i) \mid 1 \le i \le 4\}$ is a *remnant* of r with respect to a. This remnant allows us to create the rules that will copy the edges labeled a as edges labeled a', precisely when it is needed. In the same way we can define remnants of r with respect to b, c and d, respectively.

Then the mapping that defines our desired copying rule consists of the rules:

$$\exists z \exists w \big((z,\ exp_1^i, x) \wedge (x, a, y) \wedge (y,\ exp_2^i, w) \big) \rightarrow (x, a', y),$$

for $1 \le i \le 4$, together with similar rules for the remnants of b, c and d. The intuition behind this is clear, and follows from the explanation above.

Interestingly, we can express this mapping without using conjunctions in the left-hand side of rules, by taking advantage of the power of NREs. In order to define the rules we need the notion of *inverse* of an NRE *exp*, which is an NRE $(exp)^{-1}$ such that for each graph database G and pair of node ids (u, v) in G it is the case that $(u, v) \in [\![exp]\!]_G$ if and only if $(v, u) \in [\![(exp)^{-1}]\!]_G$. It is easy to prove that the class of NREs is closed under inverse, and hence that $(exp)^{-1}$ is well-defined. We can now express our mapping with rules:

$$(x,\ [(exp_1^i)^{-1}] \cdot a \cdot [exp_2^i]\,, y) \rightarrow (x, a', y), \qquad \text{for } 1 \le i \le 4,$$

together with similar rules for the remnants of b, c and d. Notice that allowing NREs in mappings is crucial for expressing this kind of rules without conjunction, as we do when studying more restricted classes of mappings in the rest of the paper.

4. GRAPH DATA EXCHANGE

Data exchange is one of the main applications of schema mappings [22, 27, 8, 3]. In this section we study *graph data exchange* under the mappings we defined in the previous section, that is, we use graph mappings for specifying how to translate graph data from a source into a target schema. More precisely, assume we have a mapping $\mathcal{M}$ from a source alphabet $\Sigma_{\mathbf{S}}$ to a target alphabet $\Sigma_{\mathbf{T}}$, and a source graph database $G_{\mathbf{S}}$. The data exchange problem consists in computing a target graph database $G_{\mathbf{T}}$ that is a solution for $G_{\mathbf{S}}$ under $\mathcal{M}$ (i.e. $G_{\mathbf{T}} \in \text{Sol}_{\mathcal{M}}(G_{\mathbf{S}})$).

4.1 Universal representatives

Given the semantics of mappings, we know that there are infinitely many solutions for a given graph database $G_{\mathbf{S}}$. This phenomenon also occurs in relational and XML data exchange [22, 5]. Thus, in data exchange one usually wants to compute a "universal representative" [22, 3, 6], which is (in very broad terms) a finite representation of the set of all solutions. In this section we show how a universal representative can be computed for each source graph database and mapping. This universal representative will turn out to be crucial also for answering queries in graph data exchange.

As it is customary in data exchange, we use databases with "incomplete" information as universal representatives [22, 3]. In particular, in order to represent the set of solutions under a graph mapping, we need graph databases that combine missing information both at the data level (missing node ids) and at the structural level (missing the precise relationship between nodes). Objects that combine these two types of incompleteness are called *graph patterns* [10]. Next we define the patterns used in our context.

Assume $\mathbf{N}$ is a countably infinite set of *labeled null values.* A graph pattern π over a finite alphabet Σ is a pair (N, D), where N is a finite set of node ids and null values (that is, $N \subseteq \mathbf{V} \cup \mathbf{N}$), and $D \subseteq N \times \text{NRE}(\Sigma) \times N$, where $\text{NRE}(\Sigma)$ denotes the set of all NREs over Σ. That is, a graph pattern is essentially a graph database whose edges are labeled by NREs (which represents the idea that the precise relationship between the terminal nodes of an edge labeled by an NRE has been lost), and that admits some node ids to be missing and replaced by unknown (null) values.

As usual, we define the semantics of graph patterns in terms of homomorphisms. Let $\pi = (N, D)$ be a graph pattern over Σ. A *homomomorphism* from π into the graph database $G = (V, E)$ is a mapping $h : N \to V$ such that: (i) h is the identity over $N \cap \mathbf{V}$ (i.e. over the node ids mentioned in N), and (ii) for each edge $(u, exp, v) \in D$ $(u, v \in N$, $exp \in \text{NRE}(\Sigma))$, it is the case that $(h(u), h(v)) \in [\![exp]\!]_G$. Whenever there is a homomorphism from π to G, we write $\pi \to G$. The set of all graph databases represented by π over Σ, denoted by $\text{Rep}_\Sigma(\pi)$, is defined as $\text{Rep}_\Sigma(\pi) = \{G \mid G$ is a graph database over Σ, and $\pi \to G\}$. If Σ is clear from the context we simply write $\text{Rep}(\pi)$.

We are now ready to formalize the notion of a universal representative for graph data exchange.

Definition 4.1 (Universal representative). *Let $\mathcal{M} = (\Sigma_{\mathbf{S}}, \Sigma_{\mathbf{T}}, \mathcal{T})$ be a mapping and $G_{\mathbf{S}}$ a graph database over $\Sigma_{\mathbf{S}}$. A graph pattern $\pi_{\mathbf{T}}$ is a* universal representative *of $G_{\mathbf{S}}$ under $\mathcal{M}$, if $\text{Sol}_{\mathcal{M}}(G_{\mathbf{S}}) = \text{Rep}_{\Sigma_{\mathbf{T}}}(\pi_{\mathbf{T}})$.*

Example 4.2. [Example 3.3 cont.] The graph database G_1 shown in Figure 2 is a a universal representative of the graph database G shown in Figure 1 under $\mathcal{M}_{12}$. The graph database G_2 shown in Figure 3 is a universal representative of G_1 under $\mathcal{M}_{23}$. □

Universal representative computation. The standard techniques for constructing universal representatives in relational data exchange are based on the *chase* [22]. Those techniques can be adapted in a very simple way to design a procedure that constructs universal representatives also in the graph database context. Such procedure works in polynomial space (and, thus, in single exponential time) when the input consists of a mapping and a source graph database (that is, in combined complexity). It works in nondeterministic logarithmic space (and, thus, in polynomial time) when the mapping is fixed and the input consists of the source graph database only (thus, in data complexity).

Proposition 4.3. *There is a procedure that, given a mapping $\mathcal{M} = (\Sigma_{\mathbf{S}}, \Sigma_{\mathbf{T}}, \mathcal{T})$ and a graph database $G_{\mathbf{S}}$ over $\Sigma_{\mathbf{S}}$, computes a graph pattern $\pi_{\mathbf{T}}$ that is a universal representative of $G_{\mathbf{S}}$ under $\mathcal{M}$ in* PSPACE. *The procedure works in* NLOGSPACE *if the mapping $\mathcal{M}$ is assumed to be fixed.*

Traditional data exchange analysis has been carried out in terms of data complexity (save for a few exceptions [28, 4]). But as we mentioned in the Introduction, this analysis is no longer appropriate for graph data exchange, due to the vast volumes of data stored by graph data applications. For instance, it is not difficult to construct a family of mappings $\mathcal{M}_n$ and source graph databases G_n of size $O(n)$, such that any universal representative of G_n under $\mathcal{M}_n$ is of size comparable to $|G_n|^{|\mathcal{M}_n|}$. Computing this representative is prohibitively expensive for big source databases, even for small mappings. Furthermore, the problem remains computationally hard in combined complexity even when this exponential blowup can be avoided. Recall that $\text{FP}^{\text{NP[log]}}$ is the class of functions that can be computed in polynomial time using a logarithmic number of calls to an NP oracle

[29]. The proof of the following theorem is by a reduction from the Chromatic Number problem.

Proposition 4.4. *The problem of computing a universal representative for a graph database $G_{\mathbf{S}}$ under a mapping $\mathcal{M} = (\Sigma_{\mathbf{S}}, \Sigma_{\mathbf{T}}, \mathcal{T})$ is $\mathrm{FP}^{\mathrm{NP[log]}}$-hard, even if restricted to inputs $G_{\mathbf{S}}$ and $\mathcal{M}$ such that there is a universal representative of $G_{\mathbf{S}}$ under $\mathcal{M}$ of size $p(|G_{\mathbf{S}}|)$, for a fixed polynomial p.*

It is thus crucial for the development of graph data exchange tools that aim to be applicable, to identify relevant classes of mappings that allow for efficient universal representative computation in combined complexity. We deal with this important issue in Section 5. Those results are also used in Section 6 to achieve efficient query answering in graph data exchange.

4.2 Query Answering

One of the main tasks in data exchange is answering queries posed over the target schema. In data exchange one is typically interested in computing the *certain* answers of queries [22, 5, 3]. Intuitively, these are the answers that hold regardless of the solution one chooses to materialize. In formal terms, given a mapping $\mathcal{M} = (\Sigma_{\mathbf{S}}, \Sigma_{\mathbf{T}}, \mathcal{T})$, a graph database $G_{\mathbf{S}}$ over $\Sigma_{\mathbf{S}}$, and a query Q over $\Sigma_{\mathbf{T}}$, we define the certain answers of Q with respect to $G_{\mathbf{S}}$ under $\mathcal{M}$, denoted by $\textsc{certain}_{\mathcal{M}}(Q, G_{\mathbf{S}})$, as the set $\bigcap\{[\![Q]\!]_{G_{\mathbf{T}}} \mid G_{\mathbf{T}} \in \mathrm{Sol}_{\mathcal{M}}(G_{\mathbf{S}})\}$, where $[\![Q]\!]_{G_{\mathbf{T}}}$ denotes the evaluation of Q over $G_{\mathbf{T}}$. We are thus interested in the following problem:

Problem:	CertainAnswers
Input:	Mapping $\mathcal{M}$ from $\Sigma_{\mathbf{S}}$ to $\Sigma_{\mathbf{T}}$, graph database $G_{\mathbf{S}}$ over $\Sigma_{\mathbf{S}}$, n-ary query Q over $\Sigma_{\mathbf{T}}$, and n-ary tuple $\bar{v} \in \mathbf{V}^n$.
Question:	Is $\bar{v} \in \textsc{certain}_{\mathcal{M}}(Q, G_{\mathbf{S}})$?

Notice that the source graph database, the mapping and the query are part of the input, and thus, we are considering the combined complexity of the problem.

The next theorem shows that the problem of computing certain answers in graph data exchange is inherently difficult, even for simple mappings and queries:

Theorem 4.5.

1. CertainAnswers *is* EXPSPACE*-complete for queries defined by CNREs. It remains hard even for CRPQs over CRPQ-mappings.*
2. CertainAnswers *is* PSPACE*-complete if restricted to queries given by RPQs over RPQ-mappings.*

The lower bound of the first part of Theorem 4.5 follows from complexity results on the containment of CRPQs [13]. The second part can be easily obtained from the fact that containment of regular expressions is PSPACE-complete. The upper bound of the first part is a new result. It shows that allowing mappings to be defined in terms of CNREs, instead of C2RPQs, increases the expressive power but not the computational cost of query answering.

From Theorem 4.5 we draw the conclusion that it is very important to identify relevant classes of mappings and queries that allow for efficient computation of certain answers in combined complexity. This is, as expected, aligned with the problem of efficient computation of universal representatives in combined complexity, mentioned at the end of Section 4.1. We deal with this issue in Section 6.

Data complexity. Although data complexity is not the main object of study of the present paper, for the reasons we explained at the end of Section 4.1, studying the data complexity of query answering in graph data exchange is still an interesting problem. For the data complexity analysis of the problem of computing certain answers, we assume queries and mappings to be fixed. That is, given a mapping $\mathcal{M}$ from $\Sigma_{\mathbf{S}}$ to $\Sigma_{\mathbf{T}}$ and an n-ary query Q over $\Sigma_{\mathbf{T}}$, we study the problem:

Problem:	CertainAnswers($\mathcal{M}, Q$)
Input:	Graph database $G_{\mathbf{S}}$ over $\Sigma_{\mathbf{S}}$ and n-ary tuple $\bar{v} \in \mathbf{V}^n$.
Question:	Is $\bar{v} \in \textsc{certain}_{\mathcal{M}}(Q, G_{\mathbf{S}})$?

It can be proved that the data complexity of the certain answers problem is intractable, even for very restricted mappings and queries, and that the use of NREs in mappings and queries does not increase the data complexity of the problem:

Theorem 4.6.

1. CertainAnswers*($\mathcal{M}, Q$) is in* coNP*, for every mapping $\mathcal{M} = (\Sigma_{\mathbf{S}}, \Sigma_{\mathbf{T}}, \mathcal{T})$ and CNRE Q over $\Sigma_{\mathbf{T}}$.*
2. *There is an RPQ-mapping $\mathcal{M}$ from $\Sigma_{\mathbf{S}}$ to $\Sigma_{\mathbf{T}}$ and an RPQ Q of the form w, for $w \in (\Sigma_{\mathbf{T}})^*$, such that the problem* CertainAnswers($\mathcal{M}, Q$) *is* coNP*-complete.*

The first part of the previous theorem extends known results on querying RPQs over patterns [10]. The second part of the theorem essentially tells us that computing certain answers of *usual* conjunctive queries over graph mappings is intractable in data complexity. In fact, it is easy to see that each RPQ of the form (x, w, y), where w is a word, can be translated as a conjunctive query over the standard relational representation of graph databases. This fact shows a striking difference between relational and graph mappings. Indeed, it is well-known that for the former the problem of computing certain answers of conjunctive queries is tractable in data complexity [22].

It is thus still relevant finding tractable cases in data complexity of the problem of computing certain answers to queries over graph mappings. This is the problem we investigate in Section 8.

5. FEASIBLE UNIVERSAL REPRESENTATIVE COMPUTATION

The standard way of constructing a universal representative in data exchange is to evaluate the left-hand side of each rule against the source database, and then populate the target as defined by the right-hand side of the respective rules. Hence the most natural way of obtaining restricted classes of graph mappings, for which universal representatives can be computed in polynomial time, is by restricting the left-hand side of rules to queries that allow for efficient computation of its answer set (instead of CNREs, or even CRPQs, that are computationally expensive).

Formally, let $\mathcal{C}$ be a class of CNREs such that the problem of computing $[\![Q(\bar{x})]\!]_G$, for a given query $Q(\bar{x}) \in \mathcal{C}$ and a graph database G, can be solved in polynomial time. Then there is a polynomial time procedure that, given a mapping $\mathcal{M} = (\Sigma_{\mathbf{S}}, \Sigma_{\mathbf{T}}, \mathcal{T})$, in which each left-hand side of a rule in $\mathcal{T}$ is a CNRE in $\mathcal{C}$, and a graph database $G_{\mathbf{S}}$ over $\Sigma_{\mathbf{S}}$, computes a universal representative of $G_{\mathbf{S}}$ under $\mathcal{M}$.

There are several relevant classes $\mathcal{C}$ of CNREs that satisfy the condition mentioned above (i.e. the set $[\![Q]\!]_G$ can be computed in polynomial time, for each graph database G and query $Q \in \mathcal{C}$). These include, for instance, syntactic restrictions of CNREs, based on *acyclicity* [38], for queries of fixed arity.[1] They also include the class of NREs, as shown in Proposition 2.4.

In the rest of the paper, we focus on a particular class of mappings that satisfies the condition mentioned above and allows for a simple definition: The class of mappings such that the left-hand side of each rule is an NRE. We pinpoint the precise complexity of the problem of computing universal representatives for this class, and discuss about the expressiveness of the class for graph data exchange purposes. It is worth remarking that the other natural choice, restricting left-hand sides of rules to acyclic CNREs, yields essentially the same class of mappings (modulo a mild restriction on connectedness), and thus all of the results below hold for acyclic CNREs as well. This follows from recent results on the expressiveness of NREs versus CNREs [11].

5.1 NRE-restricted Mappings

A mapping $\mathcal{M} = (\Sigma_{\mathbf{S}}, \Sigma_{\mathbf{T}}, \mathcal{T})$ is NRE-*restricted*, if each rule in $\mathcal{T}$ is of the form $(x, exp, y) \rightarrow \psi_t(x, y)$, where *exp* is an NRE over $\Sigma_{\mathbf{S}}$ and $\psi_t(x, y)$ is a binary CNRE over $\Sigma_{\mathbf{T}}$.

EXAMPLE 5.1. Consider again the mappings $\mathcal{M}_{12}$, $\mathcal{M}_{23}$ and $\mathcal{M}_{13}$ shown in Example 3.2. Both $\mathcal{M}_{23}$ and $\mathcal{M}_{13}$ are NRE-restricted mappings. On the other hand, the mapping $\mathcal{M}_{12}$ is not NRE-restricted. However, it can easily be shown that the NRE-restricted mapping $\mathcal{M}'_{12}$, defined by rules

$$\begin{aligned}(x, \texttt{creator}^- \cdot [\,\texttt{partOf} \cdot \texttt{series}\,], y) &\rightarrow (x, \texttt{makes}, y)\\ (y, [\,\texttt{creator}\,] \cdot \texttt{partOf} \cdot \texttt{series}\,, w) &\rightarrow (y, \texttt{inConf}, w),\end{aligned}$$

is equivalent to it. □

It follows from the comments above that universal representatives for NRE-restricted mappings can be computed in polynomial time. The following theorem states the precise complexity of the problem:

THEOREM 5.2. *There is a procedure that, given an NRE-restricted mapping $\mathcal{M} = (\Sigma_{\mathbf{S}}, \Sigma_{\mathbf{T}}, \mathcal{T})$ and a graph database $G_{\mathbf{S}}$ over $\Sigma_{\mathbf{S}}$, computes a universal representative of $G_{\mathbf{S}}$ under $\mathcal{M}$ in time $O(|G_{\mathbf{S}}|^2 \cdot |\mathcal{M}|)$.*

Thus, by focusing on NRE-restricted mappings, we reduce the complexity of universal representative computation from $|G_{\mathbf{S}}|^{O(|\mathcal{M}|)}$, implicit in Proposition 4.3, to quadratic in the size of the source graph database. Moreover, we prove that this bound is tight.

PROPOSITION 5.3. *There are families of NRE-restricted mappings $\{\mathcal{M}_n = (\Sigma^n_{\mathbf{S}}, \Sigma^n_{\mathbf{T}}, \mathcal{T}_n)\}_{n \geq 1}$ and graph databases $\{G^m_{\mathbf{S}}\}_{m \geq 1}$, such that $|\mathcal{M}_n|$ is $O(n)$, $|G^m_{\mathbf{S}}|$ is $O(m)$, and every universal representative of $G^m_{\mathbf{S}}$ under $\mathcal{M}_n$ is of size $\Omega(m^2 \cdot n)$.*

[1]Being precise, those restrictions have been defined for relational conjunctive queries, but the same complexity bounds apply for CNREs, if the structural restrictions are defined on the underlying undirected graphs of queries.

Hence, this class allows for simple definition of efficient graph mappings. A natural question is whether the class also has good expressiveness properties, i.e. if it can express interesting graph data exchange mappings. Example 5.1 already suggests this. Notice, in addition, that the navigational exchange mapping presented in the example of Section 3.1 can be expressed by means of NRE-restricted rules, as shown at the end of the section, and hence NRE-restricted mappings allow to express complex navigational properties in graph data exchange.

6. FEASIBLE QUERY ANSWERING IN COMBINED COMPLEXITY

So far, we have achieved to identify an expressive class of mappings (the NRE-restricted mappings) that has good properties in terms of computation of universal representatives. Unfortunately, this class is still not suitable for query answering tasks since Theorem 4.6 tells us that the problem of computing certain answers is hard, even in data complexity, for RPQ queries over RPQ-mappings (and hence over NRE-restricted mappings). Thus, we need a further restriction on this class of mappings in order to obtain a good class for query answering purposes.

In the same way, we cannot expect to have efficient query evaluation algorithms if we allow the whole expressive power of CNREs, or even CRPQs, as a querying mechanism, since they are computationally hard in combined complexity. For this reason we focus solely on queries defined by NREs.

Rigid mappings. The class of mappings we consider restrict the right hand side of dependencies to be conjunctive queries, that is, it prohibits the use of disjunction and recursion in the right-hand side of rules, which are the major contributors to complexity of query answering (in fact, we show in Proposition 6.3 below that lifting any of these restrictions leads to the intractability of the problem of computing certain answers).

Formally, a mapping $\mathcal{M} = (\Sigma_{\mathbf{S}}, \Sigma_{\mathbf{T}}, \mathcal{T})$ is *rigid* if the rules in $\mathcal{T}$ are of the form $\phi(\bar{x}) \rightarrow \exists \bar{z} \theta(\bar{x}, \bar{z})$, for $\phi(\bar{x})$ a CNRE over $\Sigma_{\mathbf{S}}$ and θ a conjunction of NREs of the form (z, exp', z'), where z and z' are variables in $\bar{z} \cup \bar{x}$, and exp' is an NRE over $\Sigma_{\mathbf{T}}$ that makes use of neither disjunction (operator $+$) nor Kleene-star (operator $()^*$). In this section we focus on rigid mappings that are also NRE-restricted.

EXAMPLE 6.1. The mappings $\mathcal{M}_{12}$, $\mathcal{M}_{23}$ and $\mathcal{M}_{13}$ shown in Example 3.2 are all rigid, but only $\mathcal{M}_{23}$ and $\mathcal{M}_{13}$ are rigid and NRE-restricted. □

The previous example shows that rigid NRE-restricted mappings are still capable of expressing interesting data exchange properties. Furthermore, the next theorem shows that these mappings are well-behaved in terms of combined complexity of query evaluation for NREs.

THEOREM 6.2. *Given a rigid and NRE-restricted mapping $\mathcal{M}$ from $\Sigma_{\mathbf{S}}$ to $\Sigma_{\mathbf{T}}$, a source graph database $G_{\mathbf{S}}$ and an NRE exp over $\Sigma_{\mathbf{T}}$,* CERTAINANSWERS *can be solved in time $O(|G_{\mathbf{S}}|^2 \cdot |\mathcal{M}| \cdot |exp|)$.*

To prove Theorem 6.2 we use the fact that for a rigid and NRE-restricted mapping $\mathcal{M}$, the procedure in the proof of Theorem 5.2 can be modified to yield a universal representative in which each edge is labeled with a symbol a from the alphabet $\Sigma_{\mathbf{T}}$. We call such representatives *naïve*, since they can be represented as relational *naïve tables* [26]. One can prove that over these patterns query answering for NREs can be solved by a simple process called *naïve evaluation*, that poses the NRE *exp* directly over the naïve pattern, treating nulls as if they were constants and then discarding pairs with nulls in the answer.

We show next that the class of rigid and NRE-restricted mappings is optimal, as lifting any of the restrictions leads to intractability of the certain answers problem in combined complexity.

PROPOSITION 6.3. *1. The problem* CERTAINANSWERS *is* NP*-hard, even if restricted to queries given by NREs over rigid mappings.*

2. CERTAINANSWERS *is* coNP*-hard, even if restricted to queries given by NREs over NRE-restricted mappings in which no rule uses disjunction (resp. NRE-restricted mappings in which no rule uses Kleene-star $*$).*

GAV mappings. By further restricting the class of mappings, we can obtain even linear combined complexity, in the size of the database, for the problem of computing certain answers for NREs. Consider a mapping $\mathcal{M} = (\Sigma_{\mathbf{S}}, \Sigma_{\mathbf{T}}, \mathcal{T})$ in which each rule in $\mathcal{T}$ is of the form $(x, exp, y) \to (x, a, y)$, where *exp* is an NRE over $\Sigma_{\mathbf{S}}$ and a is a symbol in $\Sigma_{\mathbf{T}}$. This class of mappings defines target symbols in terms of NRE views over the source, which resemble *global-as-view* (GAV) mappings as studied in relational databases [30]. For this reason we call mappings of the above form NRE-GAV (graph) mappings. Clearly, each NRE-GAV mapping is also rigid and NRE-restricted. The class of NRE-GAV mappings defines simple yet useful mappings, as shown by $\mathcal{M}_{23}$ and $\mathcal{M}_{13}$ in Example 3.2, and mapping $\mathcal{M}'_{12}$ in Example 5.1, which are all NRE-GAV.

THEOREM 6.4. *Given an NRE-GAV mapping $\mathcal{M}$ from $\Sigma_{\mathbf{S}}$ to $\Sigma_{\mathbf{T}}$, a graph database $G_{\mathbf{S}}$ over $\Sigma_{\mathbf{S}}$ and an NRE exp over $\Sigma_{\mathbf{T}}$, the problem* CERTAINANSWERS *can be solved in time $O(|G_{\mathbf{S}}| \cdot |exp| \cdot |\mathcal{M}|)$.*

The lower bound proved in Proposition 5.3 also holds for NRE-GAV mappings, so one cannot use universal representative computation in order to obtain Theorem 6.4. Instead, the proof of Theorem 6.4 is based on query *rewriting* techniques, as explained in the following technical result.

LEMMA 6.5. *There is a procedure that, given an NRE-GAV mapping $\mathcal{M} = (\Sigma_{\mathbf{S}}, \Sigma_{\mathbf{T}}, \mathcal{T})$, and an NRE exp over $\Sigma_{\mathbf{T}}$, computes an NRE exp' over $\Sigma_{\mathbf{S}}$, in time $O(|exp| \cdot |\mathcal{M}|)$, such that $\textsc{certain}_{\mathcal{M}}(exp, G_{\mathbf{S}}) = [\![exp']\!]_{G_{\mathbf{S}}}$ for every source graph database $G_{\mathbf{S}}$.*

Hence in order to evaluate the certain answers of an NRE *exp* over $G_{\mathbf{S}}$ under the NRE-GAV mapping $\mathcal{M}$, we can perform the following algorithm: Compute from $\mathcal{M}$ and *exp* the NRE exp'. Then evaluate exp' over $G_{\mathbf{S}}$ in time $O(|G_{\mathbf{S}}| \cdot |exp'|)$ (as stated in Proposition 2.4), and thus in time $O(|G_{\mathbf{S}}| \cdot |exp| \cdot |\mathcal{M}|)$.

7. INTERLUDE ON COMPOSING GRAPH MAPPINGS

We have seen that the class of NRE-GAV mappings has good properties for query answering. In this section we take advantage of those properties, in particular the existence of source rewritings, and make a case for the usefulness of this language in a rather different scenario: when composing schema mappings. Composition has been identified as a fundamental process for several interoperability tasks [33, 12], and, as such, it has received considerable attention in relational and XML data exchange [31, 33, 23, 35, 7, 2]. On the other hand, the composition of schema mappings for graph databases has not yet been considered in the literature.

Given mappings $\mathcal{M}_1$ and $\mathcal{M}_2$, the composition $\mathcal{M}_1 \circ \mathcal{M}_2$ is a new mapping that, intuitively, has the same effect as the application of $\mathcal{M}_1$ and $\mathcal{M}_2$ one after the other. Formally, given mappings $\mathcal{M}_1$ from Σ_1 to Σ_2, and $\mathcal{M}_2$ from Σ_2 to Σ_3, the composition of $\mathcal{M}_1$ and $\mathcal{M}_2$ is the mapping from Σ_1 to Σ_3 defined by $[\![\mathcal{M}_1]\!] \circ [\![\mathcal{M}_2]\!] = \{(G_1, G_3) \mid$ there exists G_2 over Σ_2 such that $(G_1, G_2) \in [\![\mathcal{M}_1]\!]$ and $(G_2, G_3) \in [\![\mathcal{M}_2]\!]\}$ [33, 23].

One fundamental question in this context is definability of composition: given $\mathcal{M}_1$ and $\mathcal{M}_2$ defined in some mapping language, what is the language needed to specify the composition of both mappings? Of particular interest is the search for a mapping language $\mathcal{L}$ that is *closed under composition.* This means that for any two mappings $\mathcal{M}_1$ and $\mathcal{M}_2$ specified in $\mathcal{L}$, the composition $[\![\mathcal{M}_1]\!] \circ [\![\mathcal{M}_2]\!]$ can also be specified in $\mathcal{L}$ (i.e. there is a mapping $\mathcal{M}$ in $\mathcal{L}$ such that $[\![\mathcal{M}]\!] = [\![\mathcal{M}_1]\!] \circ [\![\mathcal{M}_2]\!]$). It has been shown that in the relational scenario the language of GAV mappings is closed under composition [23]. The next result shows that for NRE-GAV mappings we obtain a similar good behavior. The proof is based on the rewriting properties of NRE-GAV mappings that we stated in the previous section.

THEOREM 7.1. *The language of NRE-GAV mappings is closed under composition.*

EXAMPLE 7.2. Consider again mappings $\mathcal{M}_{12}$, $\mathcal{M}_{23}$ and $\mathcal{M}_{13}$ in Example 3.2. Recall that $\mathcal{M}_{23}$ is the NRE-GAV mapping defined by the single rule:

$$\big(x, (\texttt{makes} \cdot \texttt{makes}^-)^+, y\big) \quad \to \quad (x, \texttt{confConnected}, y).$$

Also, we know from Example 5.1 that $\mathcal{M}_{12}$ is equivalent to the NRE-GAV mapping $\mathcal{M}'_{12}$ defined by rules:

$$\begin{array}{rcl}(x,\, \texttt{creator}^- \cdot [\,\texttt{partOf} \cdot \texttt{series}\,], y) & \to & (x, \texttt{makes}, y)\\ (y,\, [\,\texttt{creator}\,] \cdot \texttt{partOf} \cdot \texttt{series}\,, w) & \to & (y, \texttt{inConf}, w),\end{array}$$

It can be proved that the composition $[\![\mathcal{M}_{12}]\!] \circ [\![\mathcal{M}_{23}]\!]$ is defined by NRE-GAV mapping $\mathcal{M}_{13}$ from Σ_1 to Σ_3. Recall that this mapping consists of the single rule:

$$\big(x, (\texttt{creator}^- \cdot [\,\texttt{partOf} \cdot \texttt{series}] \cdot \texttt{creator})^+, y\big) \\ \to (x, \texttt{confConnected}, y).$$

As a proof of concept, if G is the graph database in Figure 1, and one exchanges the data in G by directly applying the rules in $\mathcal{M}_{13}$, then one obtains the graph database in Figure 3. This is exactly the same graph database that is obtained by first exchanging data from G by applying rules in $\mathcal{M}_{12}$, which yields the graph database G' in Figure 2, and then from G' by applying the rules in $\mathcal{M}_{23}$. $\square$

The previous work on mappings for graph databases has focused on mappings defined by 2RPQs or C2RPQs [14, 15, 16]. Thus, it is natural to ask whether one can obtain a closure result, similar to Theorem 7.1, for mappings based on those languages. We show that this is not the case, when restricted to the GAV case, which is an advantage of NRE-mappings over the other ones seen in this paper. In what follows, we use the notion of $\mathcal{L}$-GAV mappings, where $\mathcal{L}$ is class of C2RPQs, for the class of mappings specified by rules of the form $\varphi(x, y) \to (x, a, y)$, where $\varphi(x, y)$ is a binary query in $\mathcal{L}$ over the source and a is a symbol in the target.

We start with languages without conjunction, that is, 2RPQs and RPQs. The next proposition shows, in particular, that the nesting feature of NREs is necessary to obtain the closure result in Theorem 7.1.

PROPOSITION 7.3. *There exist RPQ-GAV mappings $\mathcal{M}_{12}$ from Σ_1 to Σ_2 and $\mathcal{M}_{23}$ from Σ_2 to Σ_3, such that the mapping $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is not equivalent to a 2RPQ-GAV mapping.*

One may also ask whether the use of conjunctions over 2RPQs or RPQs can lead to a closed mapping language. The following result shows that this is not the case.

PROPOSITION 7.4. *There are CRPQ-GAV mappings $\mathcal{M}_{12}$ from Σ_1 to Σ_2 and $\mathcal{M}_{23}$ from Σ_2 to Σ_3, such that the mapping $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is not equivalent to a CNRE-GAV mapping.*

As a final remark we note that the restriction to GAV mappings is crucial for obtaining the closure result in Theorem 7.1. In fact, it is not difficult to adapt results by Fagin et al. [23] to show that our graph mappings cannot express even the composition of two rigid RPQ-mappings.

8. FEASIBILITY QUERY ANSWERING IN DATA COMPLEXITY

In this section we study the data complexity of the problem of evaluating certain answers, that is, we study the problem CERTAINANSWERS$(\mathcal{M}, Q)$, for a fixed mapping $\mathcal{M}$ from $\Sigma_\mathbf{S}$ to $\Sigma_\mathbf{T}$, and a fixed n-ary query Q over $\Sigma_\mathbf{T}$. Recall that the input to this problem is a graph database $G_\mathbf{S}$ and an n-ary tuple $\bar{t}$, and the question is whether $\bar{t} \in$ CERTAIN$_\mathcal{M}(Q, G_\mathbf{S})$.

We presented in Section 6 a class of mappings (rigid NRE-restricted mappings) and queries (NREs), that allow for efficient computation of certain answers in combined complexity. It is thus possible that a similar result in data complexity can be obtained for less restrictive scenarios. For instance, using essentially the same proof techniques as in the proof of Theorem 6.2, one can show that the data complexity of computing certain answers for the whole class of CNREs, under mappings that do not make use of union or recursion in the right-hand side of rules (i.e. rigid mappings, as introduced in Section 6), is polynomial.

PROPOSITION 8.1. *Let $\mathcal{M}$ be a rigid mapping from $\Sigma_\mathbf{S}$ to $\Sigma_\mathbf{T}$ and Q a CNRE over $\Sigma_\mathbf{T}$. Then* CERTAINANSWERS$(\mathcal{M}, Q)$ *can be solved in polynomial time.*

This of course does not rule out the possibility of obtaining tractability results for more general mappings that make use of arbitrary NREs in the right-hand side of the rules. However, we know from Theorem 4.6 that the problem of computing certain answers is intractable already for RPQ-mappings and RPQs given as words, which are nothing more than usual conjunctive queries over the relational representation of databases. This shows that there is a delicate trade-off between the expressive power of mappings and the complexity of query answering, and it appears that none of the restrictions considered so far in this paper will lead us to a new tractable fragment. Instead, we follow here a different approach, based on structural properties of universal representatives and queries, as described next.

Recall that in Section 4.1 we prove that for every mapping $\mathcal{M}$ from $\Sigma_\mathbf{S}$ to $\Sigma_\mathbf{T}$, and graph database $G_\mathbf{S}$ over $\Sigma_\mathbf{S}$, there exists a graph pattern $\pi_\mathbf{T}$ such that $\text{Sol}_\mathcal{M}(G_\mathbf{S}) = \text{Rep}_{\Sigma_\mathbf{T}}(\pi_\mathbf{T})$. Moreover, this pattern $\pi_\mathbf{T}$ can be computed in polynomial time. Thus, given an n-ary query Q over $\Sigma_\mathbf{T}$, and an n-ary tuple $\bar{t}$, checking if $\bar{t} \in$ CERTAIN$_\mathcal{M}(Q, G_\mathbf{S})$ is equivalent to checking whether $\bar{t}$ belongs to the set:

$$\Box Q(\pi_\mathbf{T}) = \bigcap \{ [\![Q]\!]_{G_\mathbf{T}} \mid G_\mathbf{T} \in \text{Rep}_{\Sigma_\mathbf{T}}(\pi_\mathbf{T}) \}.$$

This means that the complexity of computing certain answers is essentially located at the level of computing the set $\Box Q(\pi_\mathbf{T})$, for a graph pattern $\pi_\mathbf{T}$. For this reason, we concentrate our efforts in finding interesting tractable cases of this particular problem. More precisely, we present a class of patterns and queries that has the desired properties regarding the computation of $\Box Q(\pi_\mathbf{T})$, and discuss on the optimality and practical applicability of such class.

It is important to note that we focus solely on graph patterns $\pi_\mathbf{T}$ that do not use null values. This simplifies to a great extent the exposition of our results and, moreover, the study of patterns without null values is interesting in its own right: This class of patterns contains universal representatives for NRE-mappings and, more generally, for all mappings that have no existential quantification in the right-hand side of rules. (In fact, for these mappings the procedure used in the proof of Proposition 4.3 always produces a universal representative $\pi_\mathbf{T}$ with such characteristics).

8.1 Structural restrictions

It follows from the lower bound in Theorem 4.6 that the problem of checking whether a tuple belongs to $\Box Q(\pi)$ is intractable, in particular, coNP-hard, even if π is a graph pattern without null values. However, by staring at the proof, one observes that this intractability arises from some rather "unnatural" classes of patterns. This gives hope to the idea of finding "natural" classes of patterns with tractable query evaluation. In this section we propose a set of structural restrictions on graph patterns, and on the interaction between patterns and queries, that gives rise to a class with good properties for computing certain answers.

We concentrate from now on a particular class of queries: CRPQs in which each regular expression occurring in the query does not use the Kleene-star (that is, the regular language defined by this expression is finite). We say that a CRPQ satisfying this restriction is *tame*. The class of tame CRPQs is relevant as it contains, among others, all conjunctive queries over the standard relational representation of graph databases. Moreover, notice that the intractability result in Theorem 4.6 was actually proved for a tame RPQ (since the query in such theorem is a single word) over an RPQ-mapping, and thus tame CRPQs are intractable even over the class of patterns that do not use null values.

Our class of patterns is defined by two conditions that, when used together, yield tractability to the problem of computing certain answers of tame CRPQs over patterns without null values. As we later show, the obtained class of patterns is, in a precise sense, maximal for tractability. Our first condition is a very simple restriction on the structure of patterns.

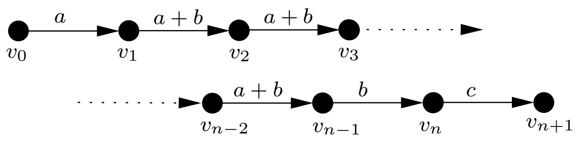
(C1) We require the *out-degree* of nodes in patterns to be bounded by a constant.

The out-degree of a node u in pattern π is defined as the number of edges of the form (u, exp, u') in π. The out-degree of a pattern π, denoted by $\mathtt{out}(\pi)$, is the maximum out-degree of a node in π. Let us denote by $\mathcal{OD}_{\leq d}$ the class of patterns π such that $\mathtt{out}(\pi)$ is at most d. Considering bounded out-degree is a common assumption in the study of complex networks [32]. This assumption reflects the idea that while incoming edges are generated distributively by all agents in a network, the outgoing edges are generated locally by a single node, and therefore its number can be assumed to be small as compared to the size of the whole network. By using condition (C1) we obtain a first simple tractability result.

PROPOSITION 8.2. *Let Q be a fixed tame CRPQ without existentially quantified variables, $\bar{t}$ a tuple of node ids, and $d \geq 0$ a fixed value. There is an algorithm that given a pattern π in $\mathcal{OD}_{\leq d}$ without null values, checks whether $\bar{t} \in \Box Q(\pi)$ in polynomial time.*

As we later show, tractability for tame CRPQs in general cannot be obtained by just bounding the out-degree of patterns (see Theorem 8.6 (2)). Thus, we provide another restriction that together with (C1) defines a tractable case for tame CRPQs. Let us illustrate the intuition of this condition by means of an example.

EXAMPLE 8.3. Let n be odd, and define π_n as:



and consider the tame CRPQs $Q = \exists x \exists y\ (x, aa + bb, y)$ and $Q' = \exists x \exists y\ (x, (a + b)bc, y)$.

Notice that the certain answers of Q and Q' over π_n are $\mathtt{true}$. In order to show this for the case of Q, one needs to check over all possible combinations resulting of assigning label either a or b to each of the edges of the form $(v_i, a + b, v_{i+1})$. On the other hand, in the case of Q' it is sufficient to inspect the labels of the last 3 edges of π_n, and thus, we essentially have to check only two combinations. $\Box$

The above example suggests that existential variables are problematic for query answering only when they can be witnessed by several nodes in the pattern, whereas queries in which the possible witnesses are limited should behave better. We formalize this intuition with our second condition, but first we need to introduce some terminology.

Let $\pi = (N, D)$ be a pattern over Σ and Q a tame CRPQ over the same schema. Assume that $\bar{x}$ is the tuple of free variables in Q and $\bar{y}$ the tuple of variables that are existentially quantified in Q, and consider a function $f : \bar{y} \to N \cup D$, that is, f maps each existentially quantified variable of Q either to a node (N) or to an edge (D) in π. Given a tuple $\bar{t}$, we say that f is *meaningful* for π, Q and $\bar{t}$, if there is a graph database $G \in \mathrm{Rep}_{\Sigma}(\pi)$ and a mapping σ, such that the following holds:

1. G is obtained from π by replacing each edge of the form $e = (u, exp, v) \in D$ with a graph database G_e, of fresh node ids except for u and v, such that $(u, v) \in [\![exp]\!]_{G_e}$,
2. σ is a mapping from the variables of Q into the nodes of G, that witnesses $\bar{t} \in [\![Q]\!]_G$, in particular, $\sigma(\bar{x}) = \bar{t}$,
3. for each y in $\bar{y}$ we have $\sigma(y) = f(y)$ if $f(y) \in N$, or $\sigma(y)$ is a node in $\rho_{f(y)}$ if $f(y) \in D$. That is, if $f(y)$ is the edge e of π, then σ maps y into a node in the graph database ρ_e that replaces e in G (and σ and f coincide for all variables that f sends to nodes of π).

We denote by $\mathtt{mf}(Q, \pi, \bar{t})$ the number of meaningful functions for π, Q and $\bar{t}$. We need a final definition before stating our condition. Given a pattern π, the *complete graph induced by* π, denoted by $\mathtt{comp}(\pi)$, is the graph database obtained from π by removing each *incomplete edge* of π, that is, each edge labeled with a regular expression that is syntactically distinct from a symbol in Σ. We are now ready to state our second structural condition, which depends on π, Q, and $\bar{t}$.

(C2) If $\bar{t}$ does not belong to the evaluation of Q over $\mathtt{comp}(\pi)$, then the number of meaningful functions for π, Q and $\bar{t}$, is logarithmically bounded by the size of π.

From a practical point of view, we can understand this condition as follows. Assume that a query Q is not implied by the complete data of π. Then one can expect that the interaction between π and Q is rather sophisticated, and hence that Q should only have a small bound on the number of potential witnesses in a completion of the pattern. This is precisely what condition (C2) expresses, assuming such bound to be logarithmic, since meaningful functions essentially encode the potential witnesses of Q in a completion of the pattern.

Formally, given a tame CRPQ Q, a tuple $\bar{t}$, and a value $k \geq 0$, we denote by $\mathcal{MF}^{Q,\bar{t}}_{\leq k}$ the class of patterns π for which either $\bar{t} \in [\![Q]\!]_{\mathtt{comp}(\pi)}$ or $\mathtt{mf}(Q, \pi, \bar{t}) \leq k \cdot \log(|\pi|)$, where $|\pi|$ is the size of π measured as its number of edges. The following lemma shows that membership in $\mathcal{OD}_{\leq d} \cap \mathcal{MF}^{Q,\bar{t}}_{\leq k}$ can be efficiently checked.

LEMMA 8.4. *Given a tame CRPQ Q, a tuple $\bar{t}$ and values $d, k \geq 0$, checking if pattern π without null values belongs to $\mathcal{OD}_{\leq d} \cap \mathcal{MF}^{Q,\bar{t}}_{\leq k}$ can be done in polynomial time w.r.t. $|\pi|$.*

This finishes the presentation of properties (C1) and (C2). The following result shows that checking whether $\bar{t} \in \Box Q(\pi)$ can be efficiently decided for patterns in $\mathcal{OD}_{\leq d} \cap \mathcal{MF}^{Q,\bar{t}}_{\leq k}$.

THEOREM 8.5. *Let Q be a fixed tame CRPQ, and $\bar{t}$ a tuple of node ids. Moreover, let $d, k \geq 0$ be fixed values. Then there is an algorithm that given a pattern π in $\mathcal{OD}_{\leq d} \cap \mathcal{MF}^{Q,\bar{t}}_{\leq k}$ without null values, checks whether $\bar{t} \in \Box Q(\pi)$ in polynomial time.*

It is possible to prove that the two described classes of patterns are maximal with respect to tractability, as lifting either condition (C1) or (C2), raises the complexity. It can also be proved that the class of tame CRPQs is maximal to obtain tractability for patterns satisfying (C1) and (C2). All this is summarized in our last result.

THEOREM 8.6.

1. *There is a tame CRPQ Q, a tuple $\bar{t}$, and a value $k \geq 0$, such that checking $\bar{t} \in \Box Q(\pi)$, for the class of patterns π in $\mathcal{MF}^{Q,\bar{t}}_{\leq k}$ without null values, is* coNP*-complete.*
2. *There is a tame CRPQ, a tuple $\bar{t}$, and a value $d \geq 0$, such that checking $\bar{t} \in \Box Q(\pi)$, for the class of patterns π in $\mathcal{OD}_{\leq d}$ without null values, is* coNP*-complete.*
3. *There exist a non-tame RPQ Q, a pair (u, v) of node ids, and values $d, k \geq 0$ such that checking whether $(u, v) \in \Box Q(\pi)$, for patterns π in $\mathcal{OD}_{\leq d} \cap \mathcal{MF}^{Q,\bar{t}}_{\leq k}$ without null values, is* coNP*-complete.*

It still deserves to be studied how to link our positive results for graph patterns to the context of computing mapping-based certain answers. We believe this to be an interesting and challenging topic for future research, and the results in this section are a first step in such direction.

9. CONCLUDING REMARKS

In this paper we have studied interoperability issues for graph databases. In particular, we have proposed a mapping language based on CNREs and studied data exchange, query answering and also the composition of mappings specified in the language. We have also identified relevant fragments of the language for which those problems can be solved efficiently. There are interesting issues that remain open. One of particular interest is the development of a mapping language able to express the composition of mappings beyond the class of GAV-NRE.

Acknowledgments We thank Leonid Libkin and Domagoj Vrgoč for their comments. Barceló is funded by Fondecyt Grant 1110171, Pérez by Fondecyt grant 11110404 and VID grant U-Inicia 11/04 Universidad de Chile, and Reutter by EPSRC grant G049165 and FET-Open Project FoX, grant agreement 233599.

10. REFERENCES

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 1999.

[2] S. Amano, L. Libkin, and F. Murlak. XML schema mappings. In *PODS*, pages 33–42, 2009.

[3] M. Arenas, P. Barceló, L. Libkin, and F. Murlak. *Relational and XML Data Exchange*. Morgan & Claypool Publishers, 2010.

[4] M. Arenas, P. Barceló, and J. L. Reutter. Query languages for data exchange: Beyond unions of conjunctive queries. *Theory Comput. Syst.*, 49(2):489–564, 2011.

[5] M. Arenas and L. Libkin. XML data exchange: Consistency and query answering. *J. ACM*, 55(2), 2008.

[6] M. Arenas, J. Pérez, and J. L. Reutter. Data exchange beyond complete data. In *PODS*, pages 83–94, 2011.

[7] M. Arenas, J. Pérez, J. L. Reutter, and C. Riveros. Composition and inversion of schema mappings. *SIGMOD Record*, 38(3):17–28, 2009.

[8] P. Barceló. Logical foundations of relational data exchange. *SIGMOD Record*, 38(1):49–58, 2009.

[9] P. Barceló, C. A. Hurtado, L. Libkin, and P. T. Wood. Expressive languages for path queries over graph-structured data. In *PODS*, pages 3–14, 2010.

[10] P. Barceló, L. Libkin, and J. L. Reutter. Querying graph patterns. In *PODS*, pages 199–210, 2011.

[11] P. Barceló, J. Pérez, and J. L. Reutter. Relative expressiveness of nested regular expressions. In *AMW*, pages 180–195, 2012.

[12] P. Bernstein. Applying model management to classical meta data problems. In *CIDR*, 2003.

[13] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Vardi. Containment of conjunctive regular path queries with inverse. In *KR*, pages 176–185, 2000.

[14] D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query processing for regular path queries with inverse. In *PODS*, pages 58–66, 2000.

[15] D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query answering and query containment over semistructured data. In *DBPL*, pages 40–61, 2001.

[16] D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. Simplifying schema mappings. In *ICDT*, pages 114–125, 2011.

[17] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, pages 77–90, 1977.

[18] M. Consens and A. Mendelzon. GraphLog: A visual formalism for real life recursion. In *PODS*, pages 404–416, 1990.

[19] P. Cudré-Mauroux and S. Elnikety. Graph data management systems for new application domains. *PVLDB*, 4(12):1510–1511, 2011.

[20] L3s dblp bibliography db: `http://dblp.l3s.de/d2r/`.

[21] R. Fagin. Inverting schema mappings. *TODS*, 32(4), 2007.

[22] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *TCS*, 336(1):89–124, 2005.

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

[24] G. H. L. Fletcher, M. Gyssens, D. Leinders, J. V. den Bussche, D. V. Gucht, S. Vansummeren, and Y. Wu. Relative expressive power of navigational querying on graphs. In *ICDT*, pages 197–207, 2011.

[25] G. Gottlob, C. Koch, and R. Pichler. Efficient algorithms for processing XPath queries. *ACM Trans. Database Syst.*, 30(2):444–491, 2005.

[26] T. Imielinski and W. Lipski. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.

[27] P. G. Kolaitis. Schema mappings, data exchange, and metadata management. In *PODS*, pages 61–75, 2005.

[28] P. G. Kolaitis, J. Panttaja, and W.-C. Tan. The complexity of data exchange. In *PODS*, pages 30–39, 2006.

[29] M. W. Krentel. The Complexity of Optimization Problems. *Journal of Computer and System Sciences*, 36(3):490–509, 1988.

[30] M. Lenzerini. Data integration: a theoretical perspective. In *PODS*, pages 233–246, 2002.

[31] J. Madhavan and A. Y. Halevy. Composing mappings among data sources. In *VLDB*, pages 572–583, 2003.

[32] G. S. Manku, M. Naor, and U. Wieder. Know thy neighbor's neighbor: the power of lookahead in randomized P2P networks. In *STOC*, pages 54–63, 2004.

[33] S. Melnik. *Generic Model Management: concepts and Algorithms*, volume 2967 of *LNCS*. Springer, 2004.

[34] A. O. Mendelzon and P. T. Wood. Finding regular simple paths in graph databases. *SIAM J. Comput.*, 24(6):1235–1258, 1995.

[35] A. Nash, P. A. Bernstein, and S. Melnik. Composition of mappings given by embedded dependencies. In *PODS*, pages 172–183, 2005.

[36] J. Pérez, M. Arenas, and C. Gutierrez. nSPARQL: A navigational language for RDF. *J. Web Sem.*, 8(4):255–270, 2010.

[37] P. T. Wood. Query languages for graph databases. *Sigmod Record*, 41(1):50–60, 2012.

[38] M. Yannakakis. Algorithms for acyclic database schemes. In *VLDB*, pages 82–94, 1981.