A survey of the applications of schema mapping and the certain answers semantics

Ariel D. Fuxman

April 21, 2004

1 Introduction

The problem of establishing associations (*mappings*) between data structured under different representations is relevant in many applications. In fact, it has recently been acknowledged that most *metadata management* applications involve the manipulation of mappings [Ber03]. In order to be able to exploit the mappings, their semantics must be defined precisely. In this paper, we study one semantics, that of *certain answers*, and survey its application to the following problems.

- Data integration: the problem of providing a uniform interface to multiple autonomous data sources [Hal01].
- Data exchange: the problem of taking data structured under a source schema and creating an instance of a target schema [FKMP03].
- Peer-to-peer mediation: the problem of answering queries in a network of autonomous peers [GHI$^+$01].
- Consistent query answering: the problem of obtaining consistent information from databases that may violate some given integrity constraints [ABC99].

Rather than explaining each problem in isolation, we take a top-down approach. First, we define a general framework, that we call a *mapping setting*, where the notion of mappings and certain answers are central. Second, we survey results from the literature for each of the data management problems just mentioned, and cast them in terms of the general framework of mapping settings. Finally, we unveil some commonalities between the problems that are not evident in the literature.

We will study models which are represented as relational schemas (thus the name *schema mappings*). We will concentrate on the impact of schema mappings on the problem of query answering. Therefore, we will not cover other issues such as the generation [PVM$^+$02] and maintenance [VMP03] of schema mappings.

Although the focus of the paper is on certain answers, we do not claim it is the only semantics for schema mappings that makes sense. Rather, we argue that it is a reasonable semantics by illustrating its usefulness on several data management applications. Other semantics have been proposed in the literature (certain answers on universal solutions [FKP03], epistemic interpretations [CGLR04], probabilistic notions [MM01, CGM01]). To the best of our knowledge, no systematic comparison among these semantics has been done so far.

The paper is organized as follows. In Section 2, we define mapping settings, and present some of their properties. Then, we survey results from the literature on data integration (Section 3), data

S	**T**
`Country(person,country)`	`Home(person,city)`
`Capital(city,country)`	`Loc(city,country)`

Figure 1: Source and target schemas used throughout the paper.

exchange (Section 4), peer-to-peer mediation (Section 5), and consistent query answering (Section 6). In Section 7, we prove some results that establish connections between the previous problems. These results are new, and the proofs are given in the Appendix. Finally, we give some conclusions and closing remarks.

2 General framework

2.1 Mapping setting

In this section, we define the general framework we will use throughout the document. We start with a definition of *mapping setting*,[1] which formalizes the notion of schema mapping.

Definition 2.1 (mapping setting) [2]*A* mapping setting $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$ *consists of a source schema* $\mathbf{S}$*, a target schema* $\mathbf{T}$*, and a mapping* $\mathcal{M}$*. The mapping* $\mathcal{M}$ *consists of a set of rules of the form* $\forall \mathbf{x} : \phi(\mathbf{x}) \rightarrow \psi(\mathbf{x})$*, where* $\phi(\mathbf{x})$ *and* $\psi(\mathbf{x})$ *are formulas over some logical formalism such that:*

- *all the free variables of* $\psi(\mathbf{x})$ *occur in* $\phi(\mathbf{x})$*.*[3]
- $\phi(\mathbf{x})$ *is a formula that is either over* $\mathbf{S}$ *or over* $\mathbf{T}$*, but not both.*
- $\psi(\mathbf{x})$ *is a formula that is either over* $\mathbf{S}$ *or over* $\mathbf{T}$*, but not both.*

Definition 2.2 (solution) *Let* $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$ *be a mapping setting. Let* I *be an instance over* $\mathbf{S}$*. We say that an instance* J *over* $\mathbf{T}$ *is a* solution *for* I *in* $\mathcal{S}$ *if* $\langle I, J \rangle \models \mathcal{M}$*.*

To avoid clutter in the notation, in the examples we will omit the universal quantifiers of the rules (but not the existential quantifiers). That is, if no quantifiers are given for a variable, it should be assumed to be universally quantified.

Example 2.3 Let $\mathbf{S}$ be a schema with relation symbols `Country(person,country)` and `Capital(city,country)`. Let $\mathbf{T}$ be a schema with relation symbols `Home(person,city)` and `Loc(city,country)`. We show these schemas in Figure 1, since they will be used throughout the paper.

As an example of a mapping setting, let $\mathcal{S}_1 = \langle \mathbf{S}, \mathbf{T}, \mathcal{M}_1 \rangle$, where $\mathcal{M}_1$ consists of the following rule:

$$\mathbf{S}.Country(p, cou) \rightarrow \exists cit : \mathbf{T}.Home(p, cit) \wedge \mathbf{T}.Loc(cit, cou)$$

Notice that we are using the convention of omitting universal quantifiers. Therefore, the previous rule should be interpreted as:

[1]In the rest of the document, whenever it is clear from the context, we will just say *setting* instead of *mapping setting*.

[2]Lenzerini [Len02] defines *data integration setting* in a similar way. However, the mapping rules that we consider in Definition 2.4 are more general than the ones in [Len02].

[3]Notice that the variables of $\mathbf{x}$ are the only ones that can be used in both sides of the rule.

$$\forall p, cou : (\mathbf{S}.Country(p, cou) \rightarrow \exists cit : \mathbf{T}.Home(p, cit) \wedge \mathbf{T}.Loc(cit, cou))$$

There may be more than one solution for a given instance I. For example, let $I = \{Country(john, canada)\}$. Consider $J_1 = \{Home(john, toronto), Loc(toronto, canada)\}$ and $J_2 = \{Home(john, montreal), Loc(montreal, canada)\}$. It is easy to see that both J_1 and J_2 are solutions for I in $\mathcal{S}_1$. •

In the rule given in the previous example, the left-hand side of the implication is over the source schema, and the right-hand side is over the target schema. We will classify rules according to the formulas on each side.

Definition 2.4 (generation, restriction, and target rules)

- *A* generation *rule is of the form* $\forall \mathbf{x} : \phi_s(\mathbf{x}) \rightarrow \psi_t(\mathbf{x})$, *where* ϕ_s *is a formula over* $\mathbf{S}$ *and* ψ_t *is a formula over* $\mathbf{T}$.
- *A* restriction *rule is of the form* $\forall \mathbf{x} : \phi_t(\mathbf{x}) \rightarrow \psi_s(\mathbf{x})$, *where* ϕ_t *is a formula over* $\mathbf{T}$ *and* ψ_s *is a formula over* $\mathbf{S}$.
- *A* target *rule is of the form* $\forall \mathbf{x} : \phi_t(\mathbf{x}) \rightarrow \psi_t(\mathbf{x})$, *where* ϕ_t *and* ψ_t *are formulas over* $\mathbf{T}$.

Notice that in the previous definition, we ignored rules whose left and right-hand sides are over the source schema. This is because we will always assume that a source instance is given as input.

Throughout the paper, we will be interested in answering queries posed on the target schema. Of course, the semantics adopted for query answering has to take into account the fact that there may not be exactly one solution for each source instance, as we illustrated in Example 2.3. We adopt a semantics based on the notion of *certain answers* [AGK91, AD98].

Definition 2.5 (certain answer) *Let* $\mathcal{S}$ *be a mapping setting. Let* I *be a source instance such that there exists some solution for* I *in* $\mathcal{S}$. *Let* q *be a query. We say that a tuple* $\mathbf{t}$ *is a* certain answer *to* q *in* $\mathcal{S}$, *denoted* $\mathbf{t} \in \mathbf{certain}(q, I, \mathcal{S})$, *if for every solution* J *for* I *in* $\mathcal{S}$, *it is the case that* $\mathbf{t} \in q(J)$.

Example 2.3 (continued) Let $I = \{Country(john, canada)\}$. Let $q(p) = \exists cit : \mathbf{T}.Home(p, cit)$. Since the rule of $\mathcal{M}_1$ must be satisfied by all solutions, there are tuples $Home(john, c)$ and $Loc(c, canada)$ for some city c, in every solution for I. Thus, $(john) \in q(J)$, for every solution J. We say that $(john)$ is a certain answer to q, and denote this by $(john) \in \mathbf{certain}(q, I, \mathcal{S})$.

Now, let $q(cit) = \exists p : \mathbf{T}.Home(p, cit)$. In this case, there are no certain answers to q. To see why, consider the solutions $J_1 = \{Home(john, toronto), Loc(toronto, canada)\}$ and $J_2 = \{Home(john, montreal), Loc(montreal, canada)\}$. Clearly, there is no tuple $\mathbf{t}$ such that $\mathbf{t} \in q(J_1) \cap q(J_2)$. •

We have already mentioned that, given a source instance I, there need not be a unique solution for I in $\mathcal{S}$. It may also be the case that there are no solutions at all, as we illustrate next.

Example 2.6 Let $\mathcal{S}_2 = \langle \mathbf{S}, \mathbf{T}, \mathcal{M}_2 \rangle$ be a mapping setting, where $\mathcal{M}_2$ consists of the following generation and restriction rules:

$$\mathbf{S}.Country(p, cou) \rightarrow \exists cit : \mathbf{T}.Home(p, cit) \wedge \mathbf{T}.Loc(cit, cou) \quad (1)$$
$$\mathbf{T}.Home(p, cit) \wedge \mathbf{T}.Loc(cit, cou) \rightarrow \mathbf{S}.Capital(cit, cou) \quad (2)$$

For the source instance $I = \{Country(john, canada), Capital(washington, us)\}$, there is no solution in $\mathcal{S}_2$. To see this, assume that there is a solution J for I in $\mathcal{S}_2$. By rule (1) of $\mathcal{M}_2$, J has tuples $Home(john, c)$ and $Loc(c, canada)$, for some city c. By rule (2) of $\mathcal{M}_2$, I has a tuple $Capital(canada, c)$; contradiction. Intuitively, there is no solution for I in $\mathcal{S}_2$ since the source has no information on what city is the capital of Canada.

Notice the effect of rules (1) and (2) on the solutions. The former specifies what *must* be in the solutions; we call it a *generation* rule. The latter constrains what can be in the solutions; we call it a *restriction* rule. •

2.2 Evaluation of mapping settings

The study of query answering involves three parameters: a mapping setting $\mathcal{S}$, a target query q, and a source instance I. We will assume that the setting $\mathcal{S}$ and the target query q are fixed, and study the following two problems:

- *Query evaluation.* Given a source instance I and a tuple $\mathbf{t}$, is it the case that $\mathbf{t} \in \mathbf{certain}(q, I, \mathcal{S})$? We will call this problem $\texttt{CERTAIN}(q, \mathcal{S})$. We will be interested in the *data complexity* of the problem.[4]

- *Query rewritability.* Let $\mathcal{L}$ be a query language. Does there exist a query $q' \in \mathcal{L}$ over $\mathbf{S}$ such that $q'(I) = \mathbf{certain}(q, I, \mathcal{S})$, for every source instance I? We will say that $\langle q, \mathcal{S}\rangle$ is *$\mathcal{L}$-rewritable*, and that q' is a *$\mathcal{L}$-rewriting* of $\langle q, \mathcal{S}\rangle$.

For the first problem, we will be particularly interested in the cases where $\texttt{CERTAIN}(q, \mathcal{S})$ has polynomial-time data complexity. For the second problem, the goal is to be able to reuse existing database technology in order to compute the certain answers. Thus, we will be interested in finding *first-order rewritings*, which can be easily translated into SQL. Since first-order queries can be evaluated in polynomial time, it follows that if $\langle q, \mathcal{S}\rangle$ is first-order rewritable, $\texttt{CERTAIN}(q, \mathcal{S})$ has polynomial data complexity. However, the converse does not hold. Furthermore, we point out that in some cases in which $\texttt{CERTAIN}(q, \mathcal{S})$ is tractable yet $\langle q, \mathcal{S}\rangle$ is not first-order rewritable, it may still be the case that $\langle q, \mathcal{S}\rangle$ is rewritable into some other language, such as Datalog.

The impact of the source and target schemas on the complexity of query answering is not symmetric. The intuition is that evaluating formulas over the source schema is easy because we are given a source instance as input to the problem. In contrast, evaluating formulas over the target schema is difficult because there may be more than one target solution for a given source instance, or none at all. Let us formalize this observation with a proposition which is not explicit in the literature. The idea is that, given a setting $\mathcal{S}$, we can produce another setting, which we call *normalized*, that is equivalent to $\mathcal{S}$ but replaces the formulas over the source by single atoms. More formally, in a *normalized* setting, if $\phi_s(\mathbf{x})$ is a formula over $\mathbf{S}$ that appears in some rule of $\mathcal{M}$, then $\phi_s(\mathbf{x})$ is an atomic formula.

Proposition 2.7 (normalization) *Let $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M}\rangle$ be a mapping setting. Assume that in every rule of $\mathcal{M}$, the formulas over $\mathbf{S}$ are expressed in some language $\mathcal{L}$ that has polynomial-time data complexity (the formulas over $\mathbf{T}$ can be expressed in any arbitrary language). Then, there exists a* normalized *setting $\mathcal{S}_{norm} = \langle \mathbf{S}_{norm}, \mathbf{T}, \mathcal{M}_{norm}\rangle$ such that for every instance I over $\mathbf{S}$, there exists an instance I' over $\mathbf{S}_{norm}$ such that :*

- *I' can be obtained in polynomial-time in the size of I, and*

- *J is a solution for I in $\mathcal{S}$ iff J is a solution for I' in $\mathcal{S}_{norm}$.*

[4]Unless stated otherwise, in the following, when we say *complexity*, we will mean *data complexity*.

CQ	conjunctive queries
UCQ	union of conjunctive queries
Datalog	Datalog queries
FO	first-order queries

Figure 2: Abbreviations used for query languages.

As a corollary, if a setting $\mathcal{S}$ is such that in all its rules the formulas over the source are in first-order or Datalog (which have polynomial-time data complexity), the problem of computing certain answers in $\mathcal{S}$ can be polynomially-reduced to the problem of computing certain answers in the normalized setting. Thus, in our study of the problem of query answering we will focus on the formulas over the target, rather than the source. That is, whenever we say that the *mapping language* of a setting $\mathcal{S}$ is $\mathcal{L}$ (denoted $\mathcal{S} \in \mathcal{L}$), we will mean that in each rule of $\mathcal{M}$, the formula(s) over the target schema is (are) written in language $\mathcal{L}$.

In its full generality, the query answering problem is undecidable. For example, let $\mathcal{S}$ be a setting with a first-order mapping language, and q be a first-order query. It is easy to reduce $\texttt{CERTAIN}(q, \mathcal{S})$ from the problem of testing whether a first-order formula admits a finite model, which is known to be undecidable (Trakhtenbrot's theorem [EF95]). We will thus be interested in *classes* of queries and mapping settings for which the query answering problem is decidable and, most importantly, tractable. The classes will typically be defined with respect to the language of the user queries and the mapping language of the setting. In Figure 2, we give the abbreviations we will use for such languages.

Definition 2.8 [5] *Let $\mathtt{Class}_q$ be a class of queries, and $\mathtt{Class}_\mathcal{S}$ be a class of mapping settings. Let $\mathcal{C}$ be a complexity class.*

- **Membership**. *We say that CERTAINCLASS($\mathtt{Class}_q, \mathtt{Class}_\mathcal{S}$) is in $\mathcal{C}$ if for* every *query $q \in \mathtt{Class}_q$ and setting $\mathcal{S} \in \mathtt{Class}_\mathcal{S}$, CERTAIN($q, \mathcal{S}$) is in $\mathcal{C}$.*
- **Completeness**. *We say that CERTAINCLASS($\mathtt{Class}_q, \mathtt{Class}_\mathcal{S}$) is $\mathcal{C}$-complete if it is in $\mathcal{C}$, and there is* at least one *query $q \in \mathtt{Class}_q$ and mapping setting $\mathcal{S} \in \mathtt{Class}_\mathcal{S}$ such that CERTAIN($q, \mathcal{S}$) is a $\mathcal{C}$-complete problem.*

Notice the asymmetry in the definitions. While for membership it is required that all $\langle q, \mathcal{S} \rangle$ be in the complexity class, for completeness it suffices to show one query q and setting $\mathcal{S}$ such that $\texttt{CERTAIN}(q, \mathcal{S})$ is $\mathcal{C}$-complete.

Example 2.9 The problem of computing certain answers is known to be in coNP for every $q \in CQ$ and $\mathcal{S} \in UCQ$ [AD98]. Now, consider the setting $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$ such that $\mathbf{S} = \{S_1, S_2\}$, $\mathbf{T} = \{Edge, Color\}$, and $\mathcal{M}$ is the following:

$$S_1(x) \rightarrow Color(x, red) \vee Color(x, green) \vee Color(x, blue)$$
$$S_2(x, y) \rightarrow Edge(x, y)$$

Let $q = \exists x, y, z : Edge(x, y) \wedge Color(x, z) \wedge Color(y, z)$. It was shown in [AD98] that $\texttt{CERTAIN}(q, \mathcal{S})$ is coNP-hard. From this particular query and setting, we can conclude that $\texttt{CERTAINCLASS}(CQ, UCQ)$ is coNP-complete. •

As illustrated in the previous example, it suffices to show just one query and setting for which the problem is intractable in order to conclude the intractability of a whole class. Therefore, when we

[5] The definition is adapted from [FKMP03], and it is implicitly used by all other authors in the literature.

say that a class is intractable, we are not precluding the fact that there may be large, and practical, subclasses for which the problem is indeed tractable. The best known tractable class consists of settings $\mathcal{S}$ with only generation rules (i.e., no restriction or target rules) such that $\mathcal{S} \in CQ$, and user queries q such that $q \in CQ$. This class has been thoroughly studied in the data integration literature, which we survey next.

3 Data Integration

Data integration systems "provide a uniform interface to multiple autonomous data sources" [Hal01]. The user interacts with the system by posing queries over a global schema, which gives an integrated conceptual model of the application domain. The global schema is *virtual*, i.e., data resides in the sources. The schemas of the data sources are hidden to the user.

In our terms, we can define a data integration system as a mapping setting $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$, where $\mathbf{S}$ is the schema of the data sources, $\mathbf{T}$ is the global schema, and $\mathcal{M}$ are the semantic connections between the global and source schemas.

3.1 Open systems

Most data integration systems consider only generation rules for their mappings. The systems that have only generation rules are sometimes called *open* because the source instance determines what *must* be in the solutions, but it does not give *negative* information (i.e., it does not specify what *must not* be in any solution). Formally, let $\mathcal{S}$ be an open setting. If J is a solution for an instance I in $\mathcal{S}$, and J' is such that $J \subseteq J'$, then J' is a solution for I in $\mathcal{S}$.

Example 2.3 (continued) The setting $\mathcal{S}_1$ is open, since it only has the following generation rule:

$$\mathbf{S}.Country(p, cou) \rightarrow \exists cit : \mathbf{T}.Home(p, cit) \wedge \mathbf{T}.Loc(cit, cou)$$

Let $I = \{Country(john, canada)\}$. Let $J = \{Loc(calgary, canada), Home(john, ottawa), Loc(ottawa, canada)\}$, which is a solution for I in $\mathcal{S}_1$. The tuple $Loc(calgary, canada)$ does not seem to be related to the sources, and the mapping does not *force* us to add it to the solution. However, it does not forbid us from including it in the solution either. In fact, we could add any arbitrary tuple to J, and still have a solution for I.

•

There are several reasons why the data integration community has focused on open settings. First, as we mentioned in the previous section, the query answering problem is tractable for a large class of open settings. We will evaluate query answering in those settings in Section 3.3. Second, given a source instance, open systems always have a solution. This is important because it was typically assumed in data integration that the global database would accommodate all data coming from the sources. In fact, there is a whole research area, *schema integration* [BLN86], which deals with the problem of *designing* a global schema in such a way that it is suitable for all the sources. In such a context, a situation in which there are no solutions for a given source is considered to be undesirable.

3.2 Global-as-View and Local-as-View

The initial work on data integration focused on mapping rules of the form $\forall \mathbf{x} : R(\mathbf{x}) \rightarrow \alpha(\mathbf{x})$ or $\forall \mathbf{x} : \alpha(\mathbf{x}) \rightarrow R(\mathbf{x})$, where $R(\mathbf{x})$ is a literal and $\alpha(\mathbf{x})$ is a formula over some logical formalism. It is usually said that $R(\mathbf{x})$ is defined as a *view*. Typically, data integration systems are classified depending on whether the source or target relations are defined as a view. In particular:

- In *global-as-view* systems (GAV), global relations are defined as views over the source schema. Formally, rules are of the form $\forall \mathbf{x} : \phi_s(\mathbf{x}) \rightarrow R_t(\mathbf{x})$, where R_t is a relation symbol from $\mathbf{T}$ and $\phi_s(\mathbf{x})$ is a formula over $\mathbf{S}$ (i.e., rather than a formula, there is just one atom on the right-hand side).

- In *local-as-view* systems (LAV), source relations are defined as views over the global schema. Formally, rules are of the form $\forall \mathbf{x} : R_s(\mathbf{x}) \rightarrow \psi_t(\mathbf{x})$, where R_s is a relation symbol from $\mathbf{S}$ and $\psi_t(\mathbf{x})$ is a formula over $\mathbf{T}$ (i.e., rather than a formula, there is just one atom on the left-hand side).

Example 3.1 The mapping setting $\mathcal{S}_1$ of Example 2.3 is a LAV setting. Now, we give an example of a GAV setting $\mathcal{S}_3 = \langle \mathbf{S}, \mathbf{T}, \mathcal{M}_3 \rangle$, where $\mathcal{M}_3$ consists of the following rule:

$$\exists p : \mathbf{S}.Capital(cit, cou) \wedge \mathbf{S}.Country(p, cou) \rightarrow \mathbf{T}.Loc(cit, cou)$$

•

A main appeal of GAV is that query answering is easy: the mapping rules indicate explicitly how to retrieve data from the sources. In particular, given a query q over the global schema, it suffices to *unfold* q using the mapping rules in order to obtain a rewriting q' of q (i.e., a query that computes the certain answers to q).

Example 3.1 (continued) Suppose that a user wants to retrieve the cities and countries from relation `Loc`. Thus, she issues the following query over the global schema:

$$q(cit, cou) = \mathbf{T}.Loc(cit, cou)$$

In order to obtain a rewriting of q, it suffices to replace $Loc(cit, cou)$ by its definition in $\mathcal{M}_3$:

$$q'(cit, cou) = \exists p : \mathbf{S}.Capital(cit, cou) \wedge \mathbf{S}.Country(p, cou)$$

•

Query processing in LAV is much more involved than in GAV, since the unfolding strategy no longer works in this case. More importantly, the complexity of query answering in LAV is much higher than in GAV, as we will see later. The intuitive explanation for this is that in GAV, given a source instance I, it suffices to concentrate on a single solution J. More formally, it can be shown that given an instance I and a GAV setting $\mathcal{S}$, there exists a solution J such that J is the intersection of every solution J' for I in $\mathcal{S}$. We illustrate this in the following example.

Example 3.1 (continued) Let $I = \{Capital(ottawa, canada), Country(john, canada)\}$ be an instance of $\mathbf{S}$. Let $J_1 = \{Loc(ottawa, canada)\}$. It is easy to see that J_1 is a solution for I in the GAV system $\mathcal{S}_3$ and that, for every solution J' for I in $\mathcal{S}_3$, $J_1 \subseteq J'$. Therefore, it suffices to concentrate on J_1 in order to obtain the certain answers.[6]

•

It is claimed in the literature [Hal01, Len02] that the main disadvantage of GAV systems is that they are not appropriate for dynamic environments, in which new sources appear frequently. For example, Halevy points out that "we need to consider the possible interaction of the new source with each of the existing sources, and this limits the ability of the GAV approach to scale to a large

[6] To be precise, the argument holds only for monotone queries.

collection of sources" [Lev00]. We argue that this is not necessarily the case. While it is true that GAV systems allow for the specification of interactions among data sources, they do not enforce it. As we show in the next example, the addition of data sources to a GAV system may simply amount to adding a disjunct to some of the rules of the mapping.

Example 3.1 (continued) Assume that a new source arrives in the system, with schema `City(person,city,country)`. We can update $\mathcal{M}_3$ by replacing its rule with the following:

$$((\exists p : \mathbf{S}.Capital(cit, cou) \wedge \mathbf{S}.Country(p, cou)) \vee \exists p : \mathbf{S}.City(p, cit, cou)) \rightarrow \mathbf{T}.Loc(cit, cou))$$

Notice that, if desired, it would be possible to establish some semantic connection between `City` and `Country`. However, we are not required to do so. Thus, when the sources are independent, the relations of each source will appear in independent disjuncts of the rules.

•

Since query answering in GAV is much easier than in LAV, a natural question is: why are LAV systems needed at all? The answer is that GAV and LAV systems have incomparable expressive power. That is, there exists a LAV system $\mathcal{S}_{lav} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$ for which there is no GAV system $\mathcal{S}_{gav} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M}' \rangle$ such that J is a solution for I in $\mathcal{S}_{lav}$ iff J is a solution for I in $\mathcal{S}_{gav}$, for every instance I. The same holds for GAV systems.

In conclusion, GAV and LAV should not be presented as complementary alternatives. When the main concern is efficiency in query processing, rather than expressive power, GAV is the clear choice. But if expressive power is an issue, then the distinction between GAV and LAV becomes somewhat artificial; a system that combines both kinds of rules is desirable.

Friedman et al. [FLM99] introduced GLAV systems, which coincide with our definition of open settings (i.e., mapping settings with only generation rules). Notice that a LAV system is a *normalized* setting, in the sense used in Proposition 2.7. As a result of that proposition, the problem of query answering in a GLAV system with first-order or datalog formulas[7] over the source schema can be reduced in polynomial time to query answering in a LAV system. Thus, in spite of its increased expressive power, GLAV does not come with a price in data complexity when compared to LAV.

3.3 Query answering in open systems

In this section, we review the literature on the problem of query answering in open data integration systems. Most of the results were given in the context of LAV systems; however, by Proposition 2.7, they are readily applicable to open systems where the formulas over the source are in first-order or Datalog.

The problem of *answering queries using views* [LMSS95] was originally studied in the context of query optimization [CKPS95]. However, this problem has an immediate application in data integration systems. The main notion that is studied in the answering queries using views literature is that of *equivalent query rewriting.*

Definition 3.2 (equivalent query rewriting) *Let $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$ be an open data integration setting. Assume that $\mathcal{S}$ is normalized. Let q be a query over $\mathbf{T}$. Let q' be a query over $\mathbf{S}$.[8] Let q'' be a query over $\mathbf{T}$ that is obtained by unfolding q' with the generation rules of $\mathcal{M}$. We say that q' is an* equivalent query rewriting *of q if for every instance J over $\mathbf{T}$, it is the case that $q(J) = q''(J)$.*

[7] Friedman et al. considered the case of datalog formulas over the source schema, but do not generalize the result to any formula in a language with polynomial-time data complexity.

[8] In the original definition [LMSS95], the rewritten query is over $\mathbf{S} \cup \mathbf{T}$ (i.e., the rewriting may have some relations from $\mathbf{T}$). This makes sense for query optimization, but not for query integration where $\mathbf{T}$ is virtual. Our definition actually corresponds to *complete rewritings* [LMSS95].

Example 3.3 [9] Let $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$ be a setting such that $\mathbf{S}$ has relation symbols S_1 and S_2 and $\mathbf{T}$ has relation symbols T_1, T_2, T_3, and T_4. Let $\mathcal{M}$ have the following rules:

$$S_1(x,y) \to \exists u, v : T_1(x,u) \wedge T_2(u,y) \wedge T_3(x,v)$$
$$S_2(x,y) \to \exists w : T_3(x,w) \wedge T_4(w,y)$$

Let $q(x,y) = \exists u, w, z : T_1(x,u) \wedge T_2(u,z) \wedge T_3(x,w) \wedge T_4(w,y)$. Let $q'(x,y) = \exists z : S_1(x,z) \wedge S_2(x,y)$. We cannot compare q and q' because they are over different schemas. Thus, Definition 3.2 resorts to the unfolding of q', which is over the same schema as q. In this case, $q''(x,y) = \exists z, u, v, w : T_1(x,u) \wedge T_2(u,z) \wedge T_3(x,v) \wedge T_3(x,w) \wedge T_4(w,y)$. It is easy to see that for every instance J, $q(J) = q''(J)$. Thus, q' is an equivalent query rewriting of q.

•

If an equivalent rewriting exists, then it computes the certain answers. However, as we show in the next example, the converse does not hold.

Example 3.4 Consider setting $\mathcal{S}_1$ of Example 2.3. We recall that $\mathcal{M}_1$ consists of the following rule:

$$\mathbf{S}.Country(p, cou) \to \exists cit : \mathbf{T}.Home(p, cit) \wedge \mathbf{T}.Loc(cit, cou)$$

Let $q(cou) = \exists cit : \mathbf{T}.Loc(cit, cou)$. Let $q'(cou) = \exists p : \mathbf{S}.Country(p, cou)$. It can be checked that q' computes the certain answers of q, for every instance I over $\mathbf{S}$. However, q' is not an equivalent query rewriting of q. To see this, we first compute the unfolding of q', $q''(cou) = \exists p, cit : \mathbf{T}.Home(p, cit) \wedge \mathbf{T}.Loc(cit, cou)$. If we let $J = \{Home(john, toronto), Loc(toronto, canada), Loc(london, england)\}$, we have that $q''(J) = \{(canada)\}$ and $q(J) = \{(canada), (england)\}$. The intuitive reason why the tuple $(england)$ does not appear in $q''(J)$ is that, according to the mapping, the source only has information about countries where at least one person from relation `Home` live. •

Although not mentioned in the seminal paper of Levy, Mendelzon et al. [LMSS95], later work on answering queries using views introduced a concept, called *maximally-contained rewriting* [DGL00], that is related to the notion of certain answers. Indeed, it can be shown that, if q' is a query in some monotone language $\mathcal{L}$, then q' is a maximally-contained rewriting of q wrt $\mathcal{L}$ if and only if q' computes the certain answers for q (i.e., it is a $\mathcal{L}$-rewriting of q, as defined in Section 2.2). For instance, the query q' of Example 3.4, which was shown not be an equivalent rewriting, is in fact a maximally-contained rewriting.

Definition 3.5 (maximally-contained rewriting) *Let $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$ be an open data integration setting. Assume that $\mathcal{S}$ is normalized. Let q be a query over $\mathbf{T}$. Let q' be a query over $\mathbf{S}$. Let q'' be a query over $\mathbf{T}$ that is obtained by unfolding q' with the generation rules of $\mathcal{M}$. We say that q' is a* contained rewriting *of q if for every instance J over $\mathbf{T}$, we have that $q''(J) \subseteq q(J)$.*

Let $\mathcal{L}$ be some query language. We say that q' is a maximally-contained rewriting *of q wrt $\mathcal{L}$ if for every contained rewriting q''' of q, q''' is such that $q'''(I) \subseteq q'(I)$, for every instance I over $\mathbf{S}$.*

The notion of maximally-contained query rewriting is closely related to the problem of *query containment* [CM77]. A query q is *contained* in a query q' if, for every instance D, $q(D) \subseteq q'(D)$. A fundamental result of Chandra and Merlin [CM77] states that a conjunctive query q is contained in a conjunctive query q' if and only if there is an homomorphism from q' to q. This can be used as the building block of an algorithm to compute a maximally-contained rewriting with respect to the language of conjunctive queries, for open settings $\mathcal{S} \in CQ$.

[9]The example is adapted from [LMSS95].

Input: $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$, an open setting such that $\mathcal{S} \in CQ$
q, a conjunctive query over $\mathbf{T}$.
Output: q', a union of conjunctive queries that is the maximally-contained rewriting of q.

0: $q' :=$ **false**
0: Let $\{r_1, \ldots, r_n\}$ be the rules of $\mathcal{M}$.
0: Let r_i be of the form $\forall \mathbf{x} : \phi_i(\mathbf{x}) \rightarrow \psi_i(\mathbf{x})$, for $1 \leq i \leq n$.
for each homomorphism h from $\{\psi_1, \ldots, \psi_n\}$ to q **do**
$q' := q' \vee \bigwedge_{1 \leq i \leq n} \phi_i(h(\mathbf{x}))$
end for

The above algorithm gives the right intuition on the role of the query containment problem, but is very inefficient, since in the worst case there may be an exponential number of homomorphisms. In fact, it follows from results in [LMSS95] that the problem of finding a maximally-contained rewriting can be reduced from the problem of query containment, which is NP-complete.[10]

The naive algorithm must be improved in two directions in order to be practical. First, the rewritten queries must be *obtained* efficiently. Second, given a source instance, the queries should be *evaluated* efficiently. For the first direction, several heuristic algorithms have been proposed. Levy, Rajaraman and Ordille [LRO96] introduced the *bucket algorithm* in the context of the Information Manifold project, which contains a pruning step that removes sources that are not relevant for the rewriting. The MiniCon algorithm [PL00] builds on the bucket algorithm, but introduces a more drastic pruning strategy by taking into account the interactions between variables in the query. The effectiveness of Minicon has been validated by extensive experimentation [PL00]. For the second direction, Ives et al. [IFF$^+$99] studied the issue of query optimization within a data integration system.

The results presented so far assume that $\mathcal{S} \in CQ$ and $q \in CQ$. Abiteboul and Duschka [AD98] showed that the addition of inequalities to the mapping language (i.e., $\mathcal{S} \in CQ^{\neq}$ and $q \in CQ$) does not have an effect on the certain answers (i.e., they are exactly the same as if the inequalities in the formulas over the target schema were omitted). Afrati et al. [ALM02] presented an extension of the Minicon algorithm for the case in which $\mathcal{S} \in CQ^{\neq}$, and the query is conjunctive with a restricted form of comparisons. In particular, the queries can have left semi-intervals, which are comparisons of the form $x < c$, where x is a variable and c is a constant.[11]

Finally, Duschka and Genesereth [DGL00] introduced an algorithm, called *inverse rules*, that works for $\mathcal{S} \in CQ$ and $q \in Datalog$. The idea of the algorithm is to produce a set of logic-programming rules $\mathtt{R_s}(\mathbf{x})$:- $\mathtt{R_t}(\mathbf{x})$, where R_s and R_t are relation symbols from $\mathbf{S}$ and $\mathbf{T}$, respectively. These rules, together with a query q, are intended to compute the certain answers for q. That is, if $q \in Datalog$, and $\mathcal{P} = \mathcal{P}^{\mathcal{S}} \cup q$, then $\mathbf{certain}(q, I, \mathcal{S}) = \mathcal{P}(I)_{\downarrow}$[12], for every instance I. Notice that the interpretation of the logical-programming rule in our notation is $\forall \mathbf{x} : R_t(\mathbf{x}) \rightarrow R_s(\mathbf{x})$, and therefore the algorithm is *inverting* the rules in $\mathcal{M}$.

Example 3.6 For the setting $\mathcal{S}_1$ of Example 2.3, the inverse rules program $\mathcal{P}^{\mathcal{S}_1}$ is the following:

$$\mathtt{Home(p, f(p, cou))} \text{ :- } \mathtt{Country(p, cou)}$$
$$\mathtt{Loc(f(p, cou), cou)} \text{ :- } \mathtt{Country(p, cou)}$$

[10] Notice that this *not* data complexity, since the input to the rewriting algorithm is a setting and a query, but not a source instance. Since the rewritten query q' can be evaluated in polynomial time on the sources, the problem is indeed tractable in data complexity.
[11] The algorithm is also applicable to right semi-intervals, where comparisons are of the form $x > c$.
[12] The operator $\downarrow$ removes the tuples that have Skolem values (i.e., instantiations of the Skolem functions)

mapping	query			
	CQ	CQ^{LSI}	UCQ	Datalog
CQ	PTIME	PTIME	PTIME	PTIME
$CQ^{\neq}$	PTIME	PTIME	PTIME	PTIME

Figure 3: Tractable cases of query answering on open systems.

mapping	query				
	CQ	$CQ^{\neq}$	UCQ	Datalog	FO
CQ	PTIME	coNP	PTIME	PTIME	undec.
$CQ^{\neq}$	PTIME	coNP	PTIME	PTIME	undec.
SOtgs	PTIME		PTIME	PTIME	undec.
UCQ	coNP	coNP	coNP	coNP	undec.
Datalog	coNP	undec.	coNP	undec.	undec.
FO	undec.	undec.	undec.	undec.	undec.

Figure 4: Complexity of query answering on open systems.

The functions that appear in the program are called Skolem functions. In [DGL00], it was shown that, in general, it is possible to obtain an equivalent Datalog program that removes such functions. •

So far, we have have focused on the tractable cases of query answering on open systems. These cases are summarized in Figure 3. In Figure 4, we show the data complexity of other classes of queries and settings. All the results, except those for mappings expressed with *second-order tgds* (SOtgds) [FKPT04], are due to Abiteboul and Duschka [AD98]. We will present second-order tgds in Section 5.1. We point out that the problem of query answering in open systems has also been studied for other query languages (e.g., languages with aggregation [GL00], description logics [BLR97]) and on other data models (e.g., semistructured model [CGLV00]).

We have already shown that the problem is tractable for $\mathcal{S} \in CQ$ and $q \in CQ$. Intuitively, if the expressive power of the mapping language is not increased, no penalty is paid when we allow union or recursion in the query language. Surprisingly, though, the addition of inequalities to the query language makes the problem intractable. The opposite situation occurs when the expressive power of the mapping language is increased. Allowing union or recursion has a considerable impact on the complexity of the problem, making it either intractable or undecidable. The undecidability for the class where $q \in Datalog$ and $\mathcal{S} \in Datalog$ follows from the undecidability of query containment for datalog queries [Shm93]. For the class where $q \in FO$ and $\mathcal{S} \in FO$, it follows from Trakhtenbrot's theorem [EF95].

3.4 Restriction rules

Although the primary focus has been on generation rules, restriction rules have received some attention in the data integration literature. For example, Abiteboul and Duschka [AD98] considered the *open world* and *closed world* assumptions. The former coincides with our definition of open settings. The latter assumes that all rules in the mapping are of the form $\forall \mathbf{x} : \phi_s(\mathbf{x}) \leftrightarrow \psi_t(\mathbf{x})$, where $\phi_s(\mathbf{x})$ is over the source, and $\psi_t(\mathbf{x})$ is over the target. Notice that each of the rules in the closed world assumption can be expressed as two rules in a mapping setting: a generation and a restriction rule. In [AD98], the complexity of computing certain answers is studied for different classes of settings under the closed world assumption. The results are summarized in Figure 5. As opposed to settings under the open

mapping	query				
	CQ	$CQ^{\neq}$	UCQ	Datalog	FO
CQ	coNP	coNP	coNP	coNP	undec.
$CQ^{\neq}$	coNP	coNP	coNP	coNP	undec.
UCQ	coNP	coNP	coNP	coNP	undec.
Datalog	undec.	undec.	undec.	undec.	undec.
FO	undec.	undec.	undec.	undec.	undec.

Figure 5: Complexity of query answering under the closed world assumption.

world assumption (see Figure 4), in this case the problem is intractable even for the class where $q \in CQ$ and $\mathcal{S} \in CQ$.

Grahne and Mendelzon [GM99] generalized the notion of open and closed world assumptions. In particular, they annotate each relation symbol of the source schema with a label that tells whether it is *sound*, *complete*, or both. A sound source is defined by a generation rule; a complete source is defined by a restriction rule; a sound and complete source is defined by the kind of rules used in the closed world assumption. Notice that mapping settings are more general than the settings defined in [GM99], since in the former the relation symbols of the source can be involved in several rules, and in different kinds of rules (i.e., generation, restriction, and target rules).

Finally, Duschka and Genesereth [DGL00] showed how to extend their inverse rules algorithm to open settings where the relations of the target schema have some functional dependencies. Notice that functional dependencies are a special class of target rules. We will show in Section 7 that the target rules that represent functional dependencies can also be expressed as restriction rules.

4 Data Exchange and Models for Incomplete Databases

"Data exchange is the problem of taking data structured under a source schema and creating an instance of a target schema that reflects the source data as accurately as possible" [FKMP03]. The difference with data integration is that the goal is to *materialize* a target instance, rather than answer queries on the sources.

The *materialized target instance* should be an instance that represents the space of all solutions for the given source instance. Fagin et al. [FKMP03] identified a class of instances, called *universal solutions*, that enjoy this property in the case of open settings. In particular, universal solutions are solutions that have homomorphisms into every other solution. Since universal solutions are not unique, in [FKMP03] it is also shown how to compute one particular universal solution, called the *canonical* universal solution, in polynomial time (for open settings $\mathcal{S}$ such that $\mathcal{S} \in CQ$). The algorithm is based on the chase procedure [MMS79]. Furthermore, it is also shown that the evaluation of a union of conjunctive queries (or Datalog query) on a universal solution gives precisely the set of certain answers. Formally, given an instance I, $\mathbf{certain}(q, I, \mathcal{S}) = q(J)_{\downarrow}$[13], for every solution J for I.

Example 4.1 Consider the setting $\mathcal{S}_1$ of Example 2.3, where the mapping has the following rule:

$$\mathbf{S}.Country(p, cou) \rightarrow \exists cit : \mathbf{T}.Home(p, cit) \wedge \mathbf{T}.Loc(cit, cou)$$

Let $I = \{Country(john, canada), Country(mary, us)\}$. By chasing I with the above rule, we obtain $J_{can} = \{Home(john, c_1), Loc(c_1, canada), Home(mary, c_2), Loc(c_2, us)\}$. By definition, J_{can} is the canonical universal solution.

•

[13]The operator $\downarrow$ removes the tuples that have variables.

The canonical universal solution can be obtained in polynomial time not only for open settings but also for settings that have a restricted form of target rules. In particular, the target rules should be such that, when used to chase the source instance, every chase sequence has a length that is polynomial in the size of the input. In [FKMP03], the authors exhibit a class of target rules, called *weakly acyclic*, which enjoy this property. This class strictly includes both full and acyclic [CK86] tuple generating dependencies.

The purpose of query rewriting in this context is to find a query over the target schema (not the source, as in data integration) which retrieves the certain answers when evaluated on the materialized target instance. In [FKMP03], it is shown that it is not always possible to obtain a query rewriting on the canonical universal solution. More specifically, they show the the existence of a setting $\mathcal{S}$ and query q, such that $\mathtt{CERTAIN}(q, \mathcal{S})$ is tractable, yet there is no first-order rewriting over the canonical universal instance. Since the canonical universal instance is just one choice of materialized target database, the following question arises: is there a transformation $\mathcal{F}$ that maps each source instance I into a materialized target instance $\mathcal{F}(I)$, and a rewriting q' such that $q'(\mathcal{F}(I))$ retrieves the certain answers?

This question was addressed by Arenas et al. [ABFL04]. They define *locally consistent* transformations, which are transformations that preserve a certain degree of uniformity in the materialized target instance, in the sense that it is not tailored to deal with a specific query. In particular, the canonical universal solution, and its core [FKP03], can be obtained by locally consistent transformations. In [ABFL04], they give some general tools to prove inexpressibility on materialized targets obtained by locally consistent transformations. It follows from their results that there are queries that do not have a first-order rewriting over any materialized target instance obtained with a locally consistent transformation.

The notion of universal solution is related to existing work in the data integration literature. In particular, the inverse rules algorithm [DGL00] implicitly constructs a canonical universal solution every time it answers a query. We illustrate this in the next example.

Example 3.6 (continued) Consider the example of the inverse rules algorithm introduced in Section 3.3. Let $I = \{Country(john, canada), Country(mary, us)\}$. The least-fixpoint model that results from applying the inverse rules program $\mathcal{P}^{\mathcal{S}_1}$ to I is $\mathcal{P}^{\mathcal{S}_1}(I) = \{Home(john, f(john, canada)),$ $Loc(f(john, canada), canada), Home(mary, f(mary, us)), Loc(f(mary, us), us)\}$. Compare this with the canonical universal solution obtained in Example 4.1, for the same setting $\mathcal{S}_1$ and instance I. In that example, we showed that the canonical universal solution for I in $\mathcal{S}_1$ is $J_{can} = \{Home(john, c_1),$ $Loc(c_1, canada), Home(mary, c_2), Loc(c_2, us)\}$. Notice that $\mathcal{P}^{\mathcal{S}_1}(I)$ and J_{can} are isomorphic instances. •

We next illustrate that, under the presence of arbitrary restriction rules, universal solutions may not be an appropriate choice of materialized target instance.

Example 4.2 Let $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$. Let $\mathbf{S} = \{S\}$ and $\mathbf{T} = \{T\}$. Let $\mathcal{M}$ have the following generation and restriction rules:

$$S(x) \rightarrow \exists y : T(y)$$
$$T(x) \rightarrow S(x)$$

Let $I = \{S(a), S(b)\}$. The only solutions for I are: $J_1 = \{T(a)\}$, $J_2 = \{T(b)\}$ and $J_3 = \{T(a), T(b)\}$. A universal solution here is for example $J_{univ} = \{T(x)\}$. Let $K = \{T(d)\}$. It is easy to see that there is a homomorphism from J_{univ} to K. However, K is not a solution for I. •

Materialized target instances (and in particular, universal solutions) contain constants and *variables* in their tuples. Therefore, they can be thought of as *incomplete databases* [IL84]. This connection was originally realized by Abiteboul and Duschka [AD98], who showed how to use *conditional tables* [AGK91] as materialized target instances.

Example 4.3 Let $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$. Let $\mathbf{S} = \{S\}$ and $\mathbf{T} = \{T\}$. Let $\mathcal{M}$ have the following generation and restriction rules:

$$S(x,y) \rightarrow \exists z : T(x,z) \wedge z \neq y$$
$$S(x,y) \rightarrow T(x,y) \vee T(y,x)$$

Let $I = \{S(a,b)\}$. By following the procedure of Abiteboul and Duschka for obtaining materialized target instances, we obtain the following conditional table:

Tuple	Condition
(a, z_1)	$z_1 \neq b$
(a, b)	$w = 0$
(b, a)	$w = 1$
(a, b)	$w \neq 0 \wedge w \neq 1$
(b, a)	$w \neq 1 \wedge w \neq 1$

•

There is a notion of certain answers for conditional tables [AGK91].[14] It is proven in [AD98] that, given a query $q \in Datalog^{\neq}$ and a setting $\mathcal{S} \in CQ^{\neq}$, the certain answers to q computed on the materialized target instance are exactly $\mathbf{certain}(q, I, \mathcal{S})$. Notice that the mapping language is more expressive than the language of conjunctive queries used in [DGL00] and [FKMP03]. In fact, when $\mathcal{S} \in CQ$, the algorithm of Abiteboul and Duschka produces a conditional table which is precisely a canonical universal solution.

5 Peer-to-peer mediation systems

Peer-to-peer (P2P) technology has been highly successful in the domain of file exchange, with popular tools such as Kazaa being used by millions of users. In a P2P system, participants (peers) rely on one another for service, blurring the distinction between clients and servers. P2P systems are founded on the principles of *peer autonomy* and *decentralized coordination*. As a result, peers do not have a global view of the system. Rather, global behavior emerges from local interactions.

As the horizon of applications of P2P technology expands, it will become necessary to place increasing emphasis on the semantics of the data being exchanged. Recently, Bernstein et al. [BGK+02] and Halevy et al. [HIST03, GHI+01] have motivated the need for Peer Data Management systems (PDMS). In a PDMS, each peer has a schema that describes the structure of its data, and can establish connections (specified as mapping rules) with other peers in order to exchange data. A PDMS is expected to satisfy the desirable properties of P2P systems. For example, the requirement of decentralized coordination precludes the existence of a central catalog. Rather, knowledge about schemas and mappings should be distributed among the peers.

A PDMS should support an arbitrary network of mappings among peers. More importantly, it should be able to exploit the transitive relationships of the network during query answering. Therefore, the problem of mapping composition, which we address in the next section, becomes very relevant.

A PDMS is essentially a set of data integration systems related through global mappings, called *peer mappings* [HIST03, CGLR04]. We reformulate their definition in terms of mapping settings as follows.

[14] In all fairness, the notion of certain answers appeared first in the context of incomplete databases.

Definition 5.1 *We say that $\mathcal{PDMS} = \langle \mathcal{P}, \mathcal{L}, \mathcal{G} \rangle$ is a PDMS if*

- $\mathcal{P} = \{P_1, \ldots, P_n\}$. *We call each P_i a* peer.
- $\mathcal{L} = \{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$ *is a set of mapping settings. We say that $\mathcal{S}_i = \langle \mathbf{L}, \mathbf{G}, \mathcal{M} \rangle$ is the* local setting *for peer P_i, and that $\mathbf{L}$ and $\mathbf{G}$ are the local and global schemas of P_i, respectively.*
- $\mathcal{G}$ *is a set of mapping settings, called* peer settings. *For each peer setting $\mathcal{S}$, the source and target schemas are the global schemas of some peers P_i and P_j, respectively, for $P_i \in \mathcal{P}$ and $P_j \in \mathcal{P}$.*

Let I be an instance over the union of all local schemas of $\mathcal{PDMS}$. We say that J is a solution for I in $\mathcal{PDMS}$ if J is an instance over the union of all local and global schemas of $\mathcal{PDMS}$ such that J is a solution for I in every local or peer setting of $\mathcal{PDMS}$.

In a PDMS, peers must establish some sort of agreement on the domain of their databases. One mechanism for achieving this are *mapping tables* [KAM03]. Mapping tables are complementary to the mappings described here: the former are defined at the data level; the latter, at the schema level. Another important issue is the protocol that the peers will use in order to coordinate the actual exchange of data. A proposal of a protocol based on active rules is given in [KKM$^+$03].

We can think of a PDMS as a directed graph whose nodes are peer schemas. In the graph, there is an arc from $\mathbf{G}_1$ to $\mathbf{G}_2$ if there is a generation rule in some peer setting $(\mathbf{G}_1, \mathbf{G}_2, \mathcal{M})$ or a restriction rule in some peer setting $(\mathbf{G}_2, \mathbf{G}_1, \mathcal{M})$. We say that this graph represents the *topology* of the PDMS. It turns out that the topology of the PDMS has a direct impact on query answering. In particular, Halevy et al. [HIST03] showed the undecidability of the problem of obtaining the certain answers for a query $q \in CQ$ on a PDMS of *arbitrary* topology, where all the settings $\mathcal{S}$ of the PDMS are such that $\mathcal{S} \in CQ$. Contrast this to the case of just one mapping setting, which is not undecidable, but coNP-complete. They also show that if the topology of the PDMS has no cycles, then the problem becomes decidable. However, such a global constraint on the network topology clearly violates the principles of PDMS.

Calvanese et al. [CGLR04] propose an approach for which obtaining the certain answers in a PDMS of arbitrary topology is decidable (and, in some cases, tractable). However, this comes at the expense of modifying the semantics of certain answers, using epistemic, rather than first-order, interpretations. The new semantics has a more restricted expressive power than the one we use here; it remains to be seen whether it is appropriate for modelling mapping settings.

5.1 Mapping Composition

The *mapping composition problem* is the following: given two mapping settings $\mathcal{S}_{12} = \langle \mathbf{S}_1, \mathbf{S}_2, \mathcal{M}_{12} \rangle$ and $\mathcal{S}_{23} = \langle \mathbf{S}_2, \mathbf{S}_3, \mathcal{M}_{23} \rangle$, obtain a setting $\mathcal{S}_{13} = \langle \mathbf{S}_1, \mathbf{S}_3, \mathcal{M}_{13} \rangle$ such that is $\mathcal{M}_{13}$ is "equivalent" to the successive applications of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$. The problem is not only relevant in peer-to-peer data mediation, but also in other applications, such as model management [Ber03] and schema evolution.

The composition problem was first studied by Madhavan and Halevy [MH03], who proposed a semantics that takes a class of queries as parameter. They focus on the class of conjunctive queries, and show that even for open settings such that $\mathcal{S} \in CQ$, the composition may not be expressible with a setting $\mathcal{S} \in CQ$ that has finite set of rules. However, they show that such a setting can be found if the queries are assumed to be in a restricted class of conjunctive queries, namely CQ_k (conjunctive queries with a fixed number of variables).

A later paper, by Fagin et al. [FKPT04], gives a semantics which is more robust, since it is not relative to a class of queries. The semantics of the problem is defined as follows. First, they define the *composition query* `CompQuery` as the set $\{(I_1, I_3) :$ there exists some instance I_2 such that I_2 is a solution for I_1 in $\mathcal{S}_{12}$, and I_3 is a solution for I_2 in $\mathcal{S}_{23}\}$. Then, they define a *composed setting* $\mathcal{S}_{13}$ as a setting such that for every instance I_1 over $\mathbf{S}_1$, I_3 is a solution for I_1 in $\mathcal{S}_{13}$ iff $(I_1, I_3) \in \texttt{CompQuery}$.

Initially, the paper focuses on open settings such that $\mathcal{S} \in CQ$. They show that if $\mathcal{M}_{12}$ has *full* rules (i.e., with no existential quantifiers), then the composed setting $\mathcal{S}_{13}$ is such that $\mathcal{S}_{13} \in CQ$. However, if $\mathcal{M}_{12}$ has arbitrary rules, there may be no composed setting $\mathcal{S}$ such that $\mathcal{S} \in CQ$ (not even with an infinite number of rules).

To amend the negative results, they introduce a new language for the mapping rules, *second-order tgds* (SOtgds), which are a class of existential second-order formulas with function symbols. Notice that we already presented a (restricted) form of second-order tgds in the context of the inverse rules algorithm (where Skolem functions where employed). Fagin et al. show that, if $\mathcal{S}_{12} \in CQ$ and $\mathcal{S}_{23} \in CQ$, there always exists a composed setting whose mapping rules are second-order tgds. The same holds when $\mathcal{S}_{12} \in SOtgds$ and $\mathcal{S}_{23} \in SOtdgs$, so second-order tgds are closed with respect to composition.

It is also shown in [FKPT04] that the chase procedure can be extended so that it produces a universal solution in polynomial time for open settings whose rules are second-order tgds. Consequently, the problem of computing the certain answers for $q \in CQ$, $\mathcal{S} \in SOtgds$ is tractable. Notice that this is not a contradiction, since second-order tgds and first-order formulas have incomparable expressive power. The conclusion is that second-order tgds are a well-behaved fragment of second-order logic for the purposes of computing the certain answers.

6 Consistent Query Answering

Consistent query answering is the problem of obtaining "consistent" information from databases that might violate some given integrity constraints. The problem has a direct application to data integration and data exchange, as we illustrate next.

Example 6.1 Let $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$ be a mapping setting. Let $\mathbf{S}$ have the relation symbols S_1`(name,salary)` and S_2`(name,salary)`, and $\mathbf{T}$ have the relation symbol `T(name,salary)`. Assume that $\mathcal{M}$ consists of the following rules:

$$S_1(na, sal) \rightarrow T(na, sal)$$
$$S_2(na, sal) \rightarrow T(na, sal)$$
$$T(na, sal) \wedge T(na, sal') \rightarrow sal = sal'$$

Notice that the third rule is a target rule that actually represents a key dependency in the relation T. Let $I = \{S_1(john, 1000), S_1(mary, 3000), S_2(john, 2000)\}$. Clearly, there is no solution for I in $\mathcal{S}$. However, it might be useful to obtain some "consistent" information from the source. For example, there seems to be no inconsistency in I about the salary of Mary. A reasonable approach is to construct the canonical universal solution with respect to the generation rules of $\mathcal{M}$, and "remember" the target rule (in this case, the key dependency) at the time of query answering.

The canonical universal solution for I with respect to the generation rules is $K = \{T(john, 1000), T(mary, 3000), T(john, 2000)\}$. Let Σ be a set that contains the functional dependency $T : name \rightarrow salary$. Now, we have an instance of the consistent query answering problem, where K is the database that violates the set of constraints Σ. •

The problem of consistent query answering also has applications to peer-to-peer mediation systems. In a PDMS, the constraints of the local and global schemas of a peer may change through time. Clearly, it is undesirable that, as a result of a change, we end up in an "inconsistent" state in which there are no solutions for the PDMS. The semantics used in consistent query answering can help to alleviate this problem.

We now explain the semantics of consistent query answering, which was originally introduced by Arenas et al. [ABC99]. An instance K is *consistent* with respect to a set of integrity constraints Σ if K satisfies Σ in the standard model-theoretic sense, i.e., $K \models \Sigma$. The main notion in the semantics is that of *repairs*. Intuitively, a repair is a consistent instance that minimally differs from the input inconsistent database.

In the following definition we will use the *symmetric difference* between two instances, which is defined as $\Delta(K, K') = (K - K') \cup (K' - K)$.

Definition 6.2 (repair) *Let K be an instance. We say that an instance $\mathcal{K}$ is a* repair *of K with respect to Σ iff $\mathcal{K} \models \Sigma$, and $\Delta(K, \mathcal{K})$ is minimal under set inclusion in the class of instances that satisfy Σ, that is, there is no instance K' such that $K' \models \Sigma$ and $\Delta(K, K') \subset \Delta(K, \mathcal{K})$.*

Notice that if K is a consistent instance ($K \models \Sigma$), then the only repair of K is K itself.

Example 6.1 (continued) Recall that $K = \{T(john, 1000), T(mary, 3000), T(john, 2000)\}$, and Σ contains the functional dependency $T : name \rightarrow salary$. Clearly, K is inconsistent with respect to Σ. There are two repairs: $\mathcal{K}_1 = \{T(john, 1000), T(mary, 3000)\}$ and $\mathcal{K}_2 = \{(john, 2000), (mary, 3000)\}$. We use the term "repair", as opposed to "minimal repair", because it is standard in the literature [ABC99]. However, notice that all repairs are minimal by definition. For example, $\{T(john, 2000)\}$ and $\{T(mary, 3000)\}$ are not repairs because their distance with respect to I is not minimal under set inclusion. The fact that this minimality condition is imposed on the repairs is crucial. Otherwise, the empty set would trivially be a repair of every instance. •

For query answering, we will consider the space of repairs of the given instance. Therefore, repairs play the same role for consistent answers as solutions do for certain answers.

Definition 6.3 (consistent answer) *Let $\mathbf{R}$ be a schema. Let K be an instance, possibly not satisfying a set Σ of integrity constraints. Let q be a query over $\mathbf{R}$. We say that a tuple $\mathbf{t}$ is a* consistent answer *with respect to Σ if $\mathbf{t} \in q(\mathcal{K})$, for every repair $\mathcal{K}$ of K with respect to Σ. We denote this as $\mathbf{t} \in \mathbf{consistent}(q, K, \Sigma)$.*

Example 6.1 (continued) Let $q_1(e) = \exists s : T(e, s)$. The consistent answers for q_1 on K are $\{(john), (mary)\}$. Let $q_2(e, s) = R(e, s)$. The consistent answer for q_2 on K is $\{(mary, 3000)\}$. Notice that the tuples $(john, 1000)$ and $(john, 2000)$ do not appear in the set of consistent answers. The reason is that neither of these tuples are present in *both* repairs. Intuitively, this reflects the fact that John's salaries are inconsistent data. •

In this context, the goal of query rewriting is, given a query q, to obtain a query q' such that $q'(K) = \mathbf{consistent}(q, K, \Sigma)$ for every instance K over $\mathbf{S}$. Notice that the only difference with the notion of query rewriting for certain answers is that both the input and rewritten queries are over the same schema.

The problem of retrieving consistent answers from inconsistent databases has only recently received some attention. Lin and Mendelzon [LM98] proposed an operator for merging databases which are potentially inconsistent with respect to a set of constraints. Their semantics differs fundamentally from that of repairs, since it is based on a majority criterion, rather than set inclusion.

To the best of our knowledge, the only approach that is based on first-order query rewriting is Arenas et al. [ABC99]. However, their rewriting applies to a very restricted class of queries, quantifier-free conjunctive queries (i.e., queries without existential quantifiers). This class is not very practical, since it does not allow projections in the queries.

There are approaches in the literature which rewrite large classes of queries, but into logic programs rather than first-order queries ([CLR03], [BB03a], [BB03b] among others). The most general proposal is the one given by Barcelo and Bertossi [BB03a], which deals with arbitrary first-order queries. However, all these approaches are mostly concerned with finding correct logic programs, rather than with producing efficient solutions. In particular, the logic programs proposed in the literature do not run in polynomial time.

7 Putting it all together

We now present some results that establish connections between the problems studied in the previous sections. The results of this section are new, and their proofs are given in the Appendix. In particular we show that:

- the functional dependencies of the target schema can be expressed as restriction rules,
- in some cases, the problem of obtaining consistent answers can be reduced to the problem of obtaining certain answers from a mapping setting,
- the problem of obtaining the certain answers to a query can be reduced to the problem of finding whether there exists a solution for a setting,
- the problem of finding whether there exists a solution for an arbitrary setting can be reduced to the composition query problem for two open settings.

Before establishing these connections, we give some auxiliary results. We define the *small subset property*, and show that every setting $\mathcal{S} \in UCQ$ has this property. Notice that the small subset property is similar to the small model property used in complexity theory, with the additional requirement that the small model must be a *subset* of the larger one.

Definition 7.1 *Let $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$ be a mapping setting. We say that $\mathcal{S}$ has the* small subset property *if for every instance I, J is a solution for I in $\mathcal{S}$ if and only if there exists $J' \subseteq J$ such that J' is a solution for I in $\mathcal{S}$, and the size of J' is polynomially bounded by the size of I.*

Lemma 7.2 *Let $\mathcal{S} \in UCQ$. Then $\mathcal{S}$ has the small subset property.*

When we give a polynomial-time reduction from a setting $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$ to a setting $\mathcal{S}' = \langle \mathbf{S}', \mathbf{T}', \mathcal{M}' \rangle$, we have to construct an instance I' over $\mathbf{S}'$ for every given instance I over $\mathbf{S}$. The size of I' must be polynomially bounded by the size of I. If $\mathcal{M}'$ has restriction rules, it might be necessary to add to I' tuples with constants that are not present in I, but in the solutions. However, there is an infinite number of solutions, and there is no bound on their size. In the following lemma, we show that we can exploit the small subset property to overcome this problem.

Lemma 7.3 *Let $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$ be a mapping setting that has the small subset property. Let I be a source instance. Let q be a monotone query. Then, there exists a set C such that the size of C is polynomially bounded by the size of I and $\mathbf{t} \in \mathbf{certain}(q, I, \mathcal{S})$ iff $\mathbf{t} \in q(J)$ for every solution J for I in $\mathcal{S}$ such that the active domain of J is included in C.*

Now we are ready to establish the connections between the different problems. First, we show that the functional dependencies of the target schema can be expressed as restriction rules.

Theorem 7.4 *Let $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$ be a mapping setting that has the small subset property. Let F_{tar} be a set of target rules that specify functional dependencies on $\mathbf{T}$. Let q be a monotone query. Then, there exists a set F_{res} of restriction rules and a setting $\mathcal{S}' = \langle \mathbf{S}', \mathbf{T}, (\mathcal{M} - F_{tar}) \cup F_{res} \rangle$ such that for every source instance I over $\mathbf{S}$, there exists I' over $\mathbf{S}'$ where:*

- *the size of I' is polynomially bounded by the size of I, and*
- $\mathbf{t} \in \mathbf{certain}(q, I, \mathcal{S})$ *iff* $\mathbf{t} \in \mathbf{certain}(q, I', \mathcal{S}')$.

Second, we show that the problem of obtaining consistent answers with respect to a particular class of constraints can be reduced to the problem of obtaining certain answers from a mapping setting. For the moment, we have been able to prove this for sets of constraints that consist of one functional dependency per relation of the schema. It is still open whether it is possible to establish a similar connection for other kinds of constraints.

Theorem 7.5 *Let $\mathbf{R}$ be a schema, and Σ be a set that consists of at most one functional dependency per relation of $\mathbf{R}$. Let q be a monotone query. Then, there exists a setting $\mathcal{S} = \langle \mathbf{R}, \mathbf{R}, \mathcal{M} \rangle$ such that $\mathcal{S} \in CQ$, and for every source instance I over $\mathbf{R}$ and tuple $\mathbf{t}$, $\mathbf{t} \in \mathbf{certain}(q, I, \mathcal{S})$ iff $\mathbf{t} \in \mathbf{consistent}(q, I, \Sigma)$.*

Third, we show that the problem of obtaining the certain answers to a query can be reduced to the problem of finding whether there exists a solution for a setting. This suggests that studying the latter problem would be very productive since the results could be reused for the former.

Theorem 7.6 *Let $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$ be a mapping setting that has the small subset property. Let $q \in CQ$. Let $\mathbf{t}$ be a tuple. Then, there exists a restriction rule r and a setting $\mathcal{S}' = \langle \mathbf{S}', \mathbf{T}, \mathcal{M} \cup \{r\} \rangle$ such that $\mathcal{S}' \in CQ$, and for every source instance I over $\mathbf{S}$, there exists I' over $\mathbf{S}'$ where:*

- *the size of I' is polynomially bounded by the size of I, and*
- $\mathbf{t} \in \mathbf{certain}(q, I, \mathcal{S})$ *iff there is no solution for I' in $\mathcal{S}'$.*

Finally, we show that the problem of finding whether there exists a solution for a mapping setting can be reduced to the composition query problem for two open settings. This suggests that the composition problem is in fact a generalization of the notion of mapping settings.

Proposition 7.7 *Let $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \mathcal{M} \rangle$ be a mapping setting. Let $\mathcal{M} = F_{gen} \cup F_{res}$, where F_{gen} is a set of generation rules, and F_{res} is a set of restriction rules. Let $\mathcal{S}_{12} = \langle \mathbf{S}, \mathbf{T}, F_{gen} \rangle$ and $\mathcal{S}_{23} = \langle \mathbf{T}, \mathbf{S}, F_{res} \rangle$. Let I be an instance over $\mathbf{S}$. Let* `Compquery` *be the composition query for $\mathcal{S}_{12}$ and $\mathcal{S}_{23}$, as defined in Section 5.1. Then, there exists a solution for I in $\mathcal{S}$ iff $(I, I) \in$* `Compquery`.

8 Conclusions

We have presented a general framework for schema mapping, based on the semantics of certain answers. We have also shown how to cast some problems that arise in several data management applications in terms of this framework. Finally, we have proved some results that reveal underlying connections among these problems.

The semantics of certain answers can be considered to be conservative, in the sense that a tuple appears in the answer to a query only if it appears in *every* solution. A complementary semantics is that of *possible answers*. Under this semantics, a tuple appears in the answer to a query only if it appears in *at least some* solution. In many applications, it would be desirable to have both semantics available to the user. For instance, in consistent query answering, the fact that a tuple is in the possible answers but not in certain answers may help to spot inconsistencies in the data. Surprisingly, the notion of possible answers has not received much attention in the literature. A reason for this might be the fact that in open mapping settings, which have been the most studied so far, every tuple is trivially a possible answer.

In between certain and possible answers, there is a whole spectrum of semantics that can be defined. For example, we may say that a tuple is in the answer if it appears in the *majority* of the solutions; or that some solutions are *preferred* to others. Although there has been some work on defining semantics based on probabilistic notions([MM01, CGM01]), much remains to be done in this direction.

Of course, the theoretical results surveyed here are of interest only if they are useful for the development of tools based on the semantics of certain answers. Most tools developed so far are in the domain of data integration, in particular for open settings. This is no surprise since, as we have shown in the paper, most tractability and rewritability results assume the input is an open setting. In other areas, such as peer-to-peer mediation systems or consistent query answering, restriction rules are essential. For settings with both generation and restriction rules, most of the results that we surveyed here are negative. So, a natural question is whether the notion of certain answers is going to have any practical impact.

In order to answer this question, we point out that all the negative results surveyed here concern *classes* of queries. As we argued in Section 2.2, it suffices to have one "difficult" query or setting in a class to make it intractable (or undecidable). Therefore, negative results do not preclude the existence of large, and practical, subclasses, for which the problem is tractable. The ultimate goal would be to have a *complete classification* such that, given a query q and a setting $\mathcal{S}$, tells us whether the problem of obtaining the certain answers for q in $\mathcal{S}$ is tractable (or rewritable). If efficient algorithms are proposed for the positive cases of the classification, we would have a solid basis towards the practical development of systems based on the certain answers semantics. We propose to do this as part of our thesis work.

References

[ABC99] M. Arenas, L. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proc. ACM PODS*, pages 68–79, 1999.

[ABFL04] M. Arenas, P. Barcelo, R. Fagin, and L. Libkin. Locally consistent transformations and query answering in data exchange. To be appear in PODS, 2004.

[AD98] S. Abiteboul and O. M. Duschka. Complexity of answering queries using materialized views. In *PODS*, pages 254–263, 1998.

[AGK91] S. Abiteboul, G. Grahne, and P. Kanellakis. On the representation and querying of sets of possible worlds. *Theoretical Computer Science*, (78):157–187, 1991.

[ALM02] F. Afrati, C. Li, and P. Mitra. Answering queries using views with arithmetic comparisons. In *PODS*, 2002.

[BB03a] P. Barcelo and L. Bertossi. Logic programs for querying inconsistent databases. In *Proc. PADL*, Springer LNCS 2562, pages 208–222, 2003.

[BB03b] L. Bravo and L. Bertossi. Logic programs for consistently querying data integration systems. In *IJCAI*, 2003.

[Ber03] P. Bernstein. Applying model management to classical meta data problems. *Conf. on Innovative Database Research (CIDR)*, 2003.

[BGK+02] P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for Peer-to-Peer computing: A vision. In *Fifth International Workshop on the Web and Databases (WebDB)*, pages 89–94, 2002.

[BLN86] C. Batini, M. Lenzerini, and S. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323 – 364, 1986.

[BLR97] C. Beeri, A. Levy, and M. Rousset. Rewriting queries using views in description logics. In *PODS*, 1997.

[CGLR04] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Logical foundations of peer-to-peer data integration. To appear in PODS, 2004.

[CGLV00] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Vardi. Answering regular path queries using views. In *ICDE*, pages 389–398, 2000.

[CGM01] K. Chen-Chuan Chang and H. Garcia-Molina. Approximate query mapping: Accounting for translation closeness. *The VLDB Journal*, 10(2-3):155–181, 2001.

[CK86] S. Cosmadakis and P. Kanellakis. Functional and inclusion dependencies: A graph-theoretic approach. *Advances in Computing Research*, 3:163–184, 1986.

[CKPS95] S. Chaudhuri, R. Krishnamurthy, S. Potamianos, and K. Shim. Optimizing queries with materialized views. In *ICDE*, pages 190–200, 1995.

[CLR03] A. Calì, D. Lembo, and R. Rosati. Query rewriting and answering under constraints in data integration systems. In *IJCAI*, 2003.

[CM77] A. Chandra and P. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, pages 77–90, 1977.

[DGL00] O. Duschka, M. Genesereth, and A. Y. Levy. Recursive query plans for data integration. *Journal of Logic Programming*, 43(1):49–73, 2000.

[EF95] H. Ebbinghaus and J. Flum. *Finite Model Theory.* Springer-Verlag, 1995.

[FKMP03] R. Fagin, P. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. In *ICDT*, pages 207–224, 2003.

[FKP03] R. Fagin, P. Kolaitis, and L. Popa. Data exchange: getting to the core. In *PODS*, 2003.

[FKPT04] R. Fagin, P. Kolaitis, L. Popa, and W. Tan. Composing schema mappings: Second-order dependencies to the rescue. To appear in PODS, 2004.

[FLM99] M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *AAAI/IAAI*, pages 67–73, 1999.

[GHI+01] S. Gribble, A. Halevy, Z. Ives, M. Rodrig, and D. Suciu. What can peer-to-peer do for databases? In *Fourth International Workshop on the Web and Databases*, 2001.

[GL00] S. Grumbach and Tinini L. On the content of materialized aggregate views. In *PODS*, 2000.

[GM99] G. Grahne and A. Mendelzon. Tableau techniques for querying information sources through global schemas. In *ICDT*, pages 332–347. Springer-Verlag, 1999.

[Hal01] A. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.

[HIST03] A. Halevy, Z. Ives, Dan Suciu, and Igor Tatarinov. Schema mediation in peer data management systems. In *ICDE*, 2003.

[IFF+99] Z. Ives, D. Florescu, M. Friedman, A. Levy, and D. Weld. An adaptive query execution system for data integration. In *SIGMOD*, 1999.

[IL84] T. Imielinski and W. Lipski. Incomplete information in relational databases. *Journal of ACM*, 31(4):761–791, 1984.

[KAM03] A. Kementsietsidis, M. Arenas, and R. Miller. Mapping data in peer-to-peer systems: Semantics and algorithmic issues. In *SIGMOD*, 2003.

[KKM+03] V. Kantere, I. Kiringa, J. Mylopoulos, A. Kementsietsidis, and M. Arenas. Coordinating peer databases using eca rules. In *Information Systems, and Peer-to-Peer Computing: First International Workshop*, 2003.

[Len02] Maurizio Lenzerini. Data integration: A theoretical perspective. In *PODS*, 2002.

[Lev00] A. Levy. Logic-based techniques in data integration. *Logic Based Artificial Intelligence*, 2000.

[LM98] J. Lin and A. Mendelzon. Merging databases under constraints. *Int. J. of Cooperative Information Systems*, 1(7):55–76, 1998.

[LMSS95] A. Levy, A. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *PODS*, 1995.

[LRO96] A. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous information sources using source descriptions. In *VLDB*, 1996.

[MH03] J. Madhavan and A. Halevy. Composing mappings among data sources. In *VLDB Conference*, 2003.

[MM01] A. Mendelzon and G. Mihaila. Querying partially sound and complete data sources. In *PODS*, 2001.

[MMS79] D. Maier, A. Mendelzon, and Y. Sagiv. Testing implications of data dependencies. *ACM Transactions on Database Systems (TODS)*, 4(4):455–469, 1979.

[PL00] R. Pottinger and A. Levy. A scalable algorithm for answering queries using views. In *VLDB*, 2000.

[PVM+02] L. Popa, Y. Velegrakis, R. Miller, M. Hernandez, and R. Fagin. Translating heterogeneous web data. In *Proceedings of the 28th VLDB Conference*, 2002.

[Shm93] O. Shmueli. Equivalence of datalog queries is undecidable. *Journal of Logic Programming*, (15):231–241, 1993.

[VMP03] Y. Velegrakis, R. J. Miller, and L. Popa. Mapping adaptation under evolving schemas. In *VLDB*, 2003.

A Appendix

Proof of Proposition 2.7

Proof We construct $\mathcal{S}_{norm}$ as follows. Let $\{r_1, \ldots, r_n\}$ be the generation rules of $\mathcal{M}$. Let $\{r_{n+1}, \ldots, r_{n+m}\}$ be the restriction rules of $\mathcal{M}$. Let $\mathbf{S}_{norm} = \mathbf{S} \cup \{R_i : 1 \leq i \leq n+m\}$. The mapping $\mathcal{M}_{norm}$ is obtained as follows:

- Add all target rules of $\mathcal{M}$.
- For each generation rule r_i of $\mathcal{M}$, of the form $\forall \mathbf{x} : \phi_s(\mathbf{x}) \rightarrow \psi_t(\mathbf{x})$, add a rule $\forall \mathbf{x} : R_i(\mathbf{x}) \rightarrow \psi_t(\mathbf{x})$.
- For each restriction rule r_i of $\mathcal{M}$, of the form $\forall \mathbf{x} : \phi_t(\mathbf{x}) \rightarrow \psi_s(\mathbf{x})$, add a rule $\forall \mathbf{x} : \phi_t(\mathbf{x}) \rightarrow R_i(\mathbf{x})$.

Let q be a query, and I be an instance over $\mathbf{S}$. We shall build an instance I' over $\mathbf{S}_{norm}$ as follows.

- For every relation symbol R of $\mathbf{S}$, we let the extension of R in I' be the extension of R in I.
- For every generation rule r_i of $\mathcal{M}$, of the form $\forall \mathbf{x} : \phi_s(\mathbf{x}) \rightarrow \psi_t(\mathbf{x})$, we let the extension of R_i in I' be the result of evaluating $\phi_s(\mathbf{x})$ on I.
- For every restriction rule r_i of $\mathcal{M}$ of the form $\forall \mathbf{x} : \phi_t(\mathbf{x}) \rightarrow \psi_s(\mathbf{x})$, we let the extension of R_i in I' be the result of evaluating $\psi_s(\mathbf{x})$ on I.

It is easy to check that J is a solution for I in $\mathcal{S}$ iff J is a solution for I' in $\mathcal{S}_{norm}$. $\square$

Proof of Lemma 7.2

Proof Assume that J is a solution for I in $\mathcal{S}$. We shall build an instance J_{small} as follows. For each generation rule of $\mathcal{M}$ of the form $\forall \mathbf{x} : \phi_s(\mathbf{x}) \rightarrow \psi_t(\mathbf{x})$, and for every tuple $\mathbf{t}$ such that $I \models \phi_s(\mathbf{t})$, we choose a *witness* set J' such that $J' \subseteq J$ and $J' \models \psi_t(\mathbf{t})$. We add J' to J_{small}. It is easy to see that we can choose J' in such a way that it has constant size (bounded by the size of $\psi_t(\mathbf{x})$).

Clearly, $\langle I, J_{small} \rangle$ satisfies the generation rules of $\mathcal{M}$. Also, since $J_{small} \subseteq J$ and J is a solution for I, $\langle J_{small}, I \rangle$ satisfies the restriction rules of $\mathcal{M}$. Thus J_{small} is a solution for I in $\mathcal{S}$. It is easy to see that the size of J_{small} is polynomial in the size of I. $\square$

Proof of Lemma 7.3

Proof Let s be the number of tuples of I. Since $\mathcal{S}$ has the small subset property, there exists a polynomial function p such that, for every solution J for I, there exists a solution $J' \subseteq J$ where the size of J' is bounded by $p(s)$. Let C be a set that contains the active domain of I and $p(s)$ new constants.

($\Rightarrow$) Follows directly by definition of certain answers.

($\Leftarrow$) Assume $\mathbf{t} \notin \mathbf{certain}(q, I, \mathcal{S})$. Then, there exists some solution J for I in $\mathcal{S}$ such that $\mathbf{t} \notin q(J)$. Since $\mathcal{S}$ has the small subset property, there exists $J' \subseteq J$ such that J' is a solution for I in $\mathcal{S}$, and the size of J' is bounded by $p(s)$. Now, let us obtain a set J'' from J' as follows. For each constant c that appears in some tuple of J', but is not in the active domain of I, we replace every occurrence of c in J' by a new constant c' from C. Since the constants in the active domain of I are not replaced, J'' is solution for I in $\mathcal{S}$. By hypothesis, $\mathbf{t} \in q(J'')$. Since J'' and J' are isomorphic, $\mathbf{t} \in q(J')$. Since $\mathbf{t} \in q(J')$, $\mathbf{t} \notin q(J)$ and $J' \subseteq J$, q is not a monotone query; contradiction. $\square$

Proof of Theorem 7.4

Proof Let Eq be a relation symbol. Let $\mathbf{S}' = \mathbf{S} \cup \{Eq\}$. For every target rule of F_{tar}, which is of the form

$$T_i(\mathbf{x}, y_1, \ldots, y_n, \mathbf{z}) \wedge T_i(\mathbf{x}, y'_1, \ldots, y'_n, \mathbf{z}') \rightarrow (y_1 = y'_1 \wedge \ldots \wedge y_n = y'_n)$$

we add a restriction rule to F_{res} of the form

$$T_i(\mathbf{x}, y_1, \ldots, y_n, \mathbf{z}) \wedge T_i(\mathbf{x}, y'_1, \ldots, y'_n, \mathbf{z}') \rightarrow (Eq(y_1, y'_1) \wedge \ldots \wedge Eq(y_n, y'_n))$$

Let I be an instance over $\mathbf{S}$. By Lemma 7.3, there exists a set C such that the size of C is polynomially bounded by the size of I and $\mathbf{t} \in \mathbf{certain}(q, I, \mathcal{S})$ iff $\mathbf{t} \in q(J)$ for every solution J for I in $\mathcal{S}$ such that the active domain of J is included in C. Let $I' = I \cup \{Eq(x, y) : x \in C, y \in C \text{ and } x = y\}$.

($\Rightarrow$) Let J' be a solution for I' in $\mathcal{S}'$. Since $J' \models F_{res}$, $J' \models F_{tar}$. Thus, J' is a solution for I in $\mathcal{S}$. Since $\mathbf{t} \in \mathbf{certain}(q, I, \mathcal{S})$, $\mathbf{t} \in q(J')$.

($\Leftarrow$) Let J be a solution for I in $\mathcal{S}$ such that the active domain of J is included in C. Clearly, J is a solution for I' in $\mathcal{S}'$. Since $\mathbf{t} \in \mathbf{certain}(q, I', \mathcal{S}')$, $\mathbf{t} \in q(J)$.

Since $\mathbf{t} \in q(J)$ for every solution J for I in $\mathcal{S}$ such that the active domain of J is included in C, by Lemma 7.3, $\mathbf{t} \in \mathbf{certain}(q, I, \mathcal{S})$.

□

Proof of Theorem 7.5

Proof We create a set F_{gen} of generation rules as follows. For every relation R of $\mathbf{R}$ such that R has no functional dependencies in Σ, we add the following rule to F_{gen} (to avoid confusion, we will denote as R_s the occurence of relation symbol R in the source schema):

$$R_s(\mathbf{x}) \rightarrow R(\mathbf{x})$$

For every relation R of $\mathbf{R}$ such that there is a functional dependency $x_1, \ldots, x_n \rightarrow x_{n+1}, \ldots, x_{n+m}$ in Σ, we add the following generation rule to $\mathcal{M}$:

$$R_s(\mathbf{x}) \rightarrow \exists x_{n+1}, \ldots, x_{n+m} : R(\mathbf{x})$$

We also create a set F_Σ, by adding one rule per functional dependency of Σ. The rule may either be the obvious target rule that simulates the functional dependency, or a restriction rule if we choose to perform the tranformation of Theorem 7.4.

Finally, we create a set F_{res} of restriction rules as follows. For every relation R of $\mathbf{R}$, we add

$$R(\mathbf{x}) \rightarrow R_s(\mathbf{x})$$

Let $\mathcal{M} = F_{gen} \cup F_\Sigma \cup F_{res}$. Let I be an instance over $\mathbf{R}$.

($\Rightarrow$) Let $\mathcal{I}$ be a repair of I wrt Σ. Since $\mathcal{I} \models \Sigma$, $\mathcal{I} \models F_\Sigma$. Since the violations of functional dependencies can be fixed only by deleting tuples, $\mathcal{I} \models F_{res}$. Assume $\langle I, \mathcal{I} \rangle \not\models F_{gen}$. Let $r \in F_{gen}$ be such that $\langle I, \mathcal{I} \rangle \not\models r$. Let $R_s(\mathbf{a})$ be a tuple such that $R_s(\mathbf{a}) \in I$ and $\langle I, \mathcal{I} \rangle \not\models F_{gen}[\mathbf{a}]$. Let $\mathcal{I}' = \mathcal{I} \cup \{R_s(\mathbf{a})\}$. Since $\langle I, \mathcal{I} \rangle \not\models r$, $\mathcal{I}'$ does not violate Σ. Since $\mathcal{I} \subset \mathcal{I}' \subseteq I$, $\mathcal{I}$ is not a repair of I wrt Σ; contradiction. We conclude that $\mathcal{I} \models F_{gen}$, and therefore $\mathcal{I}$ is a solution for I in $\mathcal{S}$. Since $\mathbf{t} \in \mathbf{certain}(q, I, \mathcal{S})$, $\mathbf{t} \in q(\mathcal{I})$.

($\Leftarrow$) Let J be a solution for I in $\mathcal{S}$. Clearly, $J \models \Sigma$. Assume there is a repair J' of I wrt Σ such that $J' \supset J$. Since $\langle I, J \rangle \models F_{gen}$ and $\langle I, J' \rangle \models F_{res}$, there are at least two tuples in J' that violate a functional dependency. Thus J' is not a repair; contradiction. We conclude that there exists a repair J' of I wrt Σ such that $J' \subseteq J$. Since $\mathbf{t} \in \mathbf{consistent}(q, I, \Sigma)$, $\mathbf{t} \in q(J')$. Since q is monotone, $\mathbf{t} \in q(J)$.

$\square$

Proof of Theorem 7.6

Proof Let q' be a boolean query obtained by replacing the free variables of q with the corresponding constants from $\mathbf{t}$. Let q' be of the form

$$\exists \mathbf{x}_1, \ldots, \mathbf{x}_n : R_1(\mathbf{x}_1) \wedge \ldots \wedge R_n(\mathbf{x}_n) \wedge \bigwedge_{i=1 \ldots m} x_{i_1} = x_{i_2}$$

where all the variables that appear in the equality literals are from the tuples $\mathbf{x}_1, \ldots, \mathbf{x}_n$. We will assume, without loss of generality, that all the variables in $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are distinct. Let $Ineq$ be a relation symbol. Let $\mathbf{S}' = \mathbf{S} \cup \{Ineq\}$. Let r be the following restriction rule:

$$\forall \mathbf{x}_1, \ldots, \mathbf{x}_n : R_1(\mathbf{x}_1) \wedge \ldots \wedge R_n(\mathbf{x}_n) \rightarrow Ineq(\mathbf{x}_1, \ldots, \mathbf{x}_n)$$

Let I be an instance over $\mathbf{S}$. By Lemma 7.3, there exists a set C such that the size of C is polynomially bounded by the size of I and $\mathbf{t} \in \mathbf{certain}(q, I, \mathcal{S})$ iff $\mathbf{t} \in q(J)$ for every solution J for I in $\mathcal{S}$ such that the active domain of J is included in C. Let $I' = I \cup \{Ineq(\mathbf{t}_1, \ldots, \mathbf{t}_n)$: the values of $\mathbf{t}_1, \ldots, \mathbf{t}_n$ are from C and $\bigvee_{i=1 \ldots m} x_{i_1} \neq x_{i_2}[\mathbf{x}_1/\mathbf{t}_1, \ldots, \mathbf{x}_n/\mathbf{t}_n]\}$.

($\Rightarrow$) Assume that there is a solution J' for I' in $\mathcal{S}'$. Clearly, J' is a solution for I in $\mathcal{S}$. Since $J' \models r$, $q'(J') = \mathbf{false}$, and $t \notin q(J')$. Therefore, $\mathbf{t} \notin \mathbf{certain}(q, I, \mathcal{S})$; contradiction.

($\Leftarrow$) Let J be a solution for I in $\mathcal{S}$ such that the active domain of J is included in C. By hypothesis, J is not a solution for I' in $\mathcal{S}'$. Thus, it is clear that $J \not\models r$. Thus, $q'(J) = \mathbf{true}$, and $t \in q(J)$.

Since $\mathbf{t} \in q(J)$ for every solution J for I in $\mathcal{S}$ such that the active domain of J is included in C, by Lemma 7.3, $\mathbf{t} \in \mathbf{certain}(q, I, \mathcal{S})$.

$\square$