PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
SCHOOL OF ENGINEERING

ON THE COMPLEXITY OF BIDIRECTIONAL CONSTRAINTS FOR DATA EXCHANGE

GABRIEL SIMÓN DIÉGUEZ FRANZANI

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Advisor:
MARCELO ARENAS S.

Santiago de Chile, December 2014

PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
SCHOOL OF ENGINEERING

ON THE COMPLEXITY OF BIDIRECTIONAL CONSTRAINTS FOR DATA EXCHANGE

GABRIEL SIMÓN DIÉGUEZ FRANZANI

Members of the Committee:
MARCELO ARENAS S.
JUAN L. REUTTER D.
JORGE PÉREZ R.
JOSÉ LUIS ALMAZÁN C.

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Santiago de Chile, December 2014

To everyone who feels to be my family.

ACKNOWLEDGEMENTS

I would like to thank the following people (in no particular order):

Marcelo Arenas for three reasons. First, for the opportunity of being part of such an amazing research group, full of incredibly talented and fun people; second, for the huge opportunity of giving lectures; and finally, for his dedication as my advisor, having always an idea when things were not going through.

My "second advisor" Jorge Pérez for his time, encouragement and support, and for always challenging me with new problems.

Juan Reutter for his incredibly useful insights for many proofs, for always having time for a little talk, and for always having the right professional or personal advice.

José Luis Almazán, president of my thesis committee, for making the final part of this long process fast and simple.

My office colleagues, the old and new (BANG!) guys, for all the good times in these years. It is an honour to be the link between two "Fishbowl 10" generations.

All other friends in DCC / PUC: the ones from my undergraduate years, the ones from the master's times, the ones who laugh at Dinkleberg, etc., and all other people in the "database gang" and the Nucleus for the fun trips and chats.

My parents, my brothers, my family and my friends for their invaluable love and support, and for always believing in me.

And finally, Daniela for being such a great support through all these years. Without the huge amounts of love, time, patience, kindness and fun she has given to me / had with me, I would not be here finishing this thesis.

My postgraduate studies were partially funded by CONICYT Master's scholarship CONICYT-PCHA/Magíster Nacional/2013 - 221320842 and the Millennium Nucleus Center for Semantic Web Research under Grant NC120004.

TABLE OF CONTENTS

LIST OF TABLES

ABSTRACT

Schema mappings are of fundamental importance in data management; they have proved to be the essential building block for several data-interoperability tasks such as data exchange, data integration and peer data management. Most of the research on schema mappings has focused on mappings specified by st-tgds, which although natural and simple to specify, fail to impose enough conditions to unambiguously define what are the instances that should be materialized when exchanging data. Recently, bidirectional constraints have been proposed to specify mappings; they impose at the same time constraints over the source and target instances participating in them, and have the potential to minimize the ambiguity in the description of the target instances.

In this thesis, we expand the formal study of bidirectional constraints; in particular, we study the computational complexity of two fundamental problems in the context of data exchange: checking the existence of solutions and answering queries. In the former, we analyze both the data and combined complexity, providing upper and lower bounds for different scenarios. In the latter, we also distinguish between several query languages with different expressive powers. In the proofs we introduce some new techniques, like a modified version of the classical chase procedure.

Keywords: Data exchange, Schema mappings, Bidirectional constraints, Query answering, Computational complexity

RESUMEN

Los mapeos de esquemas tienen una importancia fundamental en el manejo de datos, pues han mostrado ser la base para numerosas tareas de interoperabilidad de datos como intercambio de información, integración de datos y manejo de datos entre pares. La mayor parte de la investigación sobre mapeos de esquemas se ha concentrado en mapeos descritos por st-tgds, las cuales si bien son naturales y simples de especificar, no logran imponer suficientes condiciones para definir sin ambigüedad cuáles son las instancias que debieran materializarse al intercambiar información. Recientemente, se ha propuesto el uso de dependencias bidireccionales en la especificación de mapeos de esquemas, siendo capaces de imponer al mismo tiempo restricciones sobre las instancias del *source* y del *target* que participan en ellos, y teniendo el potencial de minimizar la ambigüedad en la descripción de las instancias *target*.

En esta tesis continuamos con el estudio formal sobre las dependencias bidireccionales. En particular, estudiamos la complejidad computacional de dos problemas fundamentales en el contexto de intercambio de información: verificar la existencia de soluciones y contestar consultas. En el primer caso, se analiza tanto la complejidad de los datos como la complejidad combinada, mostrando cotas superiores e inferiores en distintos escenarios. En el segundo caso, además distinguimos entre diversos lenguajes de consulta con distintos poderes expresivos. En las demostraciones introducimos algunas técnicas nuevas, como una versión modificada del clásico algoritmo de *chase*.

Palabras Claves: Intercambio de información, Mapeo de esquemas, Dependencias bidireccionales, Contestar consultas, Complejidad computacional

1. INTRODUCTION

A schema mapping is a high-level specification that describes how data from a source schema is to be mapped to a target schema. Schema mappings are of fundamental importance in data management today. In particular, they have proved to be the essential building block for several data-interoperability tasks such as data exchange (Fagin, Kolaitis, Miller, & Popa, 2005), data integration (Lenzerini, 2002), and peer data management (De Giacomo, Lembo, Lenzerini, & Rosati, 2007).

In the relational-database context, schema mappings are usually specified by using a logical language considering the set of relation names (or table names) of the database schemas as vocabulary. For example, consider two independent database schemas:

$\mathbf{S}$ containing relation $\texttt{Employee}(\texttt{name}, \texttt{lives\_in}, \texttt{works\_in})$, and

$\mathbf{T}$ containing relation $\texttt{Shuttle}(\texttt{name}, \texttt{destination})$.

Relation `Employee` in schema $\mathbf{S}$ is used to store employee names and the places where they live in and work in. Relation `Shuttle` in schema $\mathbf{T}$ is intended to store names of employees that must take the shuttle bus to reach the places where they work in (destination). A possible way of relating schemas $\mathbf{S}$ and $\mathbf{T}$ is by using the following first-order logic formula:

$$\forall x \forall z \big(\ \exists y (\ \texttt{Employee}(x, y, z) \wedge y \neq z \ ) \ \rightarrow \ \texttt{Shuttle}(x, z) \ \big). \tag{1.1}$$

The above formula essentially states that if relation `Employee` stores an employee that lives in a place different from which she/he works in, then the employee name and the place where she/he works in should be stored in relation `Shuttle`.

Formula (1.1) describes a mapping between schemas $\mathbf{S}$ and $\mathbf{T}$ that is given by an *implication* where the left-hand side of the implication is a query over $\mathbf{S}$ and the right-hand side of the implication is a query over $\mathbf{T}$. This class of implication formulas has been the most widely used to specify schema mappings both in theoretical studies (Lenzerini,

2002; Fagin, Kolaitis, Miller, & Popa, 2005; Kolaitis, 2005; De Giacomo et al., 2007; Arenas, Pérez, & Riveros, 2009; Arenas, Pérez, Reutter, & Riveros, 2010; Arenas, Pérez, & Reutter, 2011; Arenas, Pérez, & Reutter, 2013; Pérez, 2011) and in practical applications (Hernández et al., 2002; Haas, Hernández, Ho, Popa, & Roth, 2005; Bernstein, Green, Melnik, & Nash, 2006).

In particular, schema mappings specified by implication formulas have been the preferred formalism for exchanging data (Fagin, Kolaitis, Miller, & Popa, 2003, 2005; Fagin, Kolaitis, & Popa, 2003; Libkin, 2006; Gottlob & Nash, 2006; De Giacomo et al., 2007). In the data exchange context one is given a *source database instance* and a schema mapping. Then the problem is to find a *target database instance* that satisfies the constraints imposed by the schema mapping. Consider Formula (1.1) above and the database D_1 over (the source) schema $\mathbf{S}$ given by

D_1:

Emp	name	lives_in	works_in
	Juan	Santiago	Valparaiso
	Diego	Santiago	Santiago

A possible solution for the data exchange problem is the database D_2 over (the target) schema $\mathbf{T}$ given by

D_2:

Shuttle	name	destination
	Juan	Valparaiso

Notice that D_1 together with D_2 satisfy the constraints imposed by (1.1) (considering the standard first-order logic semantics). Nevertheless, there are other solutions for this data exchange problem. Consider the databases D_3 and D_4 given by

D_3:

Shuttle	name	destination
	Juan	Valparaiso
	Diego	Santiago

D_4:

Shuttle	name	destination
	Juan	Valparaiso
	Juan	Santiago
	Alberto	Curico

In this case we have that D_1 together with D_3 satisfy Formula (1.1). We also have that D_1 and D_4 satisfy Formula (1.1). Thus, the database instances D_3 and D_4, although less natural than D_2, are also solutions for the data exchange problem. This sort of anomaly is caused by the semantics of the implication formula. Notice that the formula used to exchange data is not restricting the possibility of adding arbitrary tuples to relation `Shuttle` in the target database.

The semantics of implication formulas has raised several issues in data exchange. One of them is the problem of deciding what is a good solution for the data exchange problem. In Fagin, Kolaitis, Miller, and Popa (2003) and Fagin, Kolaitis, Miller, and Popa (2005), it was proposed to consider the *minimal* (or *universal*) solutions as the only solutions that are good for data exchange. Although D_2, D_3 and D_4 are considered valid solutions for the data exchange problem, only D_2 is considered a good solution according to Fagin, Kolaitis, Miller, and Popa (2005). Towards solving the same problem, Libkin (2006) has proposed to change the semantics of schema mappings given by implication formulas by considering a *closed-world assumption*. Thus, formulas are no longer evaluated using the standard first-order logic semantics. Under the semantics proposed by Libkin (2006), D_3 and D_4 do not satisfy the constraints given by Formula (1.1), and thus, they are no longer valid solutions for D_1. Although this new semantics departs from a classical first-order logic semantics, it has proved to have good properties in terms of materialization of target instances. Other lines of research include the proposal of alternative notions for answering target queries, in particular, non-monotone queries (Hernich, 2013) and aggregate queries (Afrati & Kolaitis, 2008).

In Arenas, Diéguez, and Pérez (2014), it was argued that there is a more simple and natural way of dealing with the above mentioned issue. In that work, we decided to follow a different approach and, instead of using ad-hoc solutions for each of the mentioned issues, we used a mapping-specification language that imposes enough constraints over possible target instances in order to minimize the uncertainty when exchanging data. In

that way one can use standard first-order logic notions to define the semantics of mappings, the possible target solutions as well as the process of answering target queries. In our example, if one wants D_2 to be *the* solution for the data exchange problem, then that should be clear in the specification of the schema mapping. Thus, instead of an implication formula, one should use a *bidirectional implication*. Therefore, our schema mapping should be specified as:

$$\forall x \forall z \big(\ \exists y (\ \texttt{Employee}(x, y, z) \wedge y \neq z \ ) \ \leftrightarrow \ \texttt{Shuttle}(x, z) \ \big). \tag{1.2}$$

If one considers D_1 as a source database and the schema mapping specified by Formula (1.2), then the only possible solution for the data exchange problem is D_2, since D_2 is the only database instance over schema $\mathbf{T}$ that together with D_1 satisfies (1.2).

Thus, in Arenas, Diéguez, and Pérez (2014) we proposed to use what we call *bidirectional constraints* to specify mappings. These specifications impose at the same time constraints over the source and target instances participating in a mapping, and have the potential to minimize the ambiguity in the description of the target instances that should be materialized when exchanging data. Bidirectional constraints are formulas of the form $\forall \bar{x} \big(\varphi(\bar{x}) \ \leftrightarrow \ \psi(\bar{x}) \ \big)$ where $\varphi(\bar{x})$ is a formula over the source schema, and $\psi(\bar{x})$ is a formula over the target schema. One can obtain several different languages of bidirectional constraints depending on the formulas allowed in the source and target parts.

Although bidirectional constraints are natural in several scenarios, they have been almost disregarded in the study and use of schema mappings. The reason for that is manifold. First of all, from a logical point of view, is more simple to deal with unidirectional implications, since bidirectional implications impose more restrictions on the database instances. Second, it is not clear how to use standard database techniques like the *chase procedure* (Maier, Mendelzon, & Sagiv, 1979) with bidirectional implications. Notice that the chase procedure lies in the core of almost all algorithms used in data exchange (Fagin, Kolaitis, Miller, & Popa, 2003, 2005; Fagin, Kolaitis, & Popa, 2003; Gottlob & Nash, 2006). In Arenas, Diéguez, and Pérez (2014) it was shown that the chase is still a useful

tool in this scenario, and we expand upon that in this thesis. Last but not least, the interest on dealing with schema mappings as first class citizens in the schema mapping management area is very recent (Fagin, Kolaitis, Popa, & Tan, 2005; Madhavan & Halevy, 2003; Fagin, 2007; Kolaitis, 2005; Fagin, Kolaitis, Popa, & Tan, 2008; Arenas, Pérez, & Riveros, 2008; ten Cate & Kolaitis, 2009, 2010; Arenas, Pérez, Reutter, & Riveros, 2009b, 2009a; Arenas et al., 2010; Pérez, 2011; Melnik, Adya, & Bernstein, 2008; Bernstein, Halevy, & Pottinger, 2000; Bernstein, 2003; Melnik, 2004; Melnik, Bernstein, Halevy, & Rahm, 2005). Thus, in the short life of the area, it is natural that the researchers decided to focus in well-known and well-behaved classes of formulas to define mappings.

We do think that schema mappings specified by bidirectional implication formulas deserve a deep investigation, mainly because in many applications they are more natural than mappings defined by unidirectional implications. We also think that in several cases when users map data they are implicitly thinking in bidirectional constraints and not in unidirectional implication formulas. Thus, dealing directly with bidirectional constraints would have a considerable impact in practice, and the research in this area has the potential of laying the foundations for the the next generation data-interoperability tools. This thesis continues with the work started in Arenas, Diéguez, and Pérez (2014) in order to study the fundamental problems that arise in data exchange and schema mapping management when mappings are specified by bidirectional constraints.

Summary of contributions

In this thesis we expand the formal study of bidirectional constraints started in Arenas, Diéguez, and Pérez (2014). Specifically, we study the computational complexity of two fundamental problems in the context of data exchange: checking whether there exists a solution in a given data exchange setting, and answering queries in the data exchange context. In both cases, we provide results for data and combined complexity (Vardi, 1982), and we also distinguish between general dependencies and full dependencies. The latter are a widely used class of dependencies, which will be defined in the next chapter.

In the first part of the thesis, we study the computational complexity of the existence of solutions problem. Regarding this problem, we have the following results:

- Data complexity:
  - PTIME-membership for mappings specified by full dependencies.
  - NP-completeness for mappings specified by general dependencies.
- Combined complexity:
  - Π_2^P-completeness for mappings specified by full dependencies.
  - NEXPTIME-completeness for mappings specified by general dependencies.

In the second part, we study the problem of answering queries in the data exchange context. In this case, besides analyzing data and combined complexity, and considering general and full dependencies, we also distinguish between several query languages with different expressive powers, including non-monotone queries. The results in this part are the following:

- In data complexity, we analyzed the complexity of the problem for five widely used query languages:
  - For mappings specified by full dependencies, the results range from PTIME-membership to coNP-completeness.
  - For mappings specified by general dependencies, the problem is coNP-complete for all the considered languages.
- In combined complexity, we again analyzed the complexity of the problem for six widely used query languages:
  - For mappings specified by full dependencies, the results range from Σ_2^P-completeness to coNEXPTIME-completeness.
  - For mappings specified by general dependencies, the problem is coNEXPTIME-complete for all the considered languages.
- Finally, we proved that the problem is undecidable for unrestricted first-order queries, even when the mapping is specified by full dependencies.

Additionally, in the proofs we introduce some new techniques, like a modified version of the classical chase procedure, showing that some typical tools from the field are still useful in our setting.

Thesis outline and structure

Chapter 2 introduces all necessary notation and concepts, including notions from relational databases, data exchange and bidirectional constraints. Chapter 3 contains the complexity analysis of the EXISTENCEOFSOLUTIONS problem in several scenarios, while Chapter 4 contains it for the CERTAINANSWERS problem. Finally, Chapter 5 presents some final remarks and future lines of research.

2. PRELIMINARIES

We assume some familiarity with first-order logic, computational complexity (Papadimitriou, 1994), database theory (Abiteboul, Hull, & Vianu, 1995), and data exchange (Fagin, Kolaitis, Miller, & Popa, 2005). We also assume that data is represented in the relational model. A *relational schema*, or just *schema*, is a finite set $\{R_1, \ldots, R_n\}$ of relation symbols, each relation having a fixed arity. Given a schema $\mathbf{R}$, we denote by $\mathrm{Inst}\,(\mathbf{R})$ the set of all instances of $\mathbf{R}$.

2.1. Query languages

Information stored in databases is retrieved via queries. In this thesis we focus on queries expressed by using logical formulas, and in particular formulas in fragments of first-order logic (FO). A query Q over a schema $\mathbf{R}$ is a first-order logic formula using $\mathbf{R}$ as vocabulary. Given a query $Q(\bar{x})$, where $\bar{x}$ is the tuple of free variables mentioned in Q, the answer of Q on a particular instance J is the set $Q(J) = \{\bar{t} \mid J \models Q(\bar{t})\}$, where $\models$ denotes the standard satisfaction of FO formulas. A query without free variables is called a *boolean query*, and then we say that $Q(J) = \underline{\text{true}}$ if $J \models Q$, and $Q(J) = \underline{\text{false}}$ if $J \not\models Q$.

Besides FO, the main query languages that we consider in this thesis are the languages of conjunctive queries (CQ), unions of CQ (UCQ), and the languages obtained from them by adding the equality predicate, the inequality predicate and the negation operator (e.g. $\text{UCQ}^{=}$, $\text{CQ}^{\neq}$ and $\text{UCQ}^{\neg}$). Additionally, some restrictions to queries with negation are considered; in particular, CQs with one and two negations, and UCQs with one negation per disjunct (denoted by $\text{CQ}^{1\text{-}\neg}$, $\text{CQ}^{2\text{-}\neg}$ and $\text{UCQ}^{1\text{-}\neg}$ respectively).

We also consider the class of monotone queries, denoted by MON. This class contains all queries Q over a schema $\mathbf{R}$ that satisfy the following property: given two instances J_1, J_2 over $\mathbf{R}$ such that $J_1 \subseteq J_2$, it holds that $Q(J_1) \subseteq Q(J_2)$. Note that this is a semantic class of queries, while the previous were syntactic classes.

2.2. Schema mappings

Schema mappings are used to define a semantic relationship between two schemas. In this thesis, we use a general definition of a schema mapping; given two schemas with no relation symbol in common, $\mathbf{S}$ and $\mathbf{T}$, a schema mapping (or just a mapping) $\mathcal{M}$ between $\mathbf{S}$ and $\mathbf{T}$ is a set of pairs (I, J), where I is an instance of $\mathbf{S}$, and J is an instance of $\mathbf{T}$. That is, a mapping $\mathcal{M}$ is just a subset of $\mathrm{Inst}(\mathbf{S}) \times \mathrm{Inst}(\mathbf{T})$. Given an instance I of $\mathbf{S}$, a mapping $\mathcal{M}$ associates to I a set of *possible solutions for* I, denoted by $\textsc{Sol}_{\mathcal{M}}(I)$, given by the set $\textsc{Sol}_{\mathcal{M}}(I) = \{J \in \mathrm{Inst}(\mathbf{T}) \mid (I, J) \in \mathcal{M}\}$. From now on, assume that we have such schemas $\mathbf{S}$ and $\mathbf{T}$.

2.2.1. Specifying schema mappings

In practice, schema mappings are represented by using logical formulas. Again, we focus on using fragments of FO to specify mappings. Given a set Σ of FO sentences over vocabulary $\mathbf{S} \cup \mathbf{T}$, we say that a mapping $\mathcal{M}$ is *specified* by Σ if for every pair of instances $(I, J) \in \mathrm{Inst}(\mathbf{S}) \times \mathrm{Inst}(\mathbf{T})$ it holds that $(I, J) \in \mathcal{M}$ if and only if $(I, J) \models \Phi$ for every $\Phi \in \Sigma$. For convenience, we write the last statement as $(I, J) \models \Sigma$. Therefore, we usually refer to a mapping as a triple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, and to the set of solutions as $\textsc{Sol}_{\mathcal{M}}(I) = \{J \in \mathrm{Inst}(\mathbf{T}) \mid (I, J) \models \Sigma\}$.

The usual way to specify schema mappings was introduced by Fagin, Kolaitis, Miller, and Popa (2005). A *source-to-target dependency* from $\mathbf{S}$ to $\mathbf{T}$ is a formula of the form

$$\forall \bar{x}(\varphi(\bar{x}) \rightarrow \psi(\bar{x})) \tag{2.1}$$

where $\varphi(\bar{x})$ is an FO-formula over $\mathbf{S}$ and $\psi(\bar{x})$ is an FO-formula over $\mathbf{T}$, both formulas with $\bar{x}$ as its tuple of free variables. We usually drop the outermost universal quantification when specifying these constraints, and thus we only write $\varphi(\bar{x}) \rightarrow \psi(\bar{x})$ for formula (2.1). A mapping defined by source-to-target dependencies is called an st-mapping. Depending on which fragments of FO we use to define formulas $\varphi(\bar{x})$ and $\psi(\bar{x})$, we obtain a wide range of possible fragments of source-to-target dependencies. Given fragments $\mathcal{L}_1$ and

$\mathcal{L}_2$ of FO$^=$, an $\mathcal{L}_1$-TO-$\mathcal{L}_2$ dependency is a formula of the above form in which $\varphi(\bar{x})$ is an $\mathcal{L}_1$-formula over $\mathbf{S}$ and $\psi(\bar{x})$ is an $\mathcal{L}_2$-formula over $\mathbf{T}$. In Fagin, Kolaitis, Miller, and Popa (2005), the language of CQ-TO-CQ dependencies was chosen as the preferred formalism for specifying schema mappings, calling it *source-to-target tuple-generating dependencies (st-tgds)*. In this work, we will also consider *full* $\mathcal{L}$-TO-CQ dependencies, which are formulas in which the target part is a CQ without existential quantifiers.

In this thesis, we study mappings specified by sets of formulas of the following form

$$\forall \bar{x} \big(\varphi(\bar{x}) \ \leftrightarrow \ \psi(\bar{x}) \big), \tag{2.2}$$

where $\varphi(\bar{x})$ is an FO-formula over $\mathbf{S}$ and $\psi(\bar{x})$ is an FO-formula over $\mathbf{T}$, both formulas with $\bar{x}$ as tuple of free variables. We call this formula a *bidirectional constraint*. We also usually drop the outermost universal quantification when specifying these constraints, and thus we only write $\varphi(\bar{x}) \ \leftrightarrow \ \psi(\bar{x})$ for formula (2.2). We say that a sentence Φ is an $\langle \mathcal{L}_1, \mathcal{L}_2 \rangle$-dependency between $\mathbf{S}$ and $\mathbf{T}$, if Φ is a bidirectional constraint of the form (2.2) in which $\varphi(\bar{x})$ is in $\mathcal{L}_1$ and $\psi(\bar{x})$ is in $\mathcal{L}_2$. When the source and target schemas are clear from the context, we will only talk about $\langle \mathcal{L}_1, \mathcal{L}_2 \rangle$-dependencies. For example, consider schemas $\mathbf{S} = \{\texttt{Mother}(\cdot,\cdot), \texttt{Father}(\cdot,\cdot)\}$ and $\mathbf{T} = \{\texttt{Parent}(\cdot,\cdot)\}$. Then the following sentence

$$\big(\ \texttt{Father}(x,y) \vee \texttt{Mother}(x,y) \ \big) \ \leftrightarrow \ \texttt{Parent}(x,y) \tag{2.3}$$

is an example of a $\langle \text{UCQ}, \text{CQ} \rangle$-dependency, which states that x is a parent of y if and only if it is whether her/his father or her/his mother. As before, an $\langle \mathcal{L}, \text{CQ} \rangle$-dependency in which the target part is a CQ without existential quantifiers is called a *full* $\langle \mathcal{L}, \text{CQ} \rangle$-dependency.

2.2.2. Data exchange: Universal solutions and Query answering

In the context of data exchange, the main task is to materialize a target instance for a given source instance. Given a mapping $\mathcal{M}$ specified by FO$^=$-TO-CQ dependencies

and a source instance I, one can define a particular class of solutions in $\text{SOL}_{\mathcal{M}}(I)$ called *universal solutions*. These solutions are the *most general* among all the possible solutions for I under $\mathcal{M}$ (Fagin, Kolaitis, Miller, & Popa, 2005). Moreover, a particular class of universal solutions, called *canonical universal solutions*, can be generated (in polynomial time) by means of the classical *chase procedure* (Maier et al., 1979). We refer the reader to Fagin, Kolaitis, Miller, and Popa (2005) for precise definitions of these notions. We denote by $\textit{chase}_{\Sigma}(I)$ the result of applying the chase procedure to an instance I with a set Σ of dependencies. We call $\text{USOL}_{\mathcal{M}}(I)$ and $\text{CUS}_{\mathcal{M}}(I)$, the set of universal solutions and canonical universal solutions for I under $\mathcal{M}$, respectively. In general, for st-mappings, we have that $\text{CUS}_{\mathcal{M}}(I) \subsetneq \text{USOL}_{\mathcal{M}}(I) \subsetneq \text{SOL}_{\mathcal{M}}(I)$.

Another important aspect in data exchange is how to answer queries over the target schema. The most accepted semantics for query answering in data exchange is the *certain answers* semantics: given a mapping $\mathcal{M}$, a source instance I and a query Q over $\mathbf{T}$, the certain answers of Q with respect to I under $\mathcal{M}$ is the set

$$\text{CERTAIN}_{\mathcal{M}}(Q, I) \quad = \quad \bigcap_{J \in \text{SOL}_{\mathcal{M}}(I)} Q(J).$$

In other words, a tuple $\bar{t} \in \text{CERTAIN}_{\mathcal{M}}(Q, I)$ if $\bar{t} \in Q(J)$ for every solution J for I under $\mathcal{M}$. For boolean queries, we say that $\text{CERTAIN}_{\mathcal{M}}(Q, I) = \underline{\text{true}}$ if $Q(J) = \underline{\text{true}}$ for every solution J, and that $\text{CERTAIN}_{\mathcal{M}}(Q, I) = \underline{\text{false}}$ if there exists a solution J such that $Q(J) = \underline{\text{false}}$.

3. COMPLEXITY OF THE EXISTENCE OF SOLUTIONS PROBLEM

As it was defined in Chapter 2, in a data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ each source instance I has a corresponding set of *solutions*, denoted by $\text{SOL}_{\mathcal{M}}(I)$. This set represents all the possible ways in which data from the source instance can be exchanged to the target according to the setting. Then, a natural question arises: is there *any* way in which we can exchange data? Or, more formally, is $\text{SOL}_{\mathcal{M}}(I) \neq \varnothing$?

In the framework introduced by Fagin et al. (2005), in which Σ consists of st-tgds, this problem is trivial: for every source source instance I one can always find a solution. However, in the presence of bidirectional constraints this does not necessarily hold.

Example 1. Take a simple setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, with Σ consisting of the st-tgds

$$A(x) \rightarrow R(x)$$
$$B(x) \rightarrow R(x)$$

and a source instance $I = \{B(1)\}$. It is clear that the target instance $J = \{R(1)\}$ is a solution for I under $\mathcal{M}$. Now consider the setting $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Delta)$, with Δ consisting of the bidirectional constraints

$$A(x) \leftrightarrow R(x)$$
$$B(x) \leftrightarrow R(x)$$

Note that we only changed the implication. The previous source instance I does not have *any* solution under $\mathcal{M}'$. □

The previous example shows that asking for the existence of solutions for a particular source instance is worth studying under the settings specified by bidirectional constraints, and, in particular, for the fragment of $\langle \text{UCQ}^{=}, \text{CQ} \rangle$-dependencies introduced by Arenas, Diéguez, and Pérez (2014). In the following sections, we study the computational complexity of solving this problem in several scenarios. The results are summarized in table 3.1, showing the references to the theorems where each result is proved.

	full $\langle \text{CQ}, \text{CQ} \rangle$	full $\langle \text{UCQ}^{=}, \text{CQ} \rangle$	general $\langle \text{CQ}, \text{CQ} \rangle$	general $\langle \text{UCQ}^{=}, \text{CQ} \rangle$
data complexity	in PTIME Theorem 2	in PTIME Theorem 2	NP-complete Theorem 1	NP-complete Theorem 1
combined complexity	Π_2^P-complete Theorem 4	Π_2^P-complete Theorem 4	NEXPTIME-complete Theorem 3	NEXPTIME-complete Theorem 3

TABLE 3.1. The complexity of EXISTENCEOFSOLUTIONS for bidirectional constraints.

3.1. Data complexity

In this section we study the data complexity (Vardi, 1982) of the EXISTENCEOFSOLUTIONS problem; this is, when the mapping is considered to be *fixed*, and the input is only the source instance. This is a natural way of studying the complexity of this problem, since in practice is usual that databases are much larger than mapping specifications. Formally, the problem is defined as follows:

Problem:	EXISTENCEOFSOLUTIONS($\mathcal{M}$)
Input:	An instance I over $\mathbf{S}$.
Question:	Is $\text{SOL}_{\mathcal{M}}(I) \neq \varnothing$?

The following result establishes the upper and lower bounds of the complexity of this problem, when we consider mappings specified by $\langle \text{UCQ}^{=}, \text{CQ} \rangle$-dependencies without any further restrictions. We show that the problem is provably intractable, and the lower bound holds even when equalities and disjunctions are banned.

THEOREM 1.

(1) EXISTENCEOFSOLUTIONS($\mathcal{M}$) *is in* NP *for every mapping* $\mathcal{M}$ *specified by a set of* $\langle \text{UCQ}^{=}, \text{CQ} \rangle$*-dependencies.*

(2) There exists a mapping $\mathcal{M}$ *specified by* $\langle \text{CQ}, \text{CQ} \rangle$*-dependencies such that* EXISTENCEOFSOLUTIONS($\mathcal{M}$) *is* NP*-hard.*

PROOF.

(1) Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$ be a mapping, where Δ is a set of $\langle \text{UCQ}^{=}, \text{CQ} \rangle$-dependencies. Consider the following set of $\text{UCQ}^{=}$-TO-CQ st-dependencies:

$$\Delta^{\rightarrow} = \{\forall \bar{x} \, (\varphi(\bar{x}) \rightarrow \exists \bar{y} \psi(\bar{x}, \bar{y})) \mid \forall \bar{x} \, (\varphi(\bar{x}) \leftrightarrow \exists \bar{y} \psi(\bar{x}, \bar{y})) \in \Delta\}.$$

Consider the following Proposition:

PROPOSITION 1. *Given a source instance I, if there exists a solution J for I under $\mathcal{M}$, there exists a solution J^* for I under $\mathcal{M}$ of polynomial size (with respect to I).*

The result follows directly from Proposition 1, since checking that J^* is a solution can be done in polynomial time. Now we prove Proposition 1:
First, note that since J is a solution, it holds that $(I, J) \models \Delta$, and then $(I, J) \models \Delta^{\rightarrow}$. In this proof we use the *solution-aware* chase procedure as it is used in Fuxman, Kolaitis, Miller, and Tan (2006): we chase $(I, \varnothing)$ with $\Delta^{\rightarrow}$ and (I, J), obtaining an instance (I, J^*) such that $J^* \subseteq J$ and $(I, J^*) \models \Delta^{\rightarrow}$. It is clear that $(I, \varnothing) \subseteq (I, J)$, so by the results in Fuxman et al. (2006) it holds that J^* is of polynomial size w.r.t. $(I, \varnothing)$ (since $(I, J) \models \Delta^{\rightarrow}$ and J^* is the result of a solution-aware chase of $(I, \varnothing)$ with $\Delta^{\rightarrow}$ and (I, J)).
Now we need to show that J^* is a solution for I under $\mathcal{M}$, i.e. $(I, J^*) \models \Delta$. By contradiction, suppose that all previous statements hold, but $(I, J^*) \not\models \Delta$. Since $(I, J^*) \models \Delta^{\rightarrow}$, the only way this could happen is if there exists a dependency $\delta = \forall \bar{x}(\varphi_\delta(\bar{x}) \leftrightarrow \exists \bar{y} \psi_\delta(\bar{x}, \bar{y})) \in \Delta$, where $\psi_\delta(\bar{x}, \bar{y})$ is a conjunction of target relations, such that for a tuple of constants $\bar{a}$ it holds that $I \not\models \varphi_\delta(\bar{a})$ and $J^* \models \exists \bar{y} \psi_\delta(\bar{a}, \bar{y})$. Since $J^* \subseteq J$, by monotonicity we know that $\{\bar{b} \mid J^* \models \exists \bar{y} \psi_\delta(\bar{b}, \bar{y})\} \subseteq \{\bar{b} \mid J \models \exists \bar{y} \psi_\delta(\bar{b}, \bar{y})\}$, and therefore it is clear that $J \models \exists \bar{y} \psi_\delta(\bar{a}, \bar{y})$. Given that J is a solution, it holds that $(I, J) \models \Delta$, and in particular $(I, J) \models \delta$, and thus $I \models \varphi_\delta(\bar{a})$, which contradicts our initial supposition.

(2) We will perform a reduction from graph 3-COLORABILITY to the EXISTENCEOFSOLUTIONS($\mathcal{M}$) problem, for a mapping $\mathcal{M}$ built as follows. Let $G = (V, E)$ be a graph with 2 connected components: K_3 and the graph itself. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$ be a data exchange setting such that $\mathbf{S}$ consists of binary relation E and unary relations V, H and $Error$, $\mathbf{T}$ consists of binary relations E' and C, and the dependencies in Δ are the following:

$$V(x) \quad \leftrightarrow \quad \exists u C(x, u) \tag{3.1}$$

$$E(x, y) \quad \leftrightarrow \quad E'(x, y) \tag{3.2}$$

$$H(u) \quad \leftrightarrow \quad \exists x C(x, u) \tag{3.3}$$

$$Error(x) \quad \leftrightarrow \quad \exists y \exists u C(x, u) \wedge C(y, u) \wedge E'(x, y) \tag{3.4}$$

Consider the source instance $I(G) = (V, E, H, Error)$, where $H = \{r, g, b\}$ is a set of three colors, none of which is an element of V, and $Error = \varnothing$. It is clear that $I(G)$ can be constructed in polynomial time from G. We claim that G is 3-colorable if and only if there is a solution for $I(G)$ under $\mathcal{M}$.

($\Rightarrow$) Since G is 3-colorable, there exists a 3-coloration $col(x)$. Without loss of generality, we assume the colors assigned by $col(x)$ are the ones in H. We construct the solution J as follows:

- Start with $J = \varnothing$.
- For each $x \in V$, we add the tuple $C(x, col(x))$ to J.
- For each $(x, y) \in E$, we add the tuple $E'(x, y)$ to J.

Now we show that $(I(G), J) \models \Delta$, showing that $(I(G), J)$ satisfies each rule: (3.1) and (3.2) by construction.

(3.3) [$\rightarrow$] This side of the dependency states that every color is assigned to at least one vertex. As G contains K_3 and is 3-colorable, this holds for G.

[$\leftarrow$] This side of the dependency states that every color that is assigned to a vertex is in H. Since $col\,(x)$ only assigns colors in H, every color mentioned in any tuple $C\,(x, u)$ is in H.
(3.4) Since $Error = \varnothing$, the right-hand side of this dependency must be always false. In other words, for every vertex x, it must not exist an adjacent vertex y such that x and y have the same color assigned in C. As the colors in C come from $col\,(x)$, and G is 3-colorable, this always holds.
($\Leftarrow$) Given J such that $(I\,(G)\,, J) \models \Delta$, we generate a coloration $col\,(x)$ using dependencies (3.1) and (3.3) to choose, for every vertex $x \in V$, a color $c \in H$ such that $C\,(x, c)$. Then, $col\,(x) = c$. We now show that $col\,(x)$ is a 3-coloration: By contradiction, suppose that $col\,(x)$ is not a 3-coloration. Therefore, there exists an edge $(y, z) \in E$ such that $col\,(y) = col\,(z)$. Using dependency (3.2), we know $(y, z) \in E'$. Since the colors in $col\,(x)$ are all obtained from C, the right-hand side of (3.4) holds for vertex y. Since J is a solution for $I\,(G)$, it follows that $y \in Error$, which is a contradiction since $Error = \varnothing$ in $I\,(G)$. $\square$

A usual restriction in data exchange is to consider mappings specified by full dependencies. In this scenario, the EXISTENCEOFSOLUTIONS problem can be efficiently solved.

THEOREM 2. EXISTENCEOFSOLUTIONS($\mathcal{M}$) *is solvable in polynomial time for every mapping $\mathcal{M}$ specified by full $\langle \mathrm{UCQ}^{=}, \mathrm{CQ}\rangle$-dependencies.*

PROOF. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$ be a mapping, where Δ is a set of full $\langle \mathrm{UCQ}^{=}, \mathrm{CQ}\rangle$-dependencies. Consider the following set of full $\mathrm{UCQ}^{=}$-TO-CQ st-dependencies:

$$\Delta^{\rightarrow} = \{\forall \bar{x}\,(\varphi(\bar{x}) \rightarrow \psi(\bar{x})) \mid \forall \bar{x}\,(\varphi(\bar{x}) \leftrightarrow \psi(\bar{x})) \in \Delta\}.$$

In this proof we use the chase procedure as it is used in Fagin, Kolaitis, Miller, and Popa (2005) with the dependencies in $\Delta^{\rightarrow}$. Note that we chase with full $\mathrm{UCQ}^{=}$-TO-CQ

dependencies, and then every chase has the same result. Now call $\mathit{chase}_{\Delta^{\rightarrow}}(I)$ to the chase result for a source instance I. Consider the following Proposition regarding full mappings:

PROPOSITION 2. *Given a source instance I, it has a solution under $\mathcal{M}$ if and only if $\mathit{chase}_{\Delta^{\rightarrow}}(I)$ is a solution for I under $\mathcal{M}$.*

Since this chase procedure uses UCQ$^{=}$-TO-CQ dependencies, it terminates in polynomial time, and then $\mathit{chase}_{\Delta^{\rightarrow}}(I)$ is of polynomial size. Therefore, if Proposition 2 holds, theorem 2 holds: we only need to compute the chase and check whether it is a solution, which can be done in polynomial time. Now we prove Proposition 2:

($\Leftarrow$) This direction is trivial, since $\mathit{chase}_{\Delta^{\rightarrow}}(I)$ is a solution.

($\Rightarrow$) By contradiction, suppose that $\text{SOL}_{\mathcal{M}}(I) \neq \varnothing$ but $J = \mathit{chase}_{\Delta^{\rightarrow}}(I) \notin \text{SOL}_{\mathcal{M}}(I)$. Thus, we have that $(I, J) \not\models \Delta$, and then it must exist a dependency $\delta = \forall \bar{x}(\varphi_{\delta}(\bar{x}) \leftrightarrow \psi_{\delta}(\bar{x})) \in \Delta$, where $\psi_{\delta}(\bar{x})$ is a conjunction of target relations, such that for a tuple of constants $\bar{a}$ one of the following holds:

(1) $I \models \varphi_{\delta}(\bar{a})$ and $J \not\models \psi_{\delta}(\bar{a})$.
(2) $I \not\models \varphi_{\delta}(\bar{a})$ and $J \models \psi_{\delta}(\bar{a})$.

Note that the first scenario is not possible: given that $I \models \varphi_{\delta}(\bar{a})$, the chase procedure would have generated the atoms in $\psi_{\delta}(\bar{a})$. Now, suppose statement 2 holds, and name Q the conjunctive query $\psi_{\delta}(\bar{a})$. Since $J \models \psi_{\delta}(\bar{a})$, there exists a homomorphism $h_1 : I^Q \rightarrow J$, where I^Q is the canonical instance of Q. Consider now the st-mapping $\mathcal{M}^{\rightarrow} = (\mathbf{S}, \mathbf{T}, \Delta^{\rightarrow})$. As $\text{SOL}_{\mathcal{M}}(I) \neq \varnothing$, there exists a target instance J^* that is a solution for I under $\mathcal{M}$, and therefore $(I, J^*) \models \Delta$. Moreover, $(I, J^*) \models \Delta^{\rightarrow}$, and thus $J^* \in \text{SOL}_{\mathcal{M}^{\rightarrow}}(I)$. Now, from the results in Fagin, Kolaitis, Miller, and Popa (2005) we know that $J \in \text{USOL}_{\mathcal{M}^{\rightarrow}}(I)$, and then we know there exists a homomorphism $h_2 : J \rightarrow J^*$. Thus, there exists a homomorphism $h = h_2 \circ h_1 : I^Q \rightarrow J^*$, and then it follows that $J^* \models \psi_{\delta}(\bar{a})$. We also have that $(I, J^*) \models \delta$, and therefore $I \models \varphi_{\delta}(\bar{a})$, which contradicts statement 2. We conclude that it cannot be that $\text{SOL}_{\mathcal{M}}(I) \neq \varnothing$ but $\mathit{chase}_{\Delta^{\rightarrow}}(I) \notin \text{SOL}_{\mathcal{M}}(I)$. □

3.2. Combined complexity

In this section, we study the combined complexity (Vardi, 1982) of the EXISTENCEOFSOLUTIONS problem, when both the mapping and the source instance are part of the input. This is the most well-known notion of complexity, when one does not distinguish between different parts of the input. Formally, the problem is defined as follows:

Problem:	EXISTENCEOFSOLUTIONS
Input:	Mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$ where Δ is a set of bidirectional constraints, and an instance I over $\mathbf{S}$.
Question:	Is $\text{SOL}_{\mathcal{M}}(I) \neq \varnothing$?

Like in the previous section, we begin by analyzing the complexity of the EXISTENCEOFSOLUTIONS problem with unrestricted $\langle \text{UCQ}^{=}, \text{CQ} \rangle$-dependencies. We prove that an exponential blow-up happens with respect to the data complexity. As before, the lower bound holds without equalities nor disjunctions.

THEOREM 3.

(1) EXISTENCEOFSOLUTIONS *is in NEXPTIME for the class of mappings specified by* $\langle \text{UCQ}^{=}, \text{CQ} \rangle$*-dependencies.*

(2) For the class of mappings specified by $\langle \text{CQ}, \text{CQ} \rangle$*-dependencies,* EXISTENCEOFSOLUTIONS *is NEXPTIME-hard.*

PROOF.

(1) It is clear that this problem is in NEXPTIME: we non-deterministically guess a target instance J, and then check if $J \in \text{SOL}_{\mathcal{M}}(I)$, which can be done in exponential time. Notice that a solution J of exponential size with respect to $\mathcal{M}$ and I is guaranteed to exist if $\text{SOL}_{\mathcal{M}}(I) \neq \varnothing$, applying a solution-aware chase. As we already noted, every solution-aware chase sequence is polynomial in the

size of the source instance, as it was shown in Fuxman et al. (2006), and this expression is exponential when the schema is not fixed.

(2) To prove that the problem is NEXPTIME-hard, we will show a reduction from TILING (Papadimitriou, 1994): given a set of tile types $T = \{t_0, \ldots, t_m\}$, relations $H, V \subseteq T \times T$ (which represent horizontal and vertical adjacency constraints between tile types) and an integer n in unary, the problem is to determine if there exists a tiling of a $2^n \times 2^n$ square with tiles in T, starting with the first tile type in the origin that satisfies the constraints imposed by H and V. Formally, a tiling is a function $f : \{0, \ldots, 2^n - 1\} \times \{0, \ldots, 2^n - 1\} \to T$ such that $f(0,0) = t_0$ and for all i, j $(f(i,j), f(i+1,j)) \in H$, and $(f(i,j), f(i,j+1)) \in V$. Given this, we build a data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$, with

$$\mathbf{S} = \{T(\cdot), T_0(\cdot), \ldots, T_m(\cdot), \overline{T_0}(\cdot), \overline{H}(\cdot,\cdot), \overline{V}(\cdot,\cdot), Bin(\cdot,\cdot),$$
$$Zero(\cdot), One(\cdot), A(\cdot), B, Error(\cdot,\cdot), Error'\}$$
$$\mathbf{T} = \{Tile, T_0', \ldots, T_m', \overline{T_0}', \overline{H}', \overline{V}', Bin', Zero', One'\}$$

Intuitively, source relations T, T_i, $\overline{T_0}$, $\overline{H}$ and $\overline{V}$ come directly from the problem (where $\overline{H}$ is the complement of H, the same for V and T_0). Relation A will be used to compute all possible positions in the square in binary, but it also includes positions with 2, in order to overcome some limitations. Bin, $Zero$ and One will be used to distinguish values. Predicate B has arity $2n$ and will be used to represent a special position, which will be necessary to simulate that every tile type is used in the tiling. Predicates $Error$ and $Error'$ will represent errors in the tiling and initial condition, respectively. Relation $Error'$ has arity $2n + 1$. Finally, the target relation $Tile$ has arity $2n + 1$, where the first $2n$ parameters represent a position in the square, and the last parameter is the tile type assigned to it, and the remaining target relations will be copies of the corresponding source relations.

Source instance I will contain the following relation instances:

$$T = T, T_0 = \{t_0\}, \ldots, T_m = \{t_m\}, \overline{T_0} = T \backslash \{t_0\}, \overline{H} = T \times T \backslash H,$$
$$\overline{V} = T \times T \backslash V, Bin = \{0, 1\}, A = \{0, 1, 2\}, Zero = \{0\}, One = \{1\},$$
$$B = \{(\bar{2}, \bar{2})\}, Error = Error' = \varnothing$$

From now on, $\bar{x}$ will be shorthand for the tuple of variables $(x_1, \ldots, x_n)$, and the analogous applies to $\bar{y}$. The set Δ will have the following dependencies:

- Copying dependencies:

$$T_0(x) \leftrightarrow T_0'(x), \ldots, T_m(x) \leftrightarrow T_m'(x) \tag{3.5}$$

$$\overline{T_0}(x) \leftrightarrow \overline{T_0}'(x) \tag{3.6}$$

$$\overline{H}(x, y) \leftrightarrow \overline{H}'(x, y) \tag{3.7}$$

$$\overline{V}(x, y) \leftrightarrow \overline{V}'(x, y) \tag{3.8}$$

$$Bin(x) \leftrightarrow Bin'(x) \tag{3.9}$$

$$Zero(x) \leftrightarrow Zero'(x) \tag{3.10}$$

$$One(x) \leftrightarrow One'(x) \tag{3.11}$$

- A dependency that assigns to each position in the square a tile type, which are computed using predicate A. We also assign tile types to special positions that include 2's.

$$A(x_1) \wedge \ldots \wedge A(x_n) \wedge A(y_1) \wedge \ldots \wedge A(y_n) \leftrightarrow \exists z Tile(\bar{x}, \bar{y}, z) \tag{3.12}$$

- A dependency that ensures that every tile type assigned by equation (3.12) comes from the given set:

$$T(z) \leftrightarrow \exists \bar{x} \exists \bar{y} Tile(\bar{x}, \bar{y}, z) \tag{3.13}$$

Note that this dependency also forces to use each tile type, a restriction that it is not part of the problem. The following dependency solves this:

$$B(\bar{x},\bar{y}) \leftrightarrow \exists z_0 \ldots \exists z_m Tile(\bar{x},\bar{y},z_0) \wedge \ldots \wedge Tile(\bar{x},\bar{y},z_m) \\ \wedge T'_0(z_0) \wedge \ldots \wedge T'_m(z_m) \tag{3.14}$$

This dependency (from left to right) assigns to special position $(\bar{2},\bar{2})$ all tile types. Note that it also says (from right to left) that if a position has all tile types assigned to it, it must be this special one. Note that this is not a problem, since a valid tiling never uses more than one tile per position, and therefore we are not discarding possible tilings.

- A dependency that sets the first position to tile type t_0:

$$Error'(\bar{x},\bar{y},z) \leftrightarrow Tile(\bar{x},\bar{y},z) \wedge \overline{T_0}'(z) \\ \wedge \bigwedge_{i=1}^{n} \big(Zero'(x_i) \wedge Zero'(y_i)\big) \tag{3.15}$$

The intuition behind this dependency is that relation $Error'$ contains erronous positions. Since it is empty in I, the right part must be false in every solution, and therefore position $(\bar{0},\bar{0})$ must contain a tile of type t_0.

Finally, now we explain how to check horizontal and vertical constraints. As it was noted by Kostylev and Reutter (2013), horizontally adjacent positions in the square have the form

$$(w_h 01^{n-k-1}, w_v), (w_h 10^{n-k-1}, w_v) \tag{3.16}$$

where $k \in \{0,\ldots,n-1\}$, w_h is a binary word of length k and w_v is a binary word of length n. Similarly, vertically adjacent positions in the square have the form

$$(w_h, w_v 01^{n-k-1}), (w_h, w_v 10^{n-k-1})$$

where w_h is a binary word of length n and w_v is a binary word of length k. Thus, for each $k \in \{0, \ldots, n-1\}$ we will have dependencies

$$Error(z_1, z_2) \leftrightarrow \exists \bar{p} \exists \bar{q} \exists \bar{r} \exists w_1 \exists w_2 \exists \bar{y} Tile(\bar{p}, w_1, \bar{q}, \bar{y}, z_1)$$
$$\wedge Tile(\bar{p}, w_2, \bar{r}, \bar{y}, z_2) \wedge \overline{H}'(z_1, z_2) \wedge \alpha_k(\bar{p}, \bar{q}, \bar{r}, w_1, w_2, \bar{y}) \tag{3.17}$$

$$Error(z_1, z_2) \leftrightarrow \exists \bar{p} \exists \bar{q} \exists \bar{r} \exists w_1 \exists w_2 \exists \bar{x} Tile(\bar{x}, \bar{p}, w_1, \bar{q}, z_1)$$
$$\wedge Tile(\bar{x}, \bar{p}, w_2, \bar{r}, z_2) \wedge \overline{V}'(z_1, z_2) \wedge \alpha_k(\bar{p}, \bar{q}, \bar{r}, w_1, w_2, \bar{x}) \tag{3.18}$$

that will check horizontal and vertical constraints respectively, where

$$\bar{p} = (p_1, \ldots, p_k),\ \bar{q} = (q_1, \ldots, q_{n-k-1}),\ \bar{r} = (r_1, \ldots, r_{n-k-1}) \text{ and}$$
$$\alpha_k(\bar{p}, \bar{q}, \bar{r}, w_1, w_2, \bar{x}) = \bigwedge_{i=1}^{k} Bin'(p_i) \wedge \bigwedge_{i=1}^{n-k-1} \big(One'(q_i) \wedge Zero'(r_i)\big) \wedge$$
$$Zero'(w_1) \wedge One'(w_2) \wedge \bigwedge_{i=1}^{n} Bin'(x_i).$$

Here the intuition is the same as before, since relation $Error$ is empty in I, and therefore it cannot be that there exist two adjacent positions which tile types are in the complements of the horizontal or vertical relations, respectively. Note that we only check the restrictions for positions in binary. Positions that contain 2's are ignored.

It is clear that we can build $\mathcal{M}$ and I in polynomial time. Now we will show that there exists a $2^n \times 2^n$ tiling that satisfies the constraints if and only if there exists a solution for I under $\mathcal{M}$:

($\Rightarrow$) Given that there exists a tiling, we have the function f, which we will use to build a solution J as follows:

- Start with $J = \varnothing$.
- For each $x \in T_i$, $0 \leq i \leq m$, we add the tuple $T_i'(x)$ to J.
- For each $x \in \overline{T_0}$, we add the tuple $\overline{T_0}'(x)$ to J.
- For each $(x, y) \in \overline{H}$, we add the tuple $\overline{H}'(x, y)$ to J.
- For each $(x, y) \in \overline{V}$, we add the tuple $\overline{V}'(x, y)$ to J.

- For each $x \in Bin$, we add the tuple $Bin'(x)$ to J.
- For each $x \in Zero$, we add the tuple $Zero'(x)$ to J.
- For each $x \in One$, we add the tuple $One'(x)$ to J.
- For each position $(i, j) \in \{0, \ldots, 2^n - 1\}^2$, we add the tuple $Tile(\bar{x}_i, \bar{y}_j, f(i, j))$ to J, where $\bar{x}_i, \bar{y}_j$ are the binary representations of i and j respectively.
- For each $\bar{x}, \bar{y}$ of size n composed by 0, 1 or 2's, and such that one of them mentions at least a 2, we add the tuple $Tile(\bar{x}, \bar{y}, t_0)$.
- For each $t \in T$, we add the tuple $Tile(\bar{2}, \bar{2}, t)$ to J.

Now we show that $(I, J) \models \Delta$, showing that they satisfy each rule:

(3.5), (3.6), (3.7), (3.8), (3.9), (3.10), (3.11) and (3.12) by construction.

(3.13) $[\rightarrow]$ This side of the dependency states that every tile type is assigned to at least one position. Since special position $(\bar{2}, \bar{2})$ is assigned every tile type, this is true.

$[\leftarrow]$ This side of the dependency states that every tile type assigned to a position comes from the original set T, which is true by the way J was built.

(3.14) By construction.

(3.15) Since $Error'$ is empty in I, the right-hand side must be always false. This holds in J, because position $(\bar{0}, \bar{0})$ is the only that satisfies the first atoms, and it has assigned only tile type t_0, and therefore the last atom is not satisfied.

(3.17) Since $Error$ is empty in I, the right-hand side must be always false. This means that it cannot be that two non-compatible tile types (i.e. $(z_1, z_2) \in \overline{H}'$) are assigned to horizontally adjacent positions, which are encoded as it was explained before. To give some more detail, note that any position that mentions value 2 does not satisfy the right-hand side, because all bits are forced to be 1's or 0's. Then, this could only happen for positions represented by binary words. Now, let word $\bar{p}w_1\bar{q}$ represent some number $i \in \{0, \ldots, 2^n - 2\}$. As it was explained before, word $\bar{p}w_2\bar{r}$ would then represent number $i + 1$. Also,

let $\bar{y}$ represent number $j \in \{0, \ldots, 2^n - 1\}$. Since f is a tiling, it holds that $(f(i,j), f(i+1,j)) \in H$, and then $(z_1, z_2) \in H$, which implies that $(z_1, z_2) \notin \overline{H}'$, and therefore the right-hand side is always false.

(3.18) Analogous to (3.17).

In conclusion, given that there exists a tiling, we built a solution for I under $\mathcal{M}$.

($\Leftarrow$) Given J such that $(I, J) \models \Delta$, we generate a tiling f using dependencies (3.12) and (3.13) to choose, for every position $(i,j) \in \{0, \ldots, 2^n - 1\}^2$, a tile type $t_l \in T_l$ (and then $t_l \in T$) such that $Tile(\bar{x}_i, \bar{y}_j, t_l)$, where $\bar{x}_i, \bar{y}_j$ are the binary representations of i and j respectively, and then we make $f(i,j) = t_l$. Without loss of generality, suppose that we choose tile type t_0 for position $(0,0)$ (this is possible because (I, J) must satisfy dependency (3.15)). Now we show that f is a valid tiling:

By contradiction, suppose that f is not a valid tiling. Therefore, there exist i, j such that $(f(i,j), f(i+1,j)) \notin H$ or $(f(i,j), f(i,j+1)) \notin V$. For simplicity suppose that the first statement holds (the other is analogous). Then, $(f(i,j), f(i+1,j)) \in \overline{H}$, and by dependency (3.7) it holds that $(f(i,j), f(i+1,j)) \in \overline{H}'$. By dependency (3.12) (and by how f was built) we know $Tile(\bar{x}_i, \bar{y}_j, f(i,j))$ and $Tile(\bar{x}_{i+1}, \bar{y}_j, f(i+1,j))$ hold. Now, by equation (3.16) we know that it exists some $k \in \{0, \ldots, n-1\}$ such that $\bar{x}_i = p_1 \ldots p_k 01^{n-k-1}$ and $\bar{x}_{i+1} = p_1 \ldots p_k 10^{n-k-1}$, and then for such k we know that J satisfies the right-hand side of dependency (3.17), with $z_1 = f(i,j)$ and $z_2 = f(i+1,j)$. Therefore, since J is a solution, it must be that $I \models Error(f(i,j), f(i+1,j))$, which is a contradiction because $Error = \varnothing$ in I.

□

Regarding full dependencies, even if we restrict to them we are no longer capable of solving the problem efficiently, in terms of combined complexity. Thus, now we prove both upper and lower bounds, showing that the problem becomes complete for a complexity class in the Polynomial Hierarchy, which contains NP, and therefore is believed to

be intractable (see (Papadimitriou, 1994) for details). Again, the lower bound is still true without equalities nor disjunctions.

THEOREM 4.

(1) EXISTENCEOFSOLUTIONS *is in* Π_2^P *for the class of mappings specified by full* $\langle \text{UCQ}^=, \text{CQ} \rangle$*-dependencies.*

(2) For the class of mappings specified by full $\langle \text{CQ}, \text{CQ} \rangle$*-dependencies,* EXISTENCEOFSOLUTIONS *is* Π_2^P*-hard.*

PROOF.

(1) If the dependencies are full, then Proposition 2 holds. Therefore, we can use a non-deterministic machine with an NP oracle that does the following:

- Guess a dependency $\varphi(\bar{x}) \leftrightarrow \psi(\bar{x})$ in Δ, where φ is a $\text{UCQ}^=$ query over $\mathbf{S}$ and ψ is a CQ query over $\mathbf{T}$ without existential quantifiers. Let $\psi(\bar{x}) = R_1(\bar{x}_1) \wedge \ldots \wedge R_n(\bar{x}_n)$, with each $\bar{x}_i \subseteq \bar{x}$.
- Guess n dependencies of the form $\varphi_k(\bar{v}_k) \leftrightarrow \psi_k(\bar{v}_k)$, $1 \leq k \leq n$, in Δ, where φ_k is a $\text{UCQ}^=$ query over $\mathbf{S}$ and ψ_k is a CQ query over $\mathbf{T}$ without existential quantifiers, such that $\psi_k(\bar{v}_k) = \psi_{k1}(\bar{w}_k) \wedge R_k(\bar{y}_k) \wedge \psi_{k2}(\bar{z}_k)$, where ψ_{k1} and ψ_{k2} are (possibly empty) conjunctions of target atoms, and $\bar{w}_k, \bar{y}_k, \bar{z}_k \subseteq \bar{v}_k$.
- Guess tuples of constants $\bar{t}_1, \ldots, \bar{t}_n$ of the same arities as $R_1, \ldots, R_n$ respectively, such that $\bar{t}_1 \cup \ldots \cup \bar{t}_n = \bar{t}$ is of the same arity as φ and has the same pattern on the right-hand side. This is, $\bar{t}_1, \ldots, \bar{t}_n$ should match with $\bar{x}_1, \ldots, \bar{x}_n$.
- Guess tuples of constants $\bar{a}_k$ and $\bar{b}_k$ for each $1 \leq k \leq n$, of the same arity as ψ_{k1} and ψ_{k2} respectively, such that $\bar{a}_k \cup \bar{t}_k \cup \bar{b}_k = \bar{s}_k$ is of the same arity as φ_k and has the same pattern on the right-hand side. This is, $\bar{a}_k$, $\bar{t}_k$ and $\bar{b}_k$ should match $\bar{w}_k$, $\bar{y}_k$ and $\bar{z}_k$ respectively.
- Ask the oracle if $I \models \varphi_1(\bar{s}_1) \wedge \ldots \wedge \varphi_n(\bar{s}_n)$, and then if $I \models \varphi(\bar{t})$.

- If the answers are YES and NO, the machine accepts. Otherwise, it rejects.

In other words, the machine accepts if and only if the target instance produced by the chase is not a solution, which is equivalent to saying that there is no solution for I under $\mathcal{M}$. Thus, the complement of the existence-of-solutions problem is in Σ_2^P, and then the existence-of-solutions problem is in co-$\Sigma_2^P = \Pi_2^P$.

(2) We will show a reduction from Q3SAT (Stockmeyer, 1976; Wrathall, 1976), the problem of determining if a QBF formula of the form $\forall\bar{x}\exists\bar{y}\varphi(\bar{x},\bar{y})$, where φ is in 3-CNF and $\bar{x}$ and $\bar{y}$ form a partition of the variables mentioned in φ, is true. First, suppose that $\bar{x} = (x_1, \ldots, x_n)$. Given such a formula, we build a data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$, with $\mathbf{S}$ consisting of unary relation V and ternary relations N_0, N_1, N_2 and N_3, and $\mathbf{T}$ consisting of n-ary relation R. The set Δ will have two dependencies:

$$V(x_1) \wedge \ldots \wedge V(x_n) \leftrightarrow R(x_1, \ldots, x_n)$$

$$\exists\bar{y}\psi(\bar{x},\bar{y}) \leftrightarrow R(\bar{x})$$

where ψ is a CQ built from φ as it is explained now. First, let $\varphi = C_1 \wedge \ldots \wedge C_m$, where each C_i is a clause with three literals. Without loss of generality, suppose that negated literals are mentioned at the end of the clauses. Each clause will be replaced with a source predicate among $N_0, \ldots, N_3$ depending on how many negated literals it mentions, and using the same variables in the same order. For example, a clause without negated literals will be replaced by N_0 over the same variables mentioned in the clause, while a clause with two negated literals will be replaced by N_2; e.g. clause $p \vee q \vee \neg r$ is replaced by $N_1(p, q, r)$. Then, ψ is the CQ obtained by replacing all clauses in φ following this method and removing the quantifiers.

Finally, the source instance I contains the following tuples for source relations: $V = \{0, 1\}$, $N_0 = \{0, 1\}^3 - \{(0, 0, 0)\}$, $N_1 = \{0, 1\}^3 - \{(0, 0, 1)\}$, $N_2 =$

$\{0,1\}^3 - \{(0,1,1)\}$ and $N_3 = \{0,1\}^3 - \{(1,1,1)\}$. Intuitively, the tuples in $N_0, \ldots, N_2$ are the truth asignments that make true each kind of clause.

It is clear that both $\mathcal{M}$ and I can be built in polynomial time. Now we show that $\forall\bar{x}\exists\bar{y}\varphi(\bar{x},\bar{y})$ is true if and only if there exists a solution for I under $\mathcal{M}$:

($\Rightarrow$) It is clear that $R = \{0,1\}^n$ is a solution for I under $\mathcal{M}$. In the first place, it is the only way the first dependency would be satisfied (since the source query is satisfied with every assignment for $x_1, \ldots, x_n$, because $V(0)$ and $V(1)$ are true in I). Now, the second dependency is satisfied given that $\forall\bar{x}\exists\bar{y}\varphi(\bar{x},\bar{y})$ is true, taking exactly the same assignments, since the N predicates are defined using the propositional logic semantics (note that truth assignments are nothing more than a function from propositional variables to $\{0,1\}$, the same we need to do for the variables in ψ).

($\Leftarrow$) If there exists a solution for I under $\mathcal{M}$, the only possibility is that the solution contains all possible tuples for predicate R, since the source query is satisfied with every assignment for $x_1, \ldots, x_n$ as we mentioned before. Now, if a target instance such that $R = \{0,1\}^n$ is a solution, then $\{\bar{a} \mid I \models \exists\bar{y}\psi(\bar{a},\bar{y})\} = \{0,1\}^n$, and therefore $\forall\bar{x}\exists\bar{y}\varphi(\bar{x},\bar{y})$ is true, since for each possible assignment for the variables in $\bar{x}$, there is an assignment for the variables in $\bar{y}$ that satisfies ψ, and since the definition of N predicates is the same as the semantics of propositional logic, we can take exactly the same assignments for φ.

□

4. COMPLEXITY OF QUERY ANSWERING

Answering queries is of fundamental importance in databases, since it is the way one can obtain the information stored in them. In the context of data exchange, a usual task is to answer queries over the target schema. Then, a natural question is how to effectively answer such queries. This question has been addressed by defining which target instance one should materialize in order to answer queries in a way that is consistent with the data on the source instance.

In the work by Fagin, Kolaitis, Miller, and Popa (2005), the authors showed that one can use a universal solution to obtain the certain answers semantics for positive queries. However, in our setting a universal solution is not even guaranteed to exist.

PROPOSITION 3. *There exists a mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$, where Δ is a set of $\langle \mathrm{UCQ}^{=}, \mathrm{CQ} \rangle$-dependencies, such that there is a source instance I for which there is no universal solution under $\mathcal{M}$.*

PROOF. Take an st-mapping with the following dependencies:

$$A(x) \leftrightarrow \exists y \, (R(y) \wedge S(y))$$

$$B(x) \leftrightarrow R(x)$$

and a source instance $I = \{A(3), B(1), B(2)\}$. Applying the second dependency, it is clear that each solution for I must contain tuples $R(1)$ and $R(2)$. Furthermore, it cannot contain any other tuples in relation R. Then, in any solution we will have that $R = \{1, 2\}$. Given this, it is easy to see that both $J_1 = \{R(1), R(2), S(1)\}$ and $J_2 = \{R(1), R(2), S(2)\}$ are solutions, but there is no homomorphism from one onto the other. Moreover, it is mandatory that any solution contains $S(1)$ or $S(2)$, and therefore it is impossible to have a solution with homomorphisms to all solutions. □

Given this new scenario, it is worth studying how we can answer queries over the target schema in the presence of $\langle \text{UCQ}^{=}, \text{CQ} \rangle$-dependencies. Moreover, as it was shown in Arenas, Diéguez, and Pérez (2014), these dependencies have the potential to specify more tightly which solutions should we consider, and therefore it is interesting to analyze the query answering behaviour of non-monotone queries.

In this chapter, we present the results of the complexity analysis of the CERTAINANSWERS problem in several scenarios, including data and combined complexity analysis, and many query languages. Tables 4.1 and 4.2 summarize the results, showing the references to the theorems where each result is proved.

	CQ	MON	$\text{UCQ}^{1\text{-}\neg}$	$\text{CQ}^{2\text{-}\neg}$	$\text{UCQ}^{\neg}$	FO
full $\langle \text{UCQ}^{=}, \text{CQ} \rangle$	in PTIME Theorem 8	in PTIME Theorem 8	in PTIME Theorem 9	coNP-complete Theorems 10 & 6	coNP-complete Theorems 10 & 6	undecidable Theorem 11
general $\langle \text{UCQ}^{=}, \text{CQ} \rangle$	coNP-complete Theorems 7 & 5	coNP-complete Theorems 7 & 5	coNP-complete Theorems 7 & 6	coNP-complete Theorems 7 & 6	coNP-complete Theorems 7 & 6	undecidable Theorem 11

TABLE 4.1. The data complexity of CERTAINANSWERS under bidirectional constraints.

	CQ	$\text{UCQ}^{\neq}$	$\text{CQ}^{1\text{-}\neg}$	$\text{UCQ}^{1\text{-}\neg}$	$\text{CQ}^{2\text{-}\neg}$	$\text{UCQ}^{\neg}$
full $\langle \text{UCQ}^{=}, \text{CQ} \rangle$	Σ_2^P-complete Theorem 13	Σ_2^P-complete Theorem 13	EXPTIME- -complete Theorem 14	EXPTIME- -complete Theorem 14	coNEXPTIME- -complete Theorems 15 & 12	coNEXPTIME- -complete Theorems 15 & 12
general $\langle \text{UCQ}^{=}, \text{CQ} \rangle$	coNEXPTIME- -complete Theorem 12	coNEXPTIME- -complete Theorem 12	coNEXPTIME- -complete Theorem 12	coNEXPTIME- -complete Theorem 12	coNEXPTIME- -complete Theorem 12	coNEXPTIME- -complete Theorem 12

TABLE 4.2. The combined complexity of CERTAINANSWERS under bidirectional constraints.

4.1. Data Complexity

In this section, we study the data complexity (Vardi, 1982) of the CERTAINANSWERS problem. Similar to Chapter 3, we consider that the mapping and the query are *fixed*, and the input is only the source instance, along with the tuple we wish to check. As we said before, this is a natural way of studying the complexity of this problem, since in practice is usual that databases are much larger than mapping specifications. Formally, the problem is defined as follows:

Problem:	CERTAINANSWERS$(\mathcal{M}, Q)$
Input:	n-tuple $\bar{a}$, and an instance I over $\mathbf{S}$.
Question:	Is $\bar{a}$ in CERTAIN$_{\mathcal{M}}(Q, I)$?

4.1.1. The general case

The following results consider the general case; i.e., when the mapping is specified by unrestricted $\langle \text{UCQ}^{=}, \text{CQ} \rangle$-dependencies. Our first result establishes the upper bound for monotone queries, based on the results in Chapter 3.

THEOREM 5. CERTAINANSWERS$(\mathcal{M}, Q)$ *is in co*NP *for every mapping* $\mathcal{M}$ *specified by* $\langle \text{UCQ}^{=}, \text{CQ} \rangle$*-dependencies and every query* Q *in* MON.

PROOF. From Proposition 1 we know that given a source instance I, if there exists a solution J for I under $\mathcal{M}$, there exists a solution J^* for I under $\mathcal{M}$ of polynomial size (with respect to I). The proof of that Proposition uses the solution-aware chase to obtain such a solution, which also is contained in J. Note that chasing any solution for I will produce another solution that satisfies the previous conditions.

Now, given a n-ary monotone query Q and a n-tuple $\bar{a}$, we want to know if $\bar{a} \in$ CERTAIN$_{\mathcal{M}}(Q, I)$. Therefore, a witness for the complement of this problem is a solution J_w of polynomial size such that $\bar{a} \notin Q(J_w)$. Suppose that there exists some solution J' such that $\bar{a} \notin Q(J')$. If we perform a solution-aware chase, we obtain a solution J_w of polynomial size such that $J_w \subseteq J'$. To conclude, we need to show that $\bar{a} \notin Q(J_w)$, which follows directly from the fact that Q is a monotone query: since $J_w \subseteq J'$, it can't be that $\bar{a} \in Q(J_w)$ but $\bar{a} \notin Q(J')$. Finally, the algorithm is to guess a polynomial-size solution J_w and check if $\bar{a} \notin Q(J_w)$. □

As we mentioned before, given the greater expressive power of $\langle \text{UCQ}^{=}, \text{CQ} \rangle$-dependencies, it is worth studying its query answering capabilities regarding non-monotone queries. In particular, the following result establishes the upper bound

of the problem for queries with negation. This proof uses a custom version of the chase procedure, which is defined in detail in the proof.

THEOREM 6. CERTAINANSWERS$(\mathcal{M}, Q)$ *is in co*NP *for every mapping* $\mathcal{M}$ *specified by* $\langle \text{UCQ}^{=}, \text{CQ} \rangle$*-dependencies, and every query* Q *in* $\text{UCQ}^{\neg}$.

PROOF. Given a query Q as described, we assume it is a boolean query as in Fagin, Kolaitis, Miller, and Popa (2005). We also suppose that $Q = Q_1 \vee Q_2$, where Q_1 is a UCQ query without negation, and Q_2 is a $\text{UCQ}^{\neg}$ with at least one negated atom per disjunct. Each of these disjuncts has the form:

$$\exists \bar{x} \left(\varphi(\bar{x}) \wedge \left(\bigwedge_i \neg R_i(\bar{y}_i) \right) \right), \bar{y}_i \subseteq \bar{x}$$

where φ is a conjunction of atoms and the R_i's are target relations. Thus, it is easy to see that the negation of Q_2 yields a conjunction of a set of *disjunctive tgds* Σ of the form:

$$\forall \bar{x} \left(\varphi(\bar{x}) \rightarrow \left(\bigvee_i R_i(\bar{y}_i) \right) \right)$$

It is clear that $certain(Q, I) = \underline{\text{false}}$ if and only if there exists a solution J for I under $\mathcal{M}$ such that $J \models \Sigma$ and $J \not\models Q_1$. Consider the following Proposition:

PROPOSITION 4. *Given a source instance* I *and a query* Q *as described, if there exists a solution* J *for* I *under* $\mathcal{M}$ *such that* $J \models \Sigma$ *and* $J \not\models Q_1$*, there exists a solution* J^* *of polynomial size with respect to* I *with the same properties.*

Theorem 6 follows directly, since checking the above conditions can be done in polynomial time. We will prove Proposition 4 by using a combination of both chase and disjunctive chase procedures defined in Fagin, Kolaitis, Miller, and Popa (2005), with the solution-aware chase defined in Fuxman et al. (2006), which we conveniently call *Disjunctive Solution-Aware Chase*. As we are only using tgds, the definitions are rather straightforward.

Definition 1 (Disjunctive Solution-Aware Chase Step)**.** Let K be an instance and let d be a disjunctive tgd $\forall \bar{x}\,(\varphi(\bar{x}) \rightarrow (R_1(\bar{y}_1) \vee \ldots \vee R_m(\bar{y}_m)))$. Let K' be an instance that contains K and satisfies d. Denote by d_i the tgds obtained from d of the form $\varphi(\bar{x}) \rightarrow R_i(\bar{y}_i)$ for each $i \in \{1, \ldots, m\}$, which we say are *associated* with d. Note that, because K' satisfies d, K must satisfy at least one of the tgds associated with d. Then, let $D \subseteq \{1, \ldots, m\}$ be the set of the indexes of the tgds associated with d that K' satisfies. Let h be a homomorphism from $\varphi(\bar{x})$ to K such that there are no extensions of h to homomorphisms h'_i from $\varphi(\bar{x}) \wedge R_i(\bar{y}_i)$ to K, for each $i \in \{1, \ldots, m\}$. We say that d can be applied to K with homomorphism h and solution K'. Note that at all the d_i's can be applied to K with homomorphism h and solution K', according to the definition in Fuxman et al. (2006).

For each $j \in D$, let K_j be the result of applying d_j to K with h and solution K', according to the definition in Fuxman et al. (2006). We say that the result of applying d to K with h and solution K' is the set $\{K_j \mid j \in D\}$, and write $K \overset{d,h,K'}{\rightarrow} \{K_j \mid j \in D\}$. $\square$

In addition to the chase steps defined above, we will use solution-aware chase steps as they were defined in Fuxman et al. (2006).

Definition 2 (Disjunctive Solution-Aware Chase)**.** Let Δ be a set of tgds and let Σ be a set of disjunctive tgds. Let K be an instance and K' be an instance that contains K and satisfies $\Delta \cup \Sigma$.

- A *solution-aware chase tree of K with $\Delta \cup \Sigma$ and K'* is a tree such that:
  - the root is K, and
  - for every node K_p in the tree, let $\{K_{p1}, \ldots, K_{pr}\}$ be the set of its children. Then there must exist some dependency d in $\Delta \cup \Sigma$ and homomorphism h such that $K_p \overset{d,h,K'}{\rightarrow} \{K_{p1}, \ldots, K_{pr}\}$
- A *finite disjunctive solution-aware chase of K with $\Delta \cup \Sigma$ and K'* is a finite solution-aware chase tree such that for each leaf K_l, there is no dependency d in

$\Delta \cup \Sigma$ and there is no homomorphism h such that d can be applied to K_l with h and K'. □

It follows directly from the results in Fagin, Kolaitis, Miller, and Popa (2005) and Fuxman et al. (2006) that if the tgds are weakly acyclic, the disjunctive solution-aware chase is finite and polynomial:

PROPOSITION 5. *Let Δ be a set of weakly acyclic tgds, Σ a set of disjunctive tgds, K an instance, and K' an instance such that $K \subseteq K'$ and K' satisfies $\Delta \cup \Sigma$. Then every solution-aware chase tree of K with $\Delta \cup \Sigma$ and K' is finite. Moreover, there exists a polynomial in the size of K that bounds the depth of every such tree.*

PROOF. Let Σ' be the set of all tgds that are associated to some disjunctive tgd in Σ. Let T be a solution-aware chase tree of K with $\Delta \cup \Sigma$ and K'. Then, every path in T that starts in the root is a solution-aware chase sequence of K with K', as it was defined in Fuxman et al. (2006), which uses dependencies in $\Delta \cup \Sigma'$. Moreover, it only uses dependencies in Σ' that K' satisfies. Now, since all the tgds in Σ' are full, they form a weakly acyclic set together with Δ, and then by the results in Fuxman et al. (2006) there exists a polynomial in the size of K that bounds the length of every such path. □

Finally, we now prove Proposition 4. First, note that since J is a solution, it holds that $(I, J) \models \Delta$, and then $(I, J) \models \Delta^{\rightarrow}$. Then, we non-deterministically perform a disjunctive solution-aware chase of $(I, \varnothing)$ with $\Delta^{\rightarrow} \cup \Sigma$ and (I, J), guessing the sequence of dependencies and homomorphisms to be applied as well as the branch we pick at each step, arriving at a leaf J^*. Since $(I, \varnothing) \subseteq (I, J)$ and $(I, J) \models \Delta^{\rightarrow} \cup \Sigma$, by Proposition 5 we know that J^* is of polynomial size. It is easy to see that $J^* \subseteq J$, and then J^* is a solution for I under $\mathcal{M}$ (as it was shown in the proof of theorem 1). Moreover, it holds that $J^* \models \Sigma$, since it is a leaf in the chase tree. Finally, by monotonicity it must be that $J^* \not\models Q_1$, and therefore J^* is the instance we were looking for. □

Following directly from the results in Chapter 3, the CERTAINANSWERS problem in the general case is intractable. Unfortunately, this even holds for boolean conjunctive queries, and dependencies without equalities nor disjunctions.

THEOREM 7. *There exists a mapping $\mathcal{M}$ specified by $\langle \text{CQ}, \text{CQ} \rangle$-dependencies, and a query Q in* CQ*, such that* CERTAINANSWERS$(\mathcal{M}, Q)$ *is co*NP*-hard, even if Q is boolean.*

PROOF. This proof is almost entirely based on Theorem 1's proof. We will again perform a reduction from 3-COLORABILITY.

Recall that we have a graph $G = (V, E)$ with no self-loops, and with 2 connected components: K_3 and the graph itself. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$ be a data exchange setting such that $\mathbf{S}$ consists of binary relation E and unary relations V and H, $\mathbf{T}$ consists of binary relations E' and C, and the dependencies in Δ are the following:

$$V(x) \quad \leftrightarrow \quad \exists u C(x, u) \tag{4.1}$$

$$E(x, y) \quad \leftrightarrow \quad E'(x, y) \tag{4.2}$$

$$H(u) \quad \leftrightarrow \quad \exists x C(x, u) \tag{4.3}$$

Finally, let q be the following query over $\mathbf{T}$:

$$\exists x \exists y \exists u C(x, u) \wedge C(y, u) \wedge E'(x, y)$$

Given a graph G, consider the source instance $I_G = (V, E, H)$, where $H = \{r, g, b\}$ is a set of three colors, none of which is an element of V. It is clear that I_G can be constructed in polynomial time from G. We claim that G is 3-colorable if and only if $certain(q, I_G) = \underline{\text{false}}$. In other words, we need to show that there exists a 3-coloration of G if and only if there exists a solution J for I_G under $\mathcal{M}$ such that $q(J) = \underline{\text{false}}$.

($\Rightarrow$) We build a solution J exactly as in the proof of Theorem 1, where we showed it was indeed a solution. In this case the latter follows immediately, because we need to satisfy less dependencies. Now, it is clear that $q(J) = \underline{\text{false}}$, since otherwise there would

exist adjacent vertices x and y with the same color assigned in predicate C, which cannot be since these were assigned using the 3-coloration from G.

($\Leftarrow$) Similar to the proof of Theorem 1, given J such that $(I_G, J) \models \Delta$ and $q(J) =$ false, we generate a coloration $col\,(x)$ using dependencies (4.1) and (4.3) to choose, for every vertex $x \in V$, a color $c \in H$ such that $C\,(x, c)$. Then, $col\,(x) = c$. We now show that $col\,(x)$ is a 3-coloration:

By contradiction, suppose that $col\,(x)$ is not a 3-coloration. Therefore, there exists an edge $(y, z) \in E$ such that $col\,(y) = col\,(z)$. Using dependency (4.2), we know $(y, z) \in E'$. Since the colors in $col\,(x)$ are all obtained from C, then $q(J) =$ true, taking y, z and $col(y)$ for the existential quantifiers, which contradicts our initial setting. □

4.1.2. The full case

Now we restrict the problem to full dependencies. In the case of monotone queries, the problem can be efficiently solved.

THEOREM 8. CERTAINANSWERS$(\mathcal{M}, Q)$ *can be solved in polynomial time for every mapping* $\mathcal{M}$ *specified by full* $\langle \text{UCQ}^{=}, \text{CQ} \rangle$*-dependencies, and every query* Q *in* MON.

PROOF. Consider the following Proposition:

PROPOSITION 6. *If* Q *is a monotonic query over* $\mathbf{T}$*, then for every source instance* I *such that* $\text{SOL}_{\mathcal{M}}(I) \neq \varnothing$*, it holds that* $certain(Q, I) = Q(chase_{\Delta\rightarrow}(I))$.

Theorem 8 follows directly from the previous Proposition. First, we need a useful Lemma regarding the full scenario:

LEMMA 1. *Given a data exchange setting* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$*, where* Δ *is a set of full* $\text{UCQ}^{=}$ *dependencies, for every source instance* I *such that* $\text{SOL}_{\mathcal{M}}(I) \neq \varnothing$*, it holds that* $chase_{\Delta\rightarrow}(I) \subseteq J$ *for every* $J \in \text{SOL}_{\mathcal{M}}(I)$.

PROOF. By contradiction, suppose that there exists a solution J such that $chase_{\Delta\rightarrow}(I) \not\subseteq J$. Thus, there exists a tuple $R(\bar{a}) \in chase_{\Delta\rightarrow}(I)$ such that $R(\bar{a}) \notin J$.

Since $R(\bar{a})$ is produced by the chase procedure, there is a dependency $\varphi(\bar{x}) \leftrightarrow \psi(\bar{x})$ in Δ, where φ is a UCQ$^{=}$ query and $\psi(\bar{x}) = \psi_1(\bar{w}) \wedge R(\bar{y}) \wedge \psi_2(\bar{z})$ with $\bar{w} \cup \bar{y} \cup \bar{z} = \bar{x}$ and where ψ_1 and ψ_2 are (possibly empty) conjunctions of target atoms, and there exists an assignment $\sigma : Var \rightarrow Const$ such that $I \models \varphi(\sigma(\bar{x}))$ and $\sigma(\bar{y}) = \bar{a}$. Then, as J is a solution, it must be that $J \models \psi(\sigma(\bar{x}))$, and therefore $J \models R(\sigma(\bar{y})) = R(\bar{a})$, which contradicts our initial supposition. □

Now we prove the Proposition, showing the containment if both directions:
($\subseteq$) Given that $\text{SOL}_{\mathcal{M}}(I) \neq \varnothing$, by Proposition 2 we know that *chase*$_{\Delta\rightarrow}(I)$ is a solution, and then for every tuple $\bar{t} \in certain(Q, I)$ it holds that $t \in$ *chase*$_{\Delta\rightarrow}(I)$.
($\supseteq$) Since Q is a monotonic query, by Lemma 1 we know that $\{\bar{t} \mid \bar{t} \in Q(\textit{chase}_{\Delta\rightarrow}(I))\} \subseteq \{\bar{t} \mid \bar{t} \in Q(J)\}$ for every solution J for I under $\mathcal{M}$. Therefore, it holds that $Q(\textit{chase}_{\Delta\rightarrow}(I)) \subseteq certain(Q, I)$. □

Moreover, we have been able to find a polynomial algorithm to answer unions of conjunctive queries with restricted use of negations.

THEOREM 9. CERTAINANSWERS$(\mathcal{M}, Q)$ *can be solved in polynomial time for every mapping* $\mathcal{M}$ *specified by full* $\langle$UCQ$^{=}$, CQ$\rangle$*-dependencies, and every query* Q *in* UCQ$^{\neg}$ *with at most one negated atom per disjunct.*

PROOF. Given a query Q as described, we assume it is a boolean query as in Fagin, Kolaitis, Miller, and Popa (2005). We also suppose that $Q = Q_1 \vee Q_2$, where Q_1 is a UCQ query without negation, and Q_2 is a UCQ$^{\neg}$ with exactly one negated atom per disjunct. Each of these disjuncts has the form:

$$\exists \bar{x}\, (\varphi(\bar{x}) \wedge \neg R(\bar{y})), \bar{y} \subseteq \bar{x}$$

where φ is a conjunction of atomic formulas. Thus, it is easy to see that the negation of Q_2 yields a conjunction of a set of full tgds Σ of the form:

$$\forall \bar{x}\, (\varphi(\bar{x}) \rightarrow R(\bar{y}))$$

Suppose that we are given a source instance I such that it has a solution under $\mathcal{M}$ (otherwise, the certain answers problem is trivial). From Proposition 2 we know that $K = \mathit{chase}_{\Delta\rightarrow}(I)$ can be computed in polynomial time, and is a solution. Moreover, any chase procedure using full tgds on K terminates in polynomial time. Consider now the following Proposition:

PROPOSITION 7. *There exists a solution J for I under $\mathcal{M}$ such that $J \models \Sigma$ if and only if $\mathit{chase}_{\Sigma}(K)$ is a solution for I under $\mathcal{M}$.*

PROOF. ($\Leftarrow$) This direction is trivial, since $\mathit{chase}_{\Sigma}(K) \models \Sigma$ and is a solution.
($\Rightarrow$) By contradiction, suppose that there exists such a solution J, but $\mathit{chase}_{\Sigma}(K)$ is not a solution. Since K is a ground instance and the tgds in Σ are full, it holds that $K \subseteq \mathit{chase}_{\Sigma}(K)$. Thus, there must be a dependency $\varphi(\bar{x}) \leftrightarrow \psi(\bar{x})$ in Δ and a tuple of constants $\bar{a}$ such that $I \not\models \varphi(\bar{a})$, $K \not\models \psi(\bar{a})$ and $\mathit{chase}_{\Sigma}(K) \models \psi(\bar{a})$. For the second statement to hold, there must exist target relations $R_1, \ldots, R_n$ and an assignment $\sigma : Var \rightarrow Const$ such that:

- $\psi(\bar{x}) = R_1(\bar{y}_1) \wedge \ldots \wedge R_n(\bar{y}_n) \wedge \psi'(\bar{z})$, with $\bar{x} = \bar{y}_1 \cup \ldots \cup \bar{y}_n \cup \bar{z}$ and ψ' a conjunction of target atoms (we reorder the atoms without loss of generality),
- $\sigma(\bar{x}) = \bar{a}$,
- $K \models \psi'(\sigma(\bar{z}))$, and
- $K \not\models R_1(\sigma(\bar{y}_1)), \ldots, K \not\models R_n(\sigma(\bar{y}_n))$.

For simplicity consider that $\sigma(\bar{y}_i) = \bar{a}_i$. Now, given that $\mathit{chase}_{\Sigma}(K) \models \psi(\bar{a})$, there must exist tgds in Σ of the form:

$$\alpha_j(\bar{x}_j) \rightarrow R_j(\bar{y}'_j), \text{ with } 1 \leq j \leq n,$$

and an assignment $\sigma' : Var \rightarrow Const$ such that $K \models \alpha(\sigma'(\bar{x}_j))$ and $\sigma'(\bar{y}'_j) = \bar{a}_j$. In other words, there must exist tgds that produce the atoms which made the dependency in Δ false with K.
Now, from Lemma 1 we know that $K \subseteq J$, and then $J \models \alpha(\sigma'(\bar{x}_j))$. Given that $J \models$

Σ, we know that $J \models R_j(\sigma'(\bar{y}'_j)) = R_j(\bar{a}_j) = R_j(\sigma(\bar{y}_j))$, for every $j \in \{1, \ldots, n\}$. Moreover, it also holds that $J \models \psi'(\sigma(\bar{z}))$, and therefore $J \models \psi(\bar{a})$, which contradicts the fact that J is a solution. $\square$

Now we present the algorithm that solves the certain answers problem:

(1) Compute $K = \mathit{chase}_{\Delta\rightarrow}(I)$.

(2) Compute $J = \mathit{chase}_{\Sigma}(K)$.

(3) If J is not a solution for I under $\mathcal{M}$, return $certain(Q, I) = \underline{\text{true}}$.

(4) If J is a solution and satisfies at least one conjunctive query in Q_1, return $certain(Q, I) = \underline{\text{true}}$.

(5) Otherwise, return $certain(Q, I) = \underline{\text{false}}$.

It is clear that this algorithm runs in polynomial time. Finally, we show that it is correct:

- If the algorithm stops in step 3, by Proposition 7 we know that there is no solution that satisfies Σ. Therefore, $certain(Q_2, I) = \underline{\text{true}}$, and then $certain(Q, I) = \underline{\text{true}}$.
- If the algorithm stops in step 4: we know that K is a universal solution, and then it is easy to see that J is a universal solution for target instances that satisfy Σ. From the halting of the algorithm we know $J \models Q_1$, and therefore every solution that satisfies Σ does too. In other words, every solution that does not satisfy Q_2 satisfies Q_1, and then every solution satisfies either Q_1 or Q_2. In conclusion, every solution satisfies Q, and then $certain(Q, I) = \underline{\text{true}}$.
- If the algorithm stops in step 5, it means that $J \not\models Q_1$. Since $J \models \Sigma$, it holds that $J \not\models Q_2$, and therefore $J \not\models Q$. As J is a solution, we conclude that $certain(Q, I) = \underline{\text{false}}$.

$\square$

However, the previous bound is tight, in the sense that allowing just one more negation makes the problem intractable, even without unions.

THEOREM 10. *There exists a mapping $\mathcal{M}$ specified by full $\langle \mathrm{CQ}, \mathrm{CQ} \rangle$-dependencies and a query Q in $\mathrm{CQ}^{\neg}$ with two negated atoms, such that* CERTAINANSWERS$(\mathcal{M}, Q)$ *is co*NP*-complete.*

PROOF. By Theorem 6 we know this problem is in coNP. Now we show it is coNP-hard, performing a reduction from the well-known 3CNF-SAT problem to the complement of the certain answers problem. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$ be a data exchange setting, with

$$\mathbf{S} = \{N_0, N_1, N_2, N_3, Error, Error'\}$$
$$\mathbf{T} = \{N_0', N_1', N_2', N_3', Var, T, F\}$$

and with the set Δ consisting of the following dependencies:

$$N_0(x,y,z) \leftrightarrow N_0'(x,y,z) \wedge Var(x) \wedge Var(y) \wedge Var(z) \tag{4.4}$$
$$N_1(x,y,z) \leftrightarrow N_1'(x,y,z) \wedge Var(x) \wedge Var(y) \wedge Var(z) \tag{4.5}$$
$$N_2(x,y,z) \leftrightarrow N_2'(x,y,z) \wedge Var(x) \wedge Var(y) \wedge Var(z) \tag{4.6}$$
$$N_3(x,y,z) \leftrightarrow N_3'(x,y,z) \wedge Var(x) \wedge Var(y) \wedge Var(z) \tag{4.7}$$
$$Error(x,y,z) \leftrightarrow N_0'(x,y,z) \wedge F(x) \wedge F(y) \wedge F(z) \tag{4.8}$$
$$Error(x,y,z) \leftrightarrow N_1'(x,y,z) \wedge F(x) \wedge F(y) \wedge T(z) \tag{4.9}$$
$$Error(x,y,z) \leftrightarrow N_2'(x,y,z) \wedge F(x) \wedge T(y) \wedge T(z) \tag{4.10}$$
$$Error(x,y,z) \leftrightarrow N_3'(x,y,z) \wedge T(x) \wedge T(y) \wedge T(z) \tag{4.11}$$
$$Error'(x) \leftrightarrow T(x) \wedge F(x) \tag{4.12}$$

Consider also a $\mathrm{CQ}^{\neg}$ query $Q = \exists x(Var(x) \wedge \neg T(x) \wedge \neg F(x))$.

Now, given a 3CNF formula φ, we build a source instance I_φ as follows. We start with $I = \varnothing$. Let $\varphi = C_1 \wedge \ldots \wedge C_n$, where each C_i is a clause with three literals. Without loss of generality, suppose that negated literals are mentioned at the end of the clauses. Thus, there are four possible clause types, depending of how much negated literals they have. These types will be represented by predicates N_0 to N_3, and then we will add a tuple of the corresponding relation to I_φ for each clause. For example, if we have a clause $p \vee q \vee \neg r$, we add tuple $N_1(p, q, r)$ to I_φ. Finally, predicates $Error$ and $Error'$ will be empty in I_φ.

It is clear that I_φ can be built in polynomial time w.r.t the size of φ. Now we claim that φ is satisfiable if and only if $certain(Q, I_\varphi) = \underline{\text{false}}$. Then, we have to show that there is a truth assignment that makes φ true if and only if there exists a solution J for I_φ under $\mathcal{M}$ such that $Q(J) = \underline{\text{false}}$.

($\Rightarrow$) Given that there exists a truth assignment $\sigma : P \to \{0, 1\}$, where P is the set of propositional variables mentioned in φ, such that $\sigma(\varphi) = 1$, we build a solution J as follows:

- Start with $J = \varnothing$.
- For each tuple $(x, y, z) \in N_0$, we add tuples $N_0'(x, y, z)$, $Var(x)$, $Var(y)$ and $Var(z)$ to J.
- We do the same for predicates N_1, N_2 and N_3.
- For each $x \in Var$ such that $\sigma(x) = 1$, we add tuple $T(x)$ to J.
- For each $x \in Var$ such that $\sigma(x) = 0$, we add tuple $F(x)$ to J.

It is easy to see that $Q(J) = \underline{\text{false}}$, since we added each $x \in Var$ to either T or F depending on the value assigned by σ. Note that each $x \in Var$ is indeed assigned a value by σ, since they all come from the propositional variables mentioned in φ. Now we show that $(I_\varphi, J) \models \Delta$, showing it satisfies each rule:

(4.4), (4.5), (4.6) and (4.7) by construction.

(4.8) Since $Error$ is empty in I_φ, the right-hand side must be always false. Given that every tuple $(x, y, z) \in N_0'$ comes from N_0, for each tuple there exists a clause $C_i = x \vee y \vee z$ in φ, for which it must hold that $\sigma(C_i) = 1$. Then, it can't be that the three variables are assigned 0 by σ, and therefore at least one of them is in T and not in F, since they are all in Var. Thus, the right-hand side is false.

(4.9), (4.10) and (4.11) are analogous to (4.8), with the corresponding changes to the values assigned by σ to the variables, and to the memberships to T and F.

(4.12) Since $Error'$ is empty in I_φ, the right-hand side must be always false, which holds in J since we only added each x to T or F, not both.

In conclusion, given that there exists a truth assignment which makes φ true, we built a solution J for I_φ under $\mathcal{M}$ such that $Q(J) = \underline{\text{false}}$.

($\Leftarrow$) Given J such that $(I_\varphi, J) \models \Delta$ and $Q(J) = \underline{\text{false}}$, we generate a truth assignment σ just be checking, for each tuple (x, y, z) in N_0, N_1, N_2 or N_3, whether the constants in the tuple are in T or F, and assigning 1 or 0 to them in σ respectively. Now we show that σ is a truth assignment, and that it makes φ true:

- First we need to prove that σ assigns a value for each propositional variable x in φ. Since they are all mentioned in at least one of the tuples in N_0, N_1, N_2 or N_3, by dependencies (4.4) to (4.7) it must be that $Var(x) \in J$. Now, as $Q(J) = \underline{\text{false}}$, either $T(x)$ or $F(x)$ hold, and then either $\sigma(x) = 1$ or $\sigma(x) = 0$.
- Second, we show that σ assigns a unique value to each propositional variable x in φ. As we mentioned before, it holds that each x is in T or F. By dependency (4.12), it can't be that $x \in T$ and $x \in F$ together, and then $\sigma(x) = 1$ or $\sigma(x) = 0$, not both.
- Finally, we need to show that $\sigma(\varphi) = 1$. By contradiction, assume that $\sigma(\varphi) = 0$. Therefore, there exists a clause $C_i = l_1 \vee l_2 \vee l_3$ in φ such that $\sigma(C_i) = 0$. Let us assume that C_i has no negated atoms, with $l_1 = p$, $l_2 = q$ and $l_3 = r$. Then, it holds that $\sigma(p) = \sigma(q) = \sigma(r) = 0$. Also, there is a tuple $N_0(p, q, r) \in J$.

By dependency (4.4) it holds that $N'_0(p,q,r), Var(p), Var(q), Var(r) \in J$, and then by dependency (4.8) it must be that either $p \notin F$, $q \notin F$ or $r \notin F$. Since $Q(J) = \underline{\text{false}}$, p, q and r must be in T or F, and thus either $p \in T$, $q \in T$ or $r \in T$. Therefore, either $\sigma(p) = 1$, $\sigma(q) = 1$ or $\sigma(r) = 1$, which contradicts our initial suposition that $\sigma(C_i) = 0$. Note that the proof for the other three clause types is analogous, with the corresponding predicates and dependencies.

□

Finally, the following result states that, already in data complexity, the problem becomes undecidable for unrestricted queries in FO$^{=}$, even with $\langle \text{CQ}, \text{CQ} \rangle$-dependencies.

THEOREM 11. *There exists an* FO$^{=}$*-query* Q *and a mapping* $\mathcal{M}$ *specified by full* $\langle \text{CQ}, \text{CQ} \rangle$*-dependencies, such that* CERTAINANSWERS$(\mathcal{M}, Q)$ *is undecidable.*

PROOF. We will perform a reduction from the embedding problem for finite semigroups: given a partial finite algebra $\mathbf{B} = (B, g)$, to determine if $\mathbf{B}$ is embeddable in some finite semigroup; this is, there exists a finite semigroup $\mathbf{A} = (A, f)$ such that $B \subseteq A$ and f is an extension of g. Recall that $\mathbf{A}$ is a finite semigroup if A is a finite nonempty set, and f is a binary associative function over A. This problem was shown to be undecidable by Kolaitis, Panttaja, and Tan (2006), based on the results by Evans (1951, 1953, 1978) and Gurevich (1966).

Based on the proof by Arenas, Barceló, Libkin, and Murlak (2014), we define a mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$ as follows: $\mathbf{S}$ consists of relation U and $\mathbf{T}$ consists of relation V, with arities 3 and 4 respectively. Both encode binary functions using the first three components; the fourth is used to overcome the limitations on solutions imposed by bidirectional constraints. The source instance I will have the tuples coming from g: $I = \{U(a,b,c) \mid g(a,b) = c\}$. The set Δ consists of a single dependency

$$U(x,y,z) \leftrightarrow V(x,y,z,z) \tag{4.13}$$

which simply copies the source instance, but only checks it when the third and fourth components are the same. This lets us complete V with additional tuples, where the third and fourth components are distinct.

Consider now the following formulas:

- $Q_1 := V(x, y, z, w) \wedge V(x, y, z', w') \rightarrow z = z' \wedge w = w'$, that asserts that V encodes a function,
- $Q_2 := V(x, y, u, t_1) \wedge V(y, z, v, t_2) \wedge V(u, z, w, t_3) \rightarrow \exists t_4 V(x, v, w, t_4)$, that asserts that V encodes an associative function, and
- $Q_3 := V(x_1, x_2, x_3, x_4) \wedge V(y_1, y_2, y_3, y_4)$
  $\rightarrow \bigwedge_{1 \leq i,j \leq 3} \exists z_{ij} \exists w_{ij} V(x_i, y_i, z_{ij}, w_{ij})$, that asserts that the function encoded by V is total.

Let $Q' = Q_1 \wedge Q_2 \wedge Q_3$ and $Q = \neg Q'$. It is clear that $certain(Q, I) = \underline{\text{false}}$ iff there exists a solution J for I such that $J \not\models Q$ iff there exists a solution J for I such that $J \models Q'$. Now we show that $\mathbf{B}$ is embeddable in a finite semigroup if and only if $certain(Q, I) = \underline{\text{false}}$:

($\Rightarrow$) Suppose that $\mathbf{B}$ is embeddable in some finite semigroup $\mathbf{A} = (A, f)$. We build a target instance J as follows: for each $a, b \in A$, if $g(a, b)$ is defined we add tuple $V(a, b, f(a, b), f(a, b))$; otherwise, we add tuple $V(a, b, f(a, b), c)$, where c is a constant such that $c \neq f(a, b)$. It is clear that J is a solution, since by definition it contains all the tuples in U, and all other tuples satisfy the dependency trivially (because they do not match with the right-hand side). Now we show that $J \models Q'$:

- $J \models Q_1$: we only add one tuple for each pair $a, b \in A$, so this formula is true in J.
- $J \models Q_2$: since f is associative and considering we added all possible triples $a, b, c \in A$ such that $f(a, b) = c$, then this formula is true in J, given that it only checks it using the frst three components of V.
- $J \models Q_3$: analogous to Q_2.

($\Leftarrow$) Given a solution J such that $J \models Q'$ (and then also $J \models Q_i$), we build a finite semigroup $\mathbf{A} = (A, f)$ where A consists of all constants mentioned in the first three components of some tuple in V, and for each tuple $(a, b, c, d) \in V$ we make $f(a, b) = c$. Now we show that $\mathbf{A}$ is a finite semigroup and $\mathbf{B}$ is embeddable in $\mathbf{A}$:

(1) $\mathbf{A}$ is a finite semigroup:

- A is finite and nonempty: since J is a solution, it is finite and nonempty, and then A is too.
- f is a function: since $J \models Q_1$, for every pair $a, b \in dom(J)$ there is at most one tuple $(a, b, c, d) \in V$, for some values c, d. Thus, we only add one value for $f(a, b)$, and then f is a function.
- f is total: since $J \models Q_3$, for each pair a, b of values mentioned in the first three components of any tuple in V, there exists a tuple $(a, b, c, d) \in V$. Thus, by construction, for every pair $a', b' \in A$ there exists $c' \in A$ such that $f(a', b') = c'$, and then f is a total function.
- f is associative: since $J \models Q_2$, if we ignore the fourth component we have exactly the definition of an associative function encoded in V. Thus, as we only use the first three components to build f, it is clear that f is associative.

(2) $\mathbf{B}$ is embeddable in $\mathbf{A}$: since J is a solution, it holds that $dom(I) \subseteq dom(J)$. Moreover, this even holds if we restrict the domain of J to the first three components, since the dependency in Δ duplicates the third component in the fourth. Thus, it is straightforward that $B \subseteq A$. Also, note that $U \subseteq V$, and then for each pair a, b such that both $U(a, b, c, d)$ and $V(a, b, c', d')$ exist (for some values c, c', d, d') it must be that $c = c'$ and $d = d'$ (since $J \models Q_1$, and then there is only one tuple in V per pair). Therefore, whenever $g(a, b)$ is defined, it holds that $f(a, b) = g(a, b)$.

□

4.2. Combined Complexity

Now we analize the combined complexity (Vardi, 1982) of the CERTAINANSWERS problem, when the mapping, the query and the source instance are part of the input. Formally, the problem is defined as follows:

Problem:	CERTAINANSWERS
Input:	Mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$ with Δ a set of bidirectional constraints, n-ary query Q over $\mathbf{T}$, n-tuple $\bar{a}$, and an instance I over $\mathbf{S}$.
Question:	Is $\bar{a}$ in $\text{CERTAIN}_{\mathcal{M}}(Q, I)$?

4.2.1. The general case

In the general scenario, an exponential blow-up occurs again, due to the mapping and the query being part of the input. The following result establishes both the upper and lower bound in this case. Note that we do not include analysis for monotone queries, since they can no longer be answered in polynomial time when the mapping and the query are not fixed.

THEOREM 12.

(1) CERTAINANSWERS *is in coNEXPTIME for the class of mappings specified by* $\langle \text{UCQ}^{=}, \text{CQ} \rangle$*-dependencies and queries in* $\text{UCQ}^{\neg}$.

(2) For the class of mappings specified by $\langle \text{CQ}, \text{CQ} \rangle$*-dependencies and queries in* CQ, CERTAINANSWERS *is coNEXPTIME-hard.*

PROOF.

(1) In the proof of Theorem 6 we used the fact that every solution-aware chase sequence is polynomial in the size of the source instance, as it was shown in Fuxman et al. (2006). This expression is exponential when the schema is not fixed, and then one can modify Proposition 4, stating that J^* is of exponential size. Since checking that $J^* \models \Sigma$ and $J^* \not\models Q_1$ can be done in exponential time,

it follows that computing the certain answers of $\text{UCQ}^{\neg}$ queries is in coNEXPTIME.

(2) Recall from Theorem 3 that we showed that the existence of solutions problem is NEXPTIME-hard, by building a mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$ and a source instance I from a TILING instance. Let Q be a conjunctive query consisting of the right-hand side of a dependency 3.18 with a fixed k (which we call δ_{19}) with all variables existentially quantified:

$$\begin{aligned} Q :=& \exists z_1 \exists z_2 rhs(\delta_{19}) \\ :=& \exists z_1 \exists z_2 \exists \bar{p}\ \exists \bar{q}\ \exists \bar{r}\ \exists w_1\ \exists w_2\ \exists \bar{x}\ Tile(\bar{x}, \bar{p}, w_1, \bar{q}, z_1) \\ & \wedge Tile(\bar{x}, \bar{p}, w_2, \bar{r}, z_2) \wedge \overline{V}'(z_1, z_2) \wedge \alpha_k(\bar{p}, \bar{q}, \bar{r}, w_1, w_2, \bar{x}) \end{aligned}$$

Let $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Delta')$, where $\Delta' = \Delta - \{\delta_{19}\}$. Note that $\forall z_1 \forall z_2 I \not\models lhs(\delta_{19})$. Then:

$$\begin{aligned} certain_{\mathcal{M}'}(Q, I) = \underline{\text{false}} &\Leftrightarrow \exists J \in \text{SOL}_{\mathcal{M}'}(I) \text{ s.t. } J \not\models Q \\ &\Leftrightarrow \exists J \text{ s.t. } (I, J) \models \Delta' \text{ and } J \not\models \exists z_1 \exists z_2 rhs(\delta_{19}) \\ &\Leftrightarrow \exists J \text{ s.t. } (I, J) \models \Delta' \text{ and } \forall z_1 \forall z_2 J \not\models rhs(\delta_{19}) \\ &\Leftrightarrow \exists J \text{ s.t. } (I, J) \models \Delta' \text{ and } \forall z_1 \forall z_2 (I, J) \models \delta_{19} \\ &\Leftrightarrow \exists J \text{ s.t. } (I, J) \models \Delta \\ &\Leftrightarrow \exists J \in \text{SOL}_{\mathcal{M}}(I) \end{aligned}$$

and thus computing the certain answers of Q is coNEXPTIME-hard.

□

4.2.2. The full case

When we considered full dependencies in the data complexity analysis, the problem was polynomial even for queries with a restricted use of negation. However, this does

not hold when studying the combined complexity of the problem. Thus, a more careful analysis can be carried, considering less expressive query languages. In particular, for queries with inequalities one can exploit the fact that only a polynomial witness is needed to check its satisfaction, and thus we were able to lower the complexity. Specifically, the problem is complete for another class in the polynomial hierarchy, which is the complement of the one considered in Theorem 4. This is consistent with the other results, in which the CERTAINANSWERS problem usually falls in the complement of the complexity class corresponding to the EXISTENCEOFSOLUTIONS problem.

THEOREM 13.

(1) CERTAINANSWERS *is in* Σ_2^P *for the class of mappings specified by full* $\langle \mathrm{UCQ}^{=}, \mathrm{CQ} \rangle$*-dependencies and queries in* $\mathrm{UCQ}^{\neq}$.

(2) For the class of mappings specified by full $\langle \mathrm{CQ}, \mathrm{CQ} \rangle$*-dependencies, and queries in* CQ*,* CERTAINANSWERS *is* Σ_2^P*-hard.*

PROOF.

(1) We will use a non-deterministic turing machine with an NP oracle to decide whether $certain(Q, I) = \underline{\text{true}}$. First, the machine performs the same procedure as the one in Theorem 4 to determine if I has a solution at all. If the answers from the oracle are YES and NO, the machine accepts. Otherwise, we know there exists a solution, so the machine performs the procedure described in the proof of Theorem 8 to check the property. Moreover, we can now use the oracle to ask for the satisfaction of the formulas.

(2) From Theorem 4 we know that the existence of solutions problem is Π_2^P-hard, via a reduction from Q3SAT. Take $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$ and I as defined in the proof, which were built from a formula $\forall \bar{x} \exists \bar{y} \varphi(\bar{x}, \bar{y})$. Consider now a mapping $\mathcal{M}' = (\mathbf{S}', \mathbf{T}', \Delta')$, where $\mathbf{S}' = \mathbf{S} \cup \{A(\cdot, \cdot)\}$, $\mathbf{T}' = \mathbf{T} \cup P(\cdot, \cdot)\}$ and $\Delta' = \Delta \cup \{A(x, y) \leftrightarrow P(x, y)\}$. Let $I' = I \cup \{A(a, b)\}$. Finally, consider a boolean conjunctive query $Q = \exists x P(x, x)$. Note that it still holds that the formula is true

if and only if there exists a solution for I' under $\mathcal{M}'$. It is clear that this query is false in every solution for I', and then

$$certain_{\mathcal{M}'}(Q, I') = \underline{\text{true}} \Leftrightarrow \text{SOL}_{\mathcal{M}'}(I') = \varnothing \Leftrightarrow \forall \bar{x} \exists \bar{y} \varphi(\bar{x}, \bar{y}) \text{ is false.}$$

Therefore, the certain answers problem is Σ_2^P-hard. □

Interestingly, we found that when one knows that the instance has a solution, the problem is slightly easier to solve. This is only valid in the full scenario, since we can use the results of Proposition 2.

PROPOSITION 8.

(1) For the class of mappings $\mathcal{M}$ specified by full $\langle \text{UCQ}^{=}, \text{CQ} \rangle$-dependencies, queries in $\text{UCQ}^{\neq}$, and instances I such that $\text{SOL}_{\mathcal{M}}(I) \neq \varnothing$, CERTAINANSWERS *is in* NP.

(2) For the class of mappings $\mathcal{M}$ specified by full $\langle \text{CQ}, \text{CQ} \rangle$-dependencies, queries in CQ*, and instances I such that $\text{SOL}_{\mathcal{M}}(I) \neq \varnothing$,* CERTAINANSWERS *is* NP*-hard.*

PROOF.

(1) Since we know that I has a solution under $\mathcal{M}$, from proposition 2 we know that $J = chase_{\Delta \rightarrow}(I)$ is a solution, which we can use J to answer monotone queries. This instance J can be of exponential size, but for $\text{UCQ}^{\neq}$ queries we only need a witness of polynomial size; i.e., a polynomial subset J' of J. Therefore, we guess a polynomial sequence of dependencies, each with a tuple of constants, and we chase by checking if the left-hand side is satisfied by I (a problem known to be in NP) and then adding the corresponding tuples to J'. Then, we check if $J' \models Q$ (in NP again). This algorithm can be implemented by a non-deterministic Turing machine running in polynomial time, and decides if $certain(Q, I) = \underline{\text{true}}$.

(2) We will perform a reduction from 3-COLORABILITY, which is very similar to the proof of the combined complexity of conjunctive query evaluation. Given a

graph $G = (V, E)$, we construct a mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$, where $\mathbf{S} = \{R\}$, $\mathbf{T} = \{R'\}$ and Δ has a single copying dependency $R(x, y) \leftrightarrow R'(x, y)$. It is easy to prove that any source instance I has a solution under $\mathcal{M}$, which is unique. Also, as the mapping is full, this unique solution is $\mathit{chase}_{\Delta\rightarrow}(I)$, and then it is isomorphic to I.

Suppose that $V = \{x_1, \ldots, x_n\}$. Consider now a source instance I with $R = \{(r, g), (r, b), (g, r), (g, b), (b, r), (b, g)\}$ and a boolean conjunctive query

$$Q = \exists x_1 \ldots \exists x_n \left(\bigwedge_{(x_i, x_j) \in E} R'(x_i, x_j) \wedge R'(x_j, x_i) \right).$$

Let $\varphi(\bar{x})$ be the non-quantified part of Q. We claim that G is 3-colorable if and only iff $certain(Q, I) = \underline{\text{true}}$, or equivalently, $\mathit{chase}_{\Delta\rightarrow}(I) \models Q$:

($\Rightarrow$) First, note that

$$J = \mathit{chase}_{\Delta\rightarrow}(I) = \{R' = \{(r, g), (r, b), (g, r), (g, b), (b, r), (b, g)\}\}.$$

Given that G is 3-colorable, there exists an assignment $c : V \rightarrow \{r, g, b\}$ such that for every pair $x, y \in V$ where $E(x, y)$, it holds that $c(x) \neq c(y)$. Then, we can construct an assignment σ such that $\sigma(x_i) = c(x_i)$, and therefore $(J, \sigma) \models \varphi(\bar{x})$. This is true because each x_i, x_j has a color assigned, and they are not equal when they are connected. Thus, since J contains all possible distinct pairs, the query is satisfied.

($\Leftarrow$) If $J \models Q$, there exists an asignment $\sigma : \{x_1, \ldots, x_n\} \rightarrow \{r, g, b\}$ such that $(J, \sigma) \models \varphi(\bar{x})$. We construct a coloration c, with $c(x_i) = \sigma(x_i)$. Suppose that it is not a 3-coloration, and then there are vertices $(x_j, x_k) \in E$ such that $c(x_j) = c(x_k)$. This means that $\varphi(\bar{x})$ has a conjunct $R'(x_j, x_k) \wedge R'(x_k, x_j)$, and by the way c was built, is such that $\sigma(x_j) = \sigma(x_k)$. However, there is no pair in J of the form $R'(x, x)$, which contradicts the fact that $J \models Q$. Then, c is a 3-coloration, which concludes our proof.

□

Similar to the results regarding data complexity, when considering queries with negation we found different bounds when restricting the number of negations. When only one negation is allowed (per disjunct in the case of unions of queries), the expected exponential blow-up is confirmed.

THEOREM 14.

(1) CERTAINANSWERS *is in EXPTIME for the class of mappings specified by full* $\langle \mathrm{UCQ}^{=}, \mathrm{CQ} \rangle$*-dependencies and queries in* $\mathrm{UCQ}^{\neg}$ *with at most one negated atom per disjunct.*

(2) For the class of mappings specified by full $\langle \mathrm{CQ}, \mathrm{CQ} \rangle$*-dependencies and queries in* $\mathrm{CQ}^{\neg}$ *with exactly one negated atom,* CERTAINANSWERS *is EXPTIME-hard.*

PROOF.

(1) This result follows directly from Theorem 9. We can use the same algorithm described in its proof, which runs in exponential time in combined complexity (since both chase procedures produce instances of at most exponential size, and then checking satisfaction of FO formulas can also be done in exponential time).

(2) We will perform a reduction from the SGF sirup problem (Kolaitis et al., 2006): given a Datalog program P with one rule and at most one ground fact (i.e. an SGF sirup), a database $\mathbf{D}$ and a ground fact δ, is δ derivable from $\mathbf{D}$ via P? This problem is EXPTIME-complete in combined complexity (Gottlob & Papadimitriou, 2003).

Let the rule be of the form $A(\bar{x}) \leftarrow A_1(\bar{x}_1), \ldots, A_n(\bar{x}_n)$, where each A_i is an extensional predicate or the predicate A. Suppose that A is mentioned in the body of the rule (otherwise the problem is rather easy), and then it is initialized by a ground fact $A(a_1, \ldots, a_k) \leftarrow$. We are also given a database $\mathbf{D}$ and a fact $\delta : A(b_1, \ldots, b_k)$. We build a data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$, a conjunctive query with one negation Q and a source instance I as follows: let the schemas

be

$$\mathbf{S} = \{R_1, \ldots, R_m, R_\delta, A, C, Error\}$$
$$\mathbf{T} = \{R'_1, \ldots, R'_m, R'_\delta, A', C'\}$$

where $R_1, \ldots, R_m$ are the extensional database predicates, each with its corresponding arity; R_δ has arity k and will represent the fact δ; A is of arity $k+1$ and will be used to initialize A; C has arity 1 and will store a constant; and $Error$ has arity $k+1$ and will be used to check the fact. The set Δ consists of the following dependencies:

- Copying dependencies:

$$R_i(\bar{x}) \leftrightarrow R'_i(\bar{x}), 1 \leq i \leq m \tag{4.14}$$
$$A(x_1, \ldots, x_k, x_k) \leftrightarrow A'(x_1, \ldots, x_k, x_k) \tag{4.15}$$
$$R_\delta(\bar{x}) \leftrightarrow R'_\delta(\bar{x}) \tag{4.16}$$
$$C(x) \leftrightarrow C'(x) \tag{4.17}$$

- A dependency that will check if the fact is not derivable:

$$Error(\bar{x}, y) \leftrightarrow R'_\delta(\bar{x}) \wedge A'(\bar{x}, y) \tag{4.18}$$

Let Q be the following conjunctive query with one negated atom:

$$Q = \exists \bar{x}_1 \ldots \exists \bar{x}_n \exists \bar{x} \exists y A'_1(\bar{x}_1) \wedge \ldots \wedge A'_n(\bar{x}_n) \wedge C'(y) \wedge \neg A'(\bar{x}, y)$$

where the tuples follow the same patterns as in the sirup rule, except for A' which has an additional fresh variable, because it has arity $k+1$. Also, the variable y is fresh too.

Finally, I will have the relations $R_1, \ldots, R_m$ coming from $\mathbf{D}$, and the tuples $A(a_1, \ldots, a_k, a_k)$, $R_\delta(b_1, \ldots, b_k)$ and $C(c)$, where c is a constant that does not appear elsewhere in I.

We claim that $certain(Q, I) = \underline{\text{false}}$ if and only if $P \cup \mathbf{D} \not\models \delta$. Note that $certain(Q, I) = \underline{\text{false}}$ iff there exists a solution J such that $J \not\models Q$, or equivalently $J \models \neg Q$. Note that the negation of Q yields a target tgd:

$$\neg Q = A'_1(\bar{x}_1) \wedge \ldots \wedge A'_n(\bar{x}_n) \wedge C'(y) \wedge \rightarrow A'(\bar{x}, y)$$

Therefore, we will prove that $P \cup \mathbf{D} \not\models \delta$ if and only if there exists a solution J such that $J \models \neg Q$:

($\Rightarrow$) We first chase I with $\Delta^{\rightarrow}$, obtaining a target instance J' that is obviously a solution. Then, we chase J' with $\neg Q$, which is a full tgd, and then the chase ends. Call this instance J. Note that this last chase procedure only adds tuples of the form $A'(d_1, \ldots, d_k, c)$, where c is distinct from all d_i (because the d_i's come from $\mathbf{D}$); also, c is the same for all this tuples. Therefore, none of the new tuples matches the right-hand side of dependency 4.15, so (I, J) satisfies 4.15. Now, given that $P \cup \mathbf{D} \not\models \delta$, we know there is no tuple of the form $A'(b_1, \ldots, b_k, b'_k)$ in J', with b'_k any constant, because $A(b_1, \ldots, b_k) \notin \mathbf{D}$. Moreover, tuple $A'(b_1, \ldots, b_k, c)$ is not produced when chasing J' with $\neg Q$. Thus, (I, J) satisfies 4.18, and then J is a solution such that $J \models \neg Q$.

($\Leftarrow$) Let J be a solution as described. Then, there is no tuple of the form $A'(b_1, \ldots, b_k, b'_k)$ in J, with b'_k any constant, according to dependency 4.18. Furthermore, as $J \models \neg Q$, a tuple of this form (or any additional tuples) cannot be obtained by chasing J with $\neg Q$. Now, by Lemma 1 we know that $\mathit{chase}_{\Delta^{\rightarrow}}(I) \subseteq J$, and then one can neither obtain a tuple as described by chasing $\mathit{chase}_{\Delta^{\rightarrow}}(I)$ with $\neg Q$. Thus, by the way I, Δ and Q were constructed, is easy to see that δ cannot be derived from $\mathbf{D}$ via P.

□

Finally, with only two negations the problem is more difficult, as it was in data complexity.

THEOREM 15. *For the class of mappings specified by full* $\langle \mathrm{CQ}, \mathrm{CQ} \rangle$*-dependencies and queries in* $\mathrm{CQ}^{\neg}$ *with two negated atoms,* CERTAINANSWERS *is coNEXPTIME-complete.*

PROOF. From Theorem 12 we know this problem is in coNEXPTIME. To show coNEXPTIME-hardness, we will again perform a reduction from TILING (Papadimitriou, 1994). Given the same definitions as in Theorem 3, we build a data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$, with

$$\begin{aligned}\mathbf{S} = \{&T_0(\cdot), \overline{T_0}(\cdot), \overline{H}(\cdot,\cdot), \overline{V}(\cdot,\cdot), S(\cdot,\cdot), Last(\cdot), A(\cdot), Zero(\cdot), One(\cdot),\\ &Error, Error', Error'', C(\cdot)\}\\ \mathbf{T} = \{&Tile, First, \overline{T_0}', \overline{H}', \overline{V}', S', Last', Zero', One', C'\}\end{aligned}$$

Intuitively, source relations T_0, $\overline{T_0}$, $\overline{H}$ and $\overline{V}$ come directly from the problem (where $\overline{H}$ is the complement of H, the same for V and T_0). Predicate S will store an order of the tile types, while $Last$ will be used to distinguish a special tile that ends the order. Relation A will be used to compute all possible positions in the square in binary, $Zero$ and One will be used to distinguish those values, and C will store a constant. Predicates $Error$, $Error'$ and $Error''$ will represent errors in the tiling, initial condition and last tile, with arities $3n+2$, $2n+1$ and $2n+2$, respectively. Target relation $Tile$ has arity $2n+1$, where the first $2n$ parameters represent a position in the square, and the last parameter is the tile type assigned to it. Finally, predicate $First$ has arity $2n+2$ and will be used to store positions in the target and to choose the tiles. The remaining target relations will be copies of the corresponding source relations.

Source instance I will contain the following relation instances:

$$\begin{gathered}T_0 = \{t_0\}, \overline{T_0} = T \backslash \{t_0\}, \overline{H} = T \times T \backslash H, \overline{V} = T \times T \backslash V,\\ S = \{(t_0, t_1), (t_1, t_2), \ldots, (t_{m-1}, t_m), (t_m, t_{\text{last}})\}, Last = \{t_{\text{last}}\},\\ A = \{0, 1\}, Zero = \{0\}, One = \{1\}, Error = Error' = Error'' = \varnothing,\end{gathered}$$

and $C = \{c\}$, where c is a constant not present elsewhere in I.

From now on, $\bar{x}$ will be shorthand for the tuple of variables $(x_1, \ldots, x_n)$, and the analogous applies to $\bar{y}$. The set Δ will have the following dependencies:

- Copying dependencies:

$$\overline{T_0}(x) \leftrightarrow \overline{T_0}'(x) \tag{4.19}$$

$$\overline{H}(x) \leftrightarrow \overline{H}'(x) \tag{4.20}$$

$$\overline{V}(x) \leftrightarrow \overline{V}'(x) \tag{4.21}$$

$$S(x, y) \leftrightarrow S'(x, y) \tag{4.22}$$

$$Last(x) \leftrightarrow Last'(x) \tag{4.23}$$

$$Zero(x) \leftrightarrow Zero'(x) \tag{4.24}$$

$$One(x) \leftrightarrow One'(x) \tag{4.25}$$

$$C(x) \leftrightarrow C'(x) \tag{4.26}$$

- A dependency that, using predicate A, computes each position in the square storing it in $First$, while stating that t_0 is the first tile type that can be assigned to the position.

$$A(x_1) \wedge \ldots \wedge A(x_n) \wedge A(y_1) \wedge \ldots \wedge A(y_n) \wedge T_0(z) \leftrightarrow First(\bar{x}, \bar{y}, z, z) \tag{4.27}$$

- A dependency that sets the first position to tile type t_0, which is the same as in Theorem 3:

$$Error'(\bar{x}, \bar{y}, z) \leftrightarrow Tile(\bar{x}, \bar{y}, z) \wedge \bigwedge_{i=1}^{n} \big(Zero'(x_i) \wedge Zero'(y_i)\big) \wedge \overline{T_0}'(z) \tag{4.28}$$

- A dependency that ensures every position is assigned a tile type, by stating that the last tile cannot be used in predicate $First$:

$$Error''(\bar{x}, \bar{y}, z, z') \leftrightarrow First(\bar{x}, \bar{y}, z, z') \wedge Last'(z) \tag{4.29}$$

- For each $k \in \{0, \ldots, n-1\}$, two dependencies for checking horizontal and vertical constraints, which are basically the same as in Theorem 3 (refer to its proof for details):

$$Error(\bar{p}, w_1, \bar{q}, \bar{p}, w_2, \bar{r}, \bar{y}, z_1, z_2) \leftrightarrow Tile(\bar{p}, w_1, \bar{q}, \bar{y}, z_1)$$
$$\wedge Tile(\bar{p}, w_2, \bar{r}, \bar{y}, z_2) \wedge \overline{H}'(z_1, z_2) \wedge \alpha_k(w_1, w_2, \bar{q}, \bar{r}) \tag{4.30}$$

$$Error(\bar{p}, w_1, \bar{q}, \bar{p}, w_2, \bar{r}, \bar{x}, z_1, z_2) \leftrightarrow Tile(\bar{x}, \bar{p}, w_1, \bar{q}, z_1)$$
$$\wedge Tile(\bar{x}, \bar{p}, w_2, \bar{r}, z_2) \wedge \overline{V}'(z_1, z_2) \wedge \alpha_k(w_1, w_2, \bar{q}, \bar{r}) \tag{4.31}$$

Finally, we construct a conjunctive query with two negated atoms Q:

$$\exists \bar{x} \exists \bar{y} \exists z_1 \exists z_1' \exists z_2 \exists w \big(First(\bar{x}, \bar{y}, z_1, z_1') \wedge S'(z_1, z_2) \wedge C'(w)$$
$$\wedge \neg Tile(\bar{x}, \bar{y}, z_1) \wedge \neg First(\bar{x}, \bar{y}, z_2, w) \big)$$

Note that $\neg Q$ is a disjunctive tgd, which states that each position must be assigned a tile type in T. It does so recursively, trying to assign the first tile type or any that follows in the order S:

$$First(\bar{x}, \bar{y}, z_1, z_1') \wedge S'(z_1, z_2) \wedge C'(w) \rightarrow Tile(\bar{x}, \bar{y}, z_1) \vee First(\bar{x}, \bar{y}, z_2, w) \tag{4.32}$$

It is clear that we can build $\mathcal{M}$, I and Q in polynomial time. Recall that $certain(Q, I) = \underline{\text{false}}$ iff there exists a solution J such that $J \not\models Q$, or equivalently, $J \models \neg Q$. Now we will show that there exists a $2^n \times 2^n$ tiling that satisfies the constraints if and only if there exists a solution J for I under $\mathcal{M}$ such that $J \models \neg Q$:

($\Rightarrow$) Given that there exists a tiling, we have the function f, which we will use to build a solution J as follows:

- Start with $J = \varnothing$.

- We copy relations $\overline{T_0}$, $\overline{H}$, $\overline{V}$, S, $Last$, $Zero$, One and C from I to their corresponding relations in J.
- For each position $(i, j) \in \{0, \ldots, 2^n - 1\}^2$, where $f(i, j) = t_l$ for some $l \in \{0, \ldots, m\}$, we add tuples $First(\bar{x}_i, \bar{y}_j, t_0, t_0)$ and $Tile(\bar{x}_i, \bar{y}_j, t_l)$ to J, where $\bar{x}_i, \bar{y}_j$ are the binary representations of i and j respectively. We also add tuples $First(\bar{x}_i, \bar{y}_j, t_p, c)$ for each $p \in \{1, \ldots, l\}$.

Now we show that $(I, J) \models \Delta$ and $J \models \neg Q$:

(4.19), (4.20), (4.21), (4.22), (4.23), (4.24), (4.25) and (4.26) by construction.

(4.27) $[\rightarrow]$ By construction.

$[\leftarrow]$ This side of the dependency states that every tuple in $First$ with its last two variables equal must consist of a position in the square and tile type t_0 in the last two variables. This is true given that all tuples in $First$ that mention another tile type do not have the last two variables with the same value, since they mention constant c in the last position, which is distinct from every other constant in I, and in particular from every t_i.

(4.28), (4.30) and (4.31): refer to the proof for Theorem 3.

(4.29) This dependency states that there cannot be a tuple in $First$ with tile type t_{last} second-to-last, which is true by construction.

(4.32) For each position (i, j) let $f(i, j) = t_l$. Regarding the left-hand side, it holds that $First(\bar{x}_i, \bar{x}_j, t, t') \in J$ for each $t \in \{t_0, \ldots, t_l\}$ and some constant t'. Then, the tgd is satisfied for each $t \in \{t_0, \ldots, t_{l-1}\}$ by the second disjunct, and for t_l is satisfied by the first disjunct.

In conclusion, given that there exists a tiling, we built a solution J for I under $\mathcal{M}$ such that $J \models \neg Q$.

$(\Leftarrow)$ Given J such that $(I, J) \models \Delta$ and $J \models \neg Q$, we generate a tiling f using tgd (4.32) to choose, for every position (i, j), a tile type t_l mentioned in S' such that $Tile(\bar{x}_i, \bar{y}_j, t_l)$, making $f(i, j) = t_l$. Note that we can always choose such t_l, since by

dependency (4.29) we cannot use t_{last} in $First$, and then $Tile$ must hold at some point. Moreover, this also ensures that every t_l comes from T. Without loss of generality, suppose that we choose tile type t_0 for position $(0, 0)$ (this is possible because (I, J) must satisfy dependency (4.28)). Now we show that f is a valid tiling:

By contradiction, suppose that f is not a valid tiling. Therefore, there exist i, j such that $(f(i, j), f(i + 1, j)) \notin H$ or $(f(i, j), f(i, j + 1)) \notin V$. For simplicity suppose that the first statement holds (the other is analogous). Then, $(f(i, j), f(i + 1, j)) \in \overline{H}$, and by dependency (4.20) it holds that $(f(i, j), f(i+1, j)) \in \overline{H}'$. By tgd (4.32) (and by how f was built) we know $Tile(\bar{x}_i, \bar{y}_j, f(i, j))$ and $Tile(\bar{x}_{i+1}, \bar{y}_j, f(i + 1, j))$ hold. Now, by equation (3.16) we know that it exists some $k \in \{0, \ldots, n - 1\}$ such that $\bar{x}_i = p_1 \ldots p_k 01^{n-k-1}$ and $\bar{x}_{i+1} = p_1 \ldots p_k 10^{n-k-1}$, and then for such k we know that J satisfies the right-hand side of dependency (4.30), with $z_1 = f(i, j)$ and $z_2 = f(i + 1, j)$. Therefore, since J is a solution, it must be that $I \models Error(\bar{x}_i, \bar{x}_{i+1}, \bar{y}_j, f(i, j), f(i + 1, j))$, which is a contradiction because $Error = \varnothing$ in I. □

5. CONCLUDING REMARKS

In this work we have continued the formal study of bidirectional constraints, which was started by Arenas, Diéguez, and Pérez (2014). In particular, we have thoroughly analyzed the computational complexity of two fundamental problems in data exchange: EXISTENCEOFSOLUTIONS and CERTAINANSWERS. We have done this in both data and combined complexity, and in the case of query answering, for several query languages including non-monotone fragments. The results show that bidirectional constraints are an interesting field of research in database theory, which was not studied from a theoretical point of view prior to Arenas, Diéguez, and Pérez (2014) and this thesis. We should mention that there has been some related work driven from practical motivations, most notably Melnik et al. (2008) and Bernstein, Jacob, Pérez, Rull, and Terwilliger (2013), where bidirectional constraints based on project-select relational algebra expressions are considered in an object-to-relational mapping system. Nevertheless, to the best of our knowledge, both Arenas, Diéguez, and Pérez (2014) and this thesis present the first theoretical results on the fundamental properties of bidirectional constraints.

A natural extension of this work would be the definition of tractable fragments of bidirectional constraints, restricting the query languages involved to obtain lower complexity bounds. Moreover, restricting the dependencies to GAV constraints (Levy, Mendelzon, Sagiv, & Srivastava, 1995; Halevy, 2001; Lenzerini, 2002) would probably be a succesful line of research.

Another possibility for future research arises from the results by Arenas, Diéguez, and Pérez (2014). Bidirectional constraints have a greater expressive power in terms of better specifying the desired solutions in data exchange when compared to st-tgds, as it was shown by Arenas, Diéguez, and Pérez (2014) for full dependencies, and then several lines of research are possible regarding this, including the analysis of general dependencies, and the definition of alternative semantics for mappings specified by st-tgds using these constraints. Interesting semantics have been proposed by Libkin (2006) and Hernich

(2013), and we believe a new semantics defined in terms of bidirectional constraints would compare favourably to them.

Besides the motivation that comes from defining more strictly what are the possible target instances in data exchange, mappings specified by bidirectional constraints have several other applications. Most notably, these specifications are expressive enough to define how source constraints should be *transformed* into target constraints. If one has a source schema with, for example, key constraints, and creates a new schema and a data exchange setting to migrate the data, it is natural to expect the source key constraints to be somehow reflected in the new schema. This issue is closely related with the recently raised topic of exchanging (or transforming) knowledge bases (Arenas et al., 2013). Mappings specified by st-tgds are not expressive enough to describe these type of transformations. We argue that bidirectional constraints are a good formalism to study these complex transformation scenarios, and it would be very interesting to explore these possibilities in future work. Another motivation comes from schema mapping management, where the inputs for schema mapping operators such as the *merge*, *extract* and *diff* operators are naturally defined using bidirectional mappings (Bernstein & Melnik, 2007; Arenas et al., 2010). This is also part of our future work.

References

Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of databases*. Addison-Wesley.

Afrati, F. N., & Kolaitis, P. G. (2008). Answering aggregate queries in data exchange. In M. Lenzerini & D. Lembo (Eds.), *Pods* (p. 129-138). ACM.

Arenas, M., Barceló, P., Libkin, L., & Murlak, F. (2014). *Foundations of data exchange*. Cambridge University Press. Retrieved from http://www.cambridge.org/9781107016163

Arenas, M., Diéguez, G., & Pérez, J. (2014). Expressiveness and complexity of bidirectional constraints for data exchange. In G. Gottlob & J. Pérez (Eds.), *Proceedings of the 8th alberto mendelzon workshop on foundations of data management, cartagena de indias, colombia, june 4-6, 2014.* (Vol. 1189). CEUR-WS.org. Retrieved from http://ceur-ws.org/Vol-1189/paper_15.pdf

Arenas, M., Pérez, J., & Reutter, J. L. (2011). Data exchange beyond complete data. In M. Lenzerini & T. Schwentick (Eds.), *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems, PODS 2011, june 12-16, 2011, athens, greece* (pp. 83–94). ACM. Retrieved from http://doi.acm.org/10.1145/1989284.1989293 doi: 10.1145/1989284.1989293

Arenas, M., Pérez, J., & Reutter, J. L. (2013). Data exchange beyond complete data. *J. ACM*, *60*(4), 28.

Arenas, M., Pérez, J., Reutter, J. L., & Riveros, C. (2009a). Composition and inversion of schema mappings. *SIGMOD Record*, *38*(3), 17–28. Retrieved from http://doi.acm.org/10.1145/1815933.1815938 doi: 10.1145/1815933.1815938

Arenas, M., Pérez, J., Reutter, J. L., & Riveros, C. (2009b). Inverting schema mappings: Bridging the gap between theory and practice. *PVLDB*, *2*(1), 1018–1029. Retrieved from http://www.vldb.org/pvldb/2/vldb09-775.pdf

Arenas, M., Pérez, J., Reutter, J. L., & Riveros, C. (2010). Foundations of schema mapping management. In J. Paredaens & D. V. Gucht (Eds.), *Pods* (p. 227-238). ACM.

Arenas, M., Pérez, J., & Riveros, C. (2008). The recovery of a schema mapping: bringing exchanged data back. In M. Lenzerini & D. Lembo (Eds.), *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems, PODS 2008, june 9-11, 2008, vancouver, bc, canada* (pp. 13–22). ACM. Retrieved from http://doi.acm.org/10.1145/1376916.1376920 doi: 10.1145/1376916.1376920

Arenas, M., Pérez, J., & Riveros, C. (2009). The recovery of a schema mapping: Bringing exchanged data back. *ACM Trans. Database Syst.*, *34*(4).

Bernstein, P. A. (2003). Applying model management to classical meta data problems. In *CIDR*. Retrieved from http://www-db.cs.wisc.edu/cidr/cidr2003/program/p19.pdf

Bernstein, P. A., Green, T. J., Melnik, S., & Nash, A. (2006). Implementing mapping composition. In U. Dayal et al. (Eds.), *Proceedings of the 32nd international conference on very large data bases, seoul, korea, september 12-15, 2006* (pp. 55–66). ACM. Retrieved from http://www.vldb.org/conf/2006/p55-bernstein.pdf

Bernstein, P. A., Halevy, A. Y., & Pottinger, R. (2000). A vision of management of complex models. *SIGMOD Record*, *29*(4), 55–63. Retrieved from http://doi.acm.org/10.1145/369275.369289 doi: 10.1145/369275.369289

Bernstein, P. A., Jacob, M., Pérez, J., Rull, G., & Terwilliger, J. F. (2013). Incremental mapping compilation in an object-to-relational mapping system. In K. A. Ross, D. Srivastava, & D. Papadias (Eds.), *Sigmod conference* (p. 1269-1280). ACM.

Bernstein, P. A., & Melnik, S. (2007). Model management 2.0: manipulating richer mappings. In C. Y. Chan, B. C. Ooi, & A. Zhou (Eds.), *Sigmod conference* (p. 1-12). ACM.

De Giacomo, G., Lembo, D., Lenzerini, M., & Rosati, R. (2007). On reconciling data exchange, data integration, and peer data management. In L. Libkin (Ed.), *Pods* (p. 133-142). ACM.

Evans, T. (1951). The word problem for abstract algebras. *Journal of the London Mathematical Society*, *26*, 64–71.

Evans, T. (1953). Embeddability and the word problem. *Journal of the London Mathematical Society*, *28*, 76–80.

Evans, T. (1978). Word problems. *Bulletin of the American Mathematical Society*, *84*(5), 789–802.

Fagin, R. (2007). Inverting schema mappings. *ACM Trans. Database Syst.*, *32*(4). Retrieved from http://doi.acm.org/10.1145/1292609.1292615 doi: 10.1145/1292609.1292615

Fagin, R., Kolaitis, P. G., Miller, R. J., & Popa, L. (2003). Data exchange: Semantics and query answering. In D. Calvanese, M. Lenzerini, & R. Motwani (Eds.), *Database theory - ICDT 2003, 9th international conference, siena, italy, january 8-10, 2003, proceedings* (Vol. 2572, pp. 207–224). Springer. Retrieved from http://dx.doi.org/10.1007/3-540-36285-1_14 doi: 10.1007/3-540-36285-1_14

Fagin, R., Kolaitis, P. G., Miller, R. J., & Popa, L. (2005). Data exchange: semantics and query answering. *Theor. Comput. Sci.*, *336*(1), 89-124.

Fagin, R., Kolaitis, P. G., & Popa, L. (2003). Data exchange: getting to the core. In F. Neven, C. Beeri, & T. Milo (Eds.), *Proceedings of the twenty-second ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, june 9-12, 2003, san diego, ca, USA* (pp. 90–101). ACM. Retrieved from http://doi.acm.org/10.1145/773153.773163 doi: 10.1145/773153.773163

Fagin, R., Kolaitis, P. G., Popa, L., & Tan, W. C. (2005). Composing schema mappings: Second-order dependencies to the rescue. *ACM Trans. Database Syst.*, *30*(4), 994–1055. Retrieved from http://doi.acm.org/10.1145/1114244.1114249 doi: 10.1145/1114244.1114249

Fagin, R., Kolaitis, P. G., Popa, L., & Tan, W. C. (2008). Quasi-inverses of schema mappings. *ACM Trans. Database Syst.*, *33*(2). Retrieved from http://doi.acm.org/10.1145/1366102.1366108 doi: 10.1145/1366102.1366108

Fuxman, A., Kolaitis, P. G., Miller, R. J., & Tan, W. C. (2006). Peer data exchange. *ACM Trans. Database Syst.*, *31*(4), 1454-1498.

Gottlob, G., & Nash, A. (2006). Data exchange: computing cores in polynomial time. In S. Vansummeren (Ed.), *Proceedings of the twenty-fifth ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, june 26-28, 2006, chicago, illinois, USA* (pp. 40–49). ACM. Retrieved from http://doi.acm.org/10.1145/1142351.1142358 doi: 10.1145/1142351.1142358

Gottlob, G., & Papadimitriou, C. H. (2003). On the complexity of single-rule datalog queries. *Inf. Comput.*, *183*(1), 104–122. Retrieved from http://dx.doi.org/10.1016/S0890-5401(03)00012-9 doi: 10.1016/S0890-5401(03)00012-9

Gurevich, Y. (1966). The word problem for certain classes of semigroups. *Algebra and Logic*, *5*, 25–35.

Haas, L. M., Hernández, M. A., Ho, H., Popa, L., & Roth, M. (2005). Clio grows up: from research prototype to industrial tool. In F. Ozcan (Ed.), *Proceedings of the ACM SIGMOD international conference on management of data, baltimore, maryland, usa, june 14-16, 2005* (pp. 805–810). ACM. Retrieved from http://doi.acm.org/10.1145/1066157.1066252 doi: 10.1145/1066157.1066252

Halevy, A. Y. (2001). Answering queries using views: A survey. *VLDB J.*, *10*(4), 270–294. Retrieved from http://dx.doi.org/10.1007/s007780100054 doi: 10.1007/s007780100054

Hernández, M. A., Popa, L., Velegrakis, Y., Miller, R. J., Naumann, F., & Ho, C. (2002). Mapping XML and relational schemas with clio. In R. Agrawal & K. R. Dittrich (Eds.), *Proceedings of the 18th international conference on data engineering, san jose, ca, usa, february 26 - march 1, 2002* (pp. 498–499). IEEE Computer Society. Retrieved from http://dx.doi.org/10.1109/ICDE.2002.994768 doi: 10.1109/ICDE.2002.994768

Hernich, A. (2013). Semantics for non-monotone queries in data exchange and data integration. In P. G. Kolaitis, M. Lenzerini, & N. Schweikardt (Eds.), *Data exchange, information, and streams* (Vol. 5, p. 161-184). Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Kolaitis, P. G. (2005). Schema mappings, data exchange, and metadata management. In C. Li (Ed.), *Proceedings of the twenty-fourth ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, june 13-15, 2005, baltimore, maryland, USA* (pp. 61–75). ACM. Retrieved from http://doi.acm.org/10.1145/1065167.1065176 doi: 10.1145/1065167.1065176

Kolaitis, P. G., Panttaja, J., & Tan, W. C. (2006). The complexity of data exchange. In S. Vansummeren (Ed.), *Pods* (pp. 30–39). ACM. Retrieved from http://doi.acm.org/10.1145/1142351.1142357 doi: 10.1145/1142351.1142357

Kostylev, E. V., & Reutter, J. L. (2013). Answering counting aggregate queries over ontologies of the dl-lite family. In M. desJardins & M. L. Littman (Eds.), *Aaai.* AAAI Press.

Lenzerini, M. (2002). Data integration: A theoretical perspective. In L. Popa, S. Abiteboul, & P. G. Kolaitis (Eds.), *Pods* (p. 233-246). ACM.

Lenzerini, M., & Lembo, D. (Eds.). (2008). *Proceedings of the twenty-seventh acm sigmod-sigact-sigart symposium on principles of database systems, pods 2008, june 9-11, 2008, vancouver, bc, canada.* ACM.

Levy, A. Y., Mendelzon, A. O., Sagiv, Y., & Srivastava, D. (1995). Answering queries using views. In M. Yannakakis (Ed.), *Proceedings of the fourteenth ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, may 22-25, 1995, san jose, california, USA* (pp. 95–104). ACM Press. Retrieved from http://doi.acm.org/10.1145/212433.220198 doi: 10.1145/212433.220198

Libkin, L. (2006). Data exchange and incomplete information. In S. Vansummeren (Ed.), *Pods* (p. 60-69). ACM.

Madhavan, J., & Halevy, A. Y. (2003). Composing mappings among data sources. In *VLDB* (pp. 572–583). Retrieved from http://www.vldb.org/conf/2003/papers/S18P01.pdf

Maier, D., Mendelzon, A. O., & Sagiv, Y. (1979). Testing implications of data dependencies. *ACM Trans. Database Syst.*, *4*(4), 455–469. Retrieved from http://doi.acm.org/10.1145/320107.320115 doi: 10.1145/320107.320115

Melnik, S. (2004). *Generic model management: Concepts and algorithms* (Vol. 2967). Springer. Retrieved from http://dx.doi.org/10.1007/b97859 doi: 10.1007/b97859

Melnik, S., Adya, A., & Bernstein, P. A. (2008). Compiling mappings to bridge applications and databases. *ACM Trans. Database Syst.*, *33*(4).

Melnik, S., Bernstein, P. A., Halevy, A. Y., & Rahm, E. (2005). Supporting executable mappings in model management. In F. Ozcan (Ed.), *Proceedings of the ACM SIGMOD international conference on management of data, baltimore, maryland, usa, june 14-16, 2005* (pp. 167–178). ACM. Retrieved from http://doi.acm.org/10.1145/1066157.1066177 doi: 10.1145/1066157.1066177

Ozcan, F. (Ed.). (2005). *Proceedings of the ACM SIGMOD international conference on management of data, baltimore, maryland, usa, june 14-16, 2005.* ACM.

Papadimitriou, C. H. (1994). *Computational complexity*. Addison-Wesley.

Pérez, J. (2011). *Schema mapping management in data exchange systems* (Unpublished doctoral dissertation). Escuela de Ingeniería, Pontificia Universidad Católica de Chile.

Stockmeyer, L. J. (1976). The polynomial-time hierarchy. *Theor. Comput. Sci.*, *3*(1), 1-22.

ten Cate, B., & Kolaitis, P. G. (2009). Structural characterizations of schema-mapping languages. In R. Fagin (Ed.), *Database theory - ICDT 2009, 12th international conference, st. petersburg, russia, march 23-25, 2009, proceedings* (Vol. 361, pp. 63–72). ACM. Retrieved from http://doi.acm.org/10.1145/1514894.1514903 doi: 10.1145/1514894.1514903

ten Cate, B., & Kolaitis, P. G. (2010). Structural characterizations of schema-mapping languages. *Commun. ACM*, *53*(1), 101–110. Retrieved from http://doi.acm.org/10.1145/1629175.1629201 doi: 10.1145/1629175.1629201

Vansummeren, S. (Ed.). (2006). *Proceedings of the twenty-fifth acm sigact-sigmod-sigart symposium on principles of database systems, june 26-28, 2006, chicago, illinois, usa.* ACM.

Vardi, M. Y. (1982). The complexity of relational query languages (extended abstract). In H. R. Lewis, B. B. Simons, W. A. Burkhard, & L. H. Landweber (Eds.), *Proceedings of the 14th annual ACM symposium on theory of computing, may 5-7, 1982, san francisco, california, USA* (pp. 137–146). ACM. Retrieved from http://doi.acm.org/10.1145/800070.802186 doi: 10.1145/800070.802186

Wrathall, C. (1976). Complete sets and the polynomial-time hierarchy. *Theor. Comput. Sci.*, *3*(1), 23-33.