UNIVERSITY OF CALIFORNIA
SANTA CRUZ

MANAGING INCONSISTENCIES IN DATA EXCHANGE

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Richard L. Halpert

December 2016

The Dissertation of Richard L. Halpert
is approved:

Phokion G. Kolaitis, Co-Chair

Balder ten Cate, Co-Chair

Peter Alvaro

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Managing Inconsistencies in Data Exchange

by

Richard L. Halpert

Data exchange is the problem of transforming data structured under one schema, called the source schema, into data structured under another schema, called the target schema, in such a way that pre-specified constraints on the two schemas are satisfied. Queries may then be posed against the target schema, where the traditional semantics of data exchange, called certain answers, give those answers that hold on all acceptable choices of target instance (that is, all "solutions"). In a data exchange setting with target constraints, it is often the case that a given source instance has no solutions. Intuitively, this happens when data sources contain inconsistent or conflicting information that is exposed by the target constraints. In such cases, the traditional semantics of target queries trivialize.

In this work, we explore a new framework, called exchange-repairs, that gives meaningful answers in such cases. Informally, an exchange-repair of a source instance is a pair of instances: a source instance that differs minimally from the original, but has a solution, along with one such solution. Exchange-repairs give rise to a natural notion of exchange-repair certain (XR-certain) answers; they are the answers that hold on every exchange-repair.

We first explore the problem of computing the XR-certain answers to conjunctive queries. We show limited circumstances under which the XR-certain query answers can be expressed as the consistent query answers (in the sense of standard database repairs) on the same instance, but using another query and set of constraints. We also give a translation to disjunctive logic programming (DLP) that is

applicable for a much larger class of XR-certain query answering problems. Next, we study how to effectively perform XR-certain query answering for practical data exchange settings. We evaluate the use of modern, sophisticated DLP solvers, and develop a principled approach to optimizing their use for XR-certain query answering. Then, we investigate approximations of XR-certain semantics, and draw close parallels to those proposed for the problem of inconsistency-tolerant ontology based data access (OBDA). However, we find that the approximations known to be tractable for constraint languages common in OBDA research prove to have computational complexity no better than XR-certain semantics for the types of constraints typically studied in data exchange.

This work is dedicated to Leah, for her encouragement and love, and to Evelyn, for filling my days with excitement.

Acknowledgments

First and foremost, I wish to thank my advisors, Balder ten Cate and Phokion Kolaitis, for their relentless pursuit of precision and correctness, for their patience with me throughout my studies, and for their mentorship and caring during challenging times. Thank you, also, to Peter Alvaro for enthusiastically diving into these materials in order to serve on my reading committee. Finally, thank you to my colleagues at Ingenuity and ClearLabs for granting me the flexibility to be successful in my academic life.

Chapter 1

Introduction

Data exchange is the problem of transforming data structured under one schema, called the source schema, into data structured under a different schema, called the target schema, in such a way that pre-specified constraints on these two schemas are satisfied. Data exchange is a ubiquitous data inter-operability task that has been explored in depth during the past decade (see [5]). This task is formalized with the aid of schema mappings, which consist of a source schema, a target schema, a set of constraints between the source and target, and a further set of constraints on the target. The most thoroughly investigated schema mappings are those specified by tuple-generating dependencies (tgds, a generalization of inclusion constraints) and equality-generating dependencies (egds, a generalization of key constraints) [32]. Definitions of each are given in Chapter 2. A schema mapping is represented as a quadruple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$, where $\mathbf{S}$ is the source schema, $\mathbf{T}$ is the target schema, Σ_{st} is a set of source-to-target tgds, and Σ_{t} is a set of egds and tgds on the target schema. An example of such a schema mapping follows, along with a query over the target schema:

Example 1.1. A schema mapping $\mathcal{M}$ specified by tgds and egds, and a target

query. In this example, the egd is actually a key constraint and there are no target tgds.

$$\Sigma_{\text{st}} = \left\{ \begin{array}{lcl} \texttt{Task\_Assignments}(p,t,d) & \rightarrow & \texttt{Departments}(p,d) \wedge \texttt{Tasks}(p,t) \\ \texttt{Stakeholders\_old}(t,s) & \rightarrow & \texttt{Stakeholders\_new}(t,s) \end{array} \right\}$$

$$\Sigma_{\text{t}} = \left\{ \texttt{Departments}(p,d) \wedge \texttt{Departments}(p,d') \rightarrow d = d' \right\}$$

$$\texttt{boss}(person, stakeholder) = \exists task. \texttt{Tasks}(person, task) \wedge \texttt{Stakeholders\_new}(task, stakeholder)$$

Every schema mapping $\mathcal{M}$ gives rise to two distinct algorithmic problems. The first is the existence and construction of solutions: given a source instance, determine whether a *solution* for it exists (that is, a target instance that, together with the source, satisfies the constraints of $\mathcal{M}$) and, if it does, construct such a solution. The second problem is to compute the *certain answers* of target queries for a given source instance, where the certain answers are those that hold on every solution for the given source. For arbitrary schema mappings specified by tgds and egds, both problems can be undecidable [46]. However, as shown in [32], if the schema mapping obeys a mild structural condition called *weak acyclicity*, then both problems can be solved in polynomial time using the *chase procedure*. Given a source instance, the chase procedure attempts to build a "most general" solution for it by generating facts that satisfy each tgd as needed (using values called *nulls* when a precise constant is not specified), and by equating two nulls or equating a null to a constant, as dictated by the egds. If the chase procedure encounters an egd that equates two distinct constants, then it terminates and reports that no solution exists. Otherwise, it constructs a *universal* solution, which can also be used to compute the certain answers of conjunctive queries in time bounded by a

polynomial in the size of the source instance.

In data exchange settings with a non-empty set of target constraints, it frequently happens that a given source instance has no solution. In particular, this may happen when the source instance at hand contains inconsistencies or conflicting information that is exposed by the target constraints. The standard data exchange frameworks are not able to provide meaningful answers to target queries in such circumstances; in fact, the certain answers to every target query trivialize. In turn, this suggests that alternative semantics for target queries could be obtained by adopting the notions of database *repairs* and *consistent answers* from the study of inconsistent databases (see [9] for an overview). We note that several different types of repairs have been studied in the context of inconsistent databases; the most widely used ones are the *symmetric difference repairs*, which contain as special cases the *subset repairs* and the *superset repairs*.

How can the notions of database repairs and consistent answers be adapted to the data exchange framework? Several different approaches are possible.

One approach, which we call *materialize-then-repair*, is as follows: given a source instance, a target instance is produced by chasing with the source-to-target tgds in Σ_{st} and the target tgds in Σ_{t}, while ignoring the target egds in Σ_{t}. Since the target instance produced this way may very well violate the egds in Σ_{t}, it is treated as an inconsistent instance with respect to the target constraints. Then the answers to target queries are taken to be their consistent answers with respect to the egds in Σ_{t}. Note that a similar approach has been adopted by [10, 15] in the context of data integration. A different approach, which we call *exchange-as-repair*, treats the given source instance as an instance over the combined source and target schemas that is inconsistent with respect to the constraints of the schema mapping, and again uses the notion of consistent answers as the seman-

src

Task_Assignments		
person	task	dept
peter	tpsreport	software
peter	spaceout	software
peter	meetbobs	exec

Stakeholders_old	
task	stakeholder
tpsreport	lumbergh
tpsreport	portman
spaceout	bobs
meetbobs	bobs

tgt

Departments	
person	dept
peter	software
peter	exec

Tasks	
person	task
peter	tpsreport
peter	spaceout
peter	meetbobs

Stakeholders_new	
task	stakeholder
tpsreport	lumbergh
tpsreport	portman
spaceout	bobs
meetbobs	bobs

Figure 1.1: A source instance src and the inconsistent target instance tgt that results from chasing src with the tgds in $\mathcal{M}$.

tics for target queries (this time with respect to all of the constraints of the schema mapping). This approach is in the spirit of [42], where instances in peer data exchange that do not satisfy the schema mapping at hand are treated as inconsistent databases over a combined schema. We will now illustrate that neither of these approaches gives rise to satisfactory semantics for data exchange.

Figure 1.1 gives an example of a target instance that is produced in the materialize-then-repair approach by chasing with the tgds in Example 1.1. Clearly, tgt is inconsistent because it violates the egd in Σ_t. Consider now the subset repair tgt$'$ in Figure 1.2 of our materialized target instance tgt (note that, in this case, symmetric difference repairs coincide with subset repairs). Notice that the repair tgt$'$ places peter in the exec department, yet still has him performing tasks for the software department – the fact that the "tpsreport" and "spaceout" tasks are derived from a tuple placing peter in the software department has been lost. The only other repair of tgt also fails to reflect the shared origin of tuples in the `Tasks` and `Departments` tables, and this disconnect in the materialize-then-repair approach manifests in the consistent answers

tgt′

Departments	
person	dept
~~peter~~	~~software~~
peter	exec

Tasks	
person	task
peter	tpsreport
peter	spaceout
peter	meetbobs

Stakeholders_new	
task	stakeholder
tpsreport	lumbergh
tpsreport	portman
spaceout	bobs
meetbobs	bobs

Figure 1.2: A symmetric-difference-repair tgt′ of tgt w.r.t. Σ_t.

to target queries. In this example, the consistent answers for `boss`(peter, b) are $\{(\text{peter}, \text{bobs}), (\text{peter}, \text{portman}), (\text{peter}, \text{lumbergh})\}$. However, the last two tuples are derived from facts placing peter in the software department, even though in tgt′ he is not.

The situation is no better in the exchange-as-repair approach. Figure 1.1 depicts three repairs of this type (using symmetric difference semantics). While the first two repairs in Figure 1.1 seem reasonable, in the third we have eliminated `Task_Assignments`(peter, spaceout, software), even though our key constraint is already satisfied by the removal of `Task_Assignments`(peter, meetbobs, exec) alone. In this approach, the consistent answers of `boss`(peter, b) are $\emptyset$, despite the intuitive conclusion that peter should be performing tasks for the bobs regardless of which way we fix the department key constraint violation. For symmetric-difference repairs, it is equally valid to satisfy a violated tgd by removing tuples as by adding them. However, in a data exchange setting, we aim to materialize a target instance from scratch, so it would be more natural to satisfy violated tgds by deriving new tuples.

$(\mathrm{src}_1, \mathrm{tgt}_1)$

Task_Assignments		
person	task	dept
~~peter~~	~~tpsreport~~	~~software~~
~~peter~~	~~spaceout~~	~~software~~
peter	meetbobs	exec

Departments	
person	dept
peter	exec

Tasks	
person	task
peter	meetbobs

$(\mathrm{src}_2, \mathrm{tgt}_2)$

Task_Assignments		
person	task	dept
peter	tpsreport	software
peter	spaceout	software
~~peter~~	~~meetbobs~~	~~exec~~

Departments	
person	dept
peter	software

Tasks	
person	task
peter	tpsreport
peter	spaceout

$(\mathrm{src}_3, \mathrm{tgt}_3)$

Task_Assignments		
person	task	dept
peter	tpsreport	software
~~peter~~	~~spaceout~~	~~software~~
~~peter~~	~~meetbobs~~	~~exec~~

Departments	
person	dept
peter	software

Tasks	
person	task
peter	tpsreport

Table 1.1: Three repairs of $(\mathrm{src}, \emptyset)$ (from Figure 1.1) w.r.t. $\Sigma_{\mathrm{st}} \cup \Sigma_{\mathrm{t}}$. The `Stakeholders` tables are omitted for brevity.

1.1 Roadmap

Our research focus is on how to address the limitations noted above. Conceptually, the main contribution is the introduction of the notion of an *exchange-repair solution*. Informally, a source repair for a source instance is another source instance that differs minimally from the first but has a solution, and an exchange-repair solution is a source repair along with one such solution. Exchange-repair solutions give rise to a natural notion of *exchange-repair certain answers* (in short, *XR-certain answers*) for target queries in the context of data exchange, wherein we evaluate the query on every exchange-repair solution, and take the intersection. Note that if a source instance has a solution, then for conjunctive queries, the XR-certain answers coincide with the certain answers. If a source instance has no solutions, then unlike the certain answers, the XR-certain answers are non-trivial and meaningful.

We provide examples demonstrating that these new semantics improve upon both the materialize-then-repair approach and the exchange-as-repair approach discussed earlier. We also produce a detailed comparison of the XR-certain semantics with the main notions of inconsistency-tolerant semantics studied in data integration and in ontology-based data access. This comparison appears in Section 3.1, after we have introduced our framework and presented some basic structural properties of exchange-repairs in Chapter 3.

In Chapter 4, we focus on the problem of computing the XR-certain answers to conjunctive queries. In Section 4.1, we show that for schema mappings specified by source-to-target GAV (global-as-view) dependencies and target egds, the XR-certain answers of conjunctive queries can be rewritten as the consistent answers (in the sense of standard database repairs) of a union of conjunctive queries over the source schema with respect to a set of egds over the source schema,

thus making it possible to use a consistent query-answering system to compute XR-certain answers in data exchange. In contrast, we show that this type of rewriting is not possible for schema mappings specified by source-to-target LAV (local-as-view) dependencies and target egds, nor for schema mappings specified by source-to-target and target GAV dependencies and target egds. In Section 4.2, we examine the case of schema mappings specified by GAV s-t tgds, GAV target tgds, and target egds. We show that computing the XR-certain answers of conjunctive queries in this case can be reduced to cautious reasoning over stable models of a disjunctive logic program. Disjunctive logic programming is a suitable formalism for coping with the intractability of XR-certain query answering because it goes beyond the natural expressiveness of SQL while still remaining in a declarative framework.

Chapter 5 introduces a technique to significantly extend the applicability of our reduction to disjunctive logic programming. The main result is that, for schema mappings consisting of GLAV (global-and-local-as-view) s-t tgds, weakly acyclic sets of GLAV target tgds, and target egds, we show that the XR-certain answers of conjunctive queries can be rewritten as the XR-certain answers of conjunctive queries with respect to a schema mapping consisting of GAV source-to-target tgds, GAV target tgds, and target egds. In fact, we prove the stronger result that such a rewriting is possible for schema mappings specified by a second-order source-to-target tgd, a weakly acyclic second-order target tgd, and a set of target egds.

In Chapter 6, we combine the techniques from Chapters 4 and 5 with a disjunctive logic program solver to produce a reference implementation of XR-certain query answering. We introduce a practical data exchange scenario in computational genomics, and use it as a case study to evaluate the effectiveness of the ref-

erence implementation. We then explore a technique that, for each possible query answer, computes a narrow *focus area* of the source and target instances that is sufficient to decide whether the possible query answer is indeed an XR-certain query answer. This allows us to compute XR-certain by solving many small hard problems rather than one monolithic problem. We present an optimized implementation of XR-certain query answering based on this approach, and evaluate its performance on the same genomics data scenario. Our optimized implementation proves to be ten to one thousand times faster than the reference implementation on this dataset.

Lastly, in Chapter 7, we study several proposed approximation semantics for inconsistency-tolerant query answering from the domain of ontology-based data access. We introduce analogous notions of approximation in our setting, and study their applicability and their complexity. Although each approximation that we study is known to be tractable for various classes of constraints considered in ontology-based data access, we find that the same notions are intractable for the classes of constraints typically considered in data exchange.

Chapter 2

Preliminaries

Our approach to handling inconsistencies in data exchange builds upon foundational work in three areas: relational databases, data exchange, and database repair.

Relational Databases: instances, queries, and homomorphisms

The relational model of databases (introduced by E.F. Codd [22] in 1970) has proved adequate for many use cases in industry, while simultaneously being mathematically rigorous enough for theoretical research. Relational databases are typically described as instances of a schema, where the latter is defined as a collection of relational symbols with designated arities. An instance of the schema is then a collection of interpretations of those relation symbols over some particular domain. In the case of data exchange, this domain typically includes constants (that is, values such as might usually be inserted into a database) and "labeled nulls" (representatives of unknown values). The purpose of labeled nulls will become clear in the next section.

Fix an infinite set Const of elements, and an infinite set Nulls of elements such that Const and Nulls are disjoint. A *schema* $\mathbf{R}$ is a set $(R_1, \ldots R_n)$, where

$R_1, \ldots, R_n$ are relation symbols, each with a designated arity. An $\mathbf{R}$-*instance* I is a set $(R_1^I, \ldots, R_n^I)$ of logical relations interpreting the relation symbols $(R_1, \ldots, R_n)$, respectively, where the set of values appearing in $(R_1^I, \ldots, R_n^I)$, called the *active domain* of I, is contained in $\text{Const} \cup \text{Nulls}$. The active domain of an instance I is denoted $\text{adom}(I)$.

In addition to the above definition, it is often convenient for clarity and/or conciseness to view an instance as a set of facts. A *fact* of an $\mathbf{R}$-instance I is an expression of the form $R(a_1, \ldots, a_k)$, where R is a relation symbol of arity k in $\mathbf{R}$ and $(a_1, \ldots, a_k)$ is a member of the relation R^I on I that interprets the relation symbol R.

We use this second definition widely because it allows us to easily express some critical notions:

- An $\mathbf{R}$-instance I' is a *sub-instance* of an $\mathbf{R}$-instance I if $I' \subseteq I$, where I' and I are viewed as sets of facts; and
- If I is an $\mathbf{R}$-instance and $\mathbf{R}' \subseteq \mathbf{R}$, then the $\mathbf{R}'$-*restriction* of I, denoted $I\big|_{\mathbf{R}'}$, is the subinstance of I containing only those facts that involve relations from $\mathbf{R}'$.

It is often necessary to express more subtle relationships between instances than can be captured using subset relationships. We make use of several kinds of mapping functions that preserve certain instance characteristics. The most important of these is a "homomorphism". A *homomorphism* between two instances K and K' is a map h from the active domain of K to the active domain of K' that is the identity function on all elements of Const and such that for every atom $\text{R}(v_1, ..., v_n) \in K$ we have that $\text{R}(h(v_1), ..., h(v_n)) \in K'$. In effect, this type of homomorphism maps nulls to either constants or nulls in such a way that K becomes a sub-instance of K'.

Typically, data is extracted from a database instance using a query, which in the most general sense is an expression describing which data to extract. In this work, a *query* over a schema $\mathbf{R}$ is a first-order formula over $\mathbf{R}$. An important subclass of queries are the "conjunctive queries", which correspond to the select-project-join fragment of Codd's relational algebra, and constitute the core syntax of the most widely used database query language, SQL. A *conjunctive query* (CQ) over the schema $\mathbf{R}$ is a first-order formula of the form $q(\mathbf{x})$:- $\exists \mathbf{y} \phi(\mathbf{x}, \mathbf{y})$, where $\phi(\mathbf{x}, \mathbf{y})$ is a conjunction of relational atoms contructed using relations from $\mathbf{R}$ and variables from $\mathbf{x} \cup \mathbf{y}$. We assume an infinite set Vars of variables containing both $\mathbf{x}$ and $\mathbf{y}$. A *boolean query* is one where $\mathbf{x} = \emptyset$, and we say a query is *projection-free* if $\mathbf{y} = \emptyset$. A *union of conjunctive queries* (UCQ) is a first-order formula of form $q(\mathbf{x})$:- $\exists \mathbf{y} \phi(\mathbf{x}, \mathbf{y})$, where $\phi(\mathbf{x}, \mathbf{y})$ is the disjunction of one or more conjunctions of relational atoms.

We consider a boolean query q :- ϕ to be "satisfied" by an instance I, denoted $I \models q$, if there exists an assignment s such that $I, s \models \Phi$. For a query with k free variables, the *answers* to q on I, denoted $q(I)$, are the distinct tuples of the form $(a_1, \ldots, a_k)$ such that $I \models q(a_1, \ldots, a_k)$. Additionally, we denote by $q(I)\downarrow$ the answers of q on I that contain only values from Const.

Data Exchange

In data exchange, data is transformed from a source schema to a target schema. Desired properties of databases are typically captured by rules called "constraints" (or equivalently, "dependencies"). The relationship between source and target is specified by constraints involving the source and target schemas, and target instances may be further specified by target constraints.

Common data exchange operations such as copying a relation from source to

target, taking a projection, creating new attributes with unknown values, and joining or splitting relations, are all encompassed by the source-to-target version of a well-studied class of constraints called "tuple generating dependencies", or "tgds". Tuple generating dependencies are a generalization of inclusion dependencies and foreign-key dependencies, two common practical classes of database constraints.

Definition 2.1 (tuple generating dependency [2])**.** A *tuple-generating dependency* (TGD) is an expression of the form $\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\psi(\mathbf{x}, \mathbf{y}))$, where $\phi(\mathbf{x})$ and $\psi(\mathbf{x}, \mathbf{y})$ are conjunctions of atoms over some relational schema.

Tuple generating dependencies are also known as GLAV (Global-and-Local-As-View) constraints. Two important special cases are the GAV constraints, those with a single relational atom as the consequent, and the LAV constraints, those with a single relational atom as the implicant. The former are the tgds of the form $\forall \mathbf{x}, \mathbf{y}(\phi(\mathbf{x}, \mathbf{y}) \rightarrow \mathrm{P}(\mathbf{x}))$, and the latter are the tgds of the form $\forall \mathbf{x}(\mathrm{R}(\mathbf{x}) \rightarrow \exists \mathbf{y}\psi(\mathbf{x}, \mathbf{y}))$, where P and R are individual relation symbols. An essential characteristic of GAV tgds is that they do not contain existentially quantified variables. The class of all tgds with no existentially quantified variables are called *full*, which subsumes the class of GAV tgds. Moreover, every full tgd $\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \mathrm{P}_1(\mathbf{x}_1) \wedge \cdots \wedge \mathrm{P}_\mathrm{n}(\mathbf{x}_n))$, where each $\mathbf{x}_i \subseteq \mathbf{x}$, is logically equivalent to the set of GAV tgds:

$$\left\{ \begin{array}{c} \forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \mathrm{P}_1(\mathbf{x}_1)) \\ \cdots \\ \forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \mathrm{P}_\mathrm{n}(\mathbf{x}_n)) \end{array} \right\}$$

Suppose, as in data exchange, that we have two disjoint relational schemas $\mathbf{S}$ and $\mathbf{T}$, called the *source* schema and the *target* schema. A *source-to-target tgd* (s-t tgd) is a tgd as above such that $\phi(\mathbf{x})$ is a conjunction over $\mathbf{S}$ and $\psi(\mathbf{x}, \mathbf{y})$

is a conjunction over $\mathbf{T}$. In contrast, we sometimes refer to tgds over the target schema as *target tgds*.

In addition to s-t tgds and target tgds, in our work we consider the impact of another major class of target constraint called "equality generating dependencies" (egds). Equality generating dependencies capture key constraints[1] and functional dependencies, the former of which is likely the most widely used type of database constraint in industry.

Definition 2.2 (equality generating dependency [2])**.** An *equality-generating dependency* (EGD) is an expression of the form $\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow x_i = x_j)$ with $\phi(\mathbf{x})$ a conjunction of atoms over a relational schema.

Since tgds and egds both begin with a universal quantifier, for the sake of readability, we will frequently drop universal quantifiers when writing tgds and egds.

Data exchange is formalized by a *schema mapping*, a quadruple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$, where $\mathbf{S}$ is a source schema, $\mathbf{T}$ is a target schema, Σ_{st} is a finite set of source-to-target constraints, and Σ_{t} is a finite set of constraints over the target schema [32].

In this work we study several classes of schema mappings specified using different constraint languages. We will use the notation:

- GLAV - the class of global-and-local-as-view constraints (i.e., tgds)
- GAV - the class of global-as-view constraints
- LAV - the class of local-as-view constraints
- EGD - the class of egds.

[1] often called "primary key" and "unique key" constraints.

If C is a class of sets of source-to-target dependencies and D is a class of sets of target dependencies, then the notation C+D denotes the class of all schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ such that Σ_{st} is a member of C and Σ_{t} is a member of D. For example, GLAV+EGD denotes the class of all schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ such that Σ_{st} is a finite set of S-T TGDs and Σ_{t} is a finite set of EGDs. Moreover, we will use the notation (D_1, D_2) to denote the union of two classes D_1 and D_2 of sets of target dependencies. For example, GAV+(GAV, EGD) denotes the class of all schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ such that Σ_{st} is a set of GAV S-T TGDs and Σ_{t} is the union of a finite set of GAV target tgds with a finite set of target egds.

In [32], the authors define a *solution* for a data exchange setting as a target instance that satisfies the target constraints, and, together with the source, satisfies the source-to-target constraints as well. Even more importantly, they establish criteria for a "good" solution.

Definition 2.3. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a schema mapping. A target instance J is a *solution* for a source instance I w.r.t. $\mathcal{M}$ if J is finite, and the pair (I, J) satisfies $\mathcal{M}$, i.e., I and J together satisfy Σ_{st}, and J satisfies Σ_{t}. A *universal* solution for I is a solution J for I such that for any solution J' for I, there is a homomorphism h from J to J'.

A universal solution constitutes a "most-general" solution, in the sense that it has a homomorphism into every other solution, and thus it captures no more information than any other solution. We will see below that a universal solution also contains no less information than necessary for answering target UCQs.

Note that, by definition, instances are finite. Additionally, by convention, we will assume that source instances do not contain null values. If $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ is an arbitrary schema mapping, then a given source instance may have no solution

or it may have a solution, but no (finite) universal solution. However, if Σ_t is the union of a *weakly acyclic* set of target TGDs and a set of EGDs, then a solution exists if and only if a universal solution exists [32]. Moreover, the *chase procedure* can be used to determine if, given a source instance I, a solution for I exists and, if it does, to actually construct a universal solution $chase(I, \mathcal{M})$ for I in time polynomial in the size of I. The solution $chase(I, \mathcal{M})$ is called a *canonical universal solution* of I (or *the* canonical universal solution of I, if the TGDs in $\mathcal{M}$ are full). The definition of weak acyclicity is given next, followed by the definition of the *chase procedure*.

Definition 2.4 (weakly acyclic [32])**.** Let Σ be a set of TGDs over a schema $\mathbf{T}$. Construct a directed graph, called the *dependency graph*, as follows:

- Nodes: For every pair $(R,\ A)$ with R a relation symbol in $\mathbf{T}$ and A an attribute of R, there is a distinct node; call such a pair $(R,\ A)$ a *position*.
- Edges: For every TGD $\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{x}, \mathbf{y}))$ in Σ and for every x in $\mathbf{x}$ that occurs in ψ, and for every occurrence of x in ϕ in position $(R,\ A_i)$:
  1. For every occurrence of x in ψ in position $(S,\ B_j)$, add an edge $(R, A_i) \rightarrow (S, B_j)$ (if it does not already exist).
  2. For every existentially quantified variable y and for every occurrence of y in ψ in position $(T,\ C_k)$, add a *special edge* $(R, A_i) \rightarrow (T, C_k)$ (if it does not already exist).

We say that Σ is *weakly acyclic* if the dependency graph has no cycle going through a special edge. We say that a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ is weakly acyclic if the set of tgds in Σ_t is weakly acyclic.

The TGD $\forall x \forall y(E(x, y) \rightarrow \exists z\, E(x, z))$ is weakly acyclic; in contrast, the TGD $\forall x \forall y(E(x, y) \rightarrow \exists z\, E(y, z))$ is not, because the dependency graph contains a

special self-loop. Moreover, every set of GAV TGDs is weakly acyclic, since the dependency graph contains no special edges in this case.

We will also make use of the notion of *rank* [32]. Let Σ be a finite weakly acyclic set of TGDs. For every node (R, A) in the dependency graph of Σ, define an *incoming path* to be any (finite or infinite) path ending in (R, A). Define the *rank* of (R, A), denoted by $rank(R, A)$, as the maximum number of special edges on any such incoming path. Since Σ is weakly acyclic, there are no cycles going through special edges; hence, $rank(R, A)$ is finite. The *rank* of Σ, denoted $rank(\Sigma)$ is the maximum of $rank(R, A)$ over all positions (R, A) in the dependency graph of Σ.

What follows is the definition of the *chase procedure*.

Definition 2.5 (chase procedure [32])**.** Let K be an instance.

(tgd) Let d be a TGD $\phi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$. Let h be a homomorphism from $\phi(\mathbf{x})$ to K (where $\phi(\mathbf{x})$ is treated as a collection of relational facts, i.e., an instance) such that there is no extension of h to a homomorphism h' from $\phi(\mathbf{x}) \wedge \psi(\mathbf{x}, \mathbf{y})$ to K. We say that *d can be applied to K with homomorphism h.*

Let K' be the union of K with the set of facts obtained by: (a) extending h to h' such that each variable in $\mathbf{y}$ is assigned a fresh labeled null, followed by (b) taking the image of the atoms of ψ under h'. We say that *the result of applying d to K with h* is K', and write $K \xrightarrow{d,h} K'$.

(egd) Let d be an EGD $\phi(\mathbf{x}) \rightarrow (x_1 = x_2)$. Let h be a homomorphism from $\phi(\mathbf{x})$ to K such that $h(x_1) \neq h(x_2)$. We say that *d can be applied to K with homomorphism h.* We distinguish two cases.

- If both $h(x_1)$ and $h(x_2)$ are in Const then we say that *the result of*

applying d to K with h is "failure", and write $K \xrightarrow{d,h} \perp$.

- Otherwise, let K' be K where we identify $h(x_1)$ and $h(x_2)$ as follows: if one is a constant, then the labeled null is replaced everywhere by the constant; if both are labeled nulls, then one is replaced everywhere by the other. We say that *the result of applying d to K with h is K'*, and write $K \xrightarrow{d,h} K'$.

In the above, $K \xrightarrow{d,h} K'$ (including the case where K' is $\perp$) is called a *chase step*. We now define chase sequences and finite chases.

Let Σ be a set of TGDs and EGDs, and let K be an instance.

- A *chase sequence of K with Σ* is a sequence (finite or infinite) of chase steps $K_i \xrightarrow{d_i,h_i} K_{i+1}$, with $i = 0, 1, ...$, with $K = K_0$ and d_i a dependency in Σ.

- A *finite chase of K with Σ* is a finite chase sequence $K_i \xrightarrow{d,h} K_{i+1}$, $0 \leq i < m$, with the requirement that either (a) $K_m = \perp$ or (b) there is no dependency d_i of Σ and there is no homomorphism h_i such that d_i can be applied to K_m with h_i. We say that K_m is the result of the finite chase. We refer to case (a) as the case of a *failing finite chase* and we refer to case (b) as the case of a *successful finite chase*.

In the context of data exchange, we chase the source instance first with the source-to-target constraints, and then continue chasing with the target constraints. The nature of S-T TGDs ensure that no atoms are created over the source schema, so in this setting the result of chasing a source instance I with a schema mapping $\mathcal{M}$ is a pair (I, J) where J is a target instance. We usually refer to J alone as the result of the chase. Note that the result of the chase is unique up to renaming of nulls.

In our work, we focus primarily on weakly acyclic schema mappings, and we frequently leverage the following result: If Σ is the union of a set of egds with a weakly acyclic set of tgds, then there exists a polynomial in the size of an instance K that bounds the length of every chase sequence of K with Σ [32]. In data exchange terms, if the target tgds are weakly acyclic, then the chase terminates (successfully or unsuccessfully) in polynomial time.

We now give the traditional semantics for answering target queries in data exchange.

Definition 2.6 (certain answers [32])**.** If q is a query over the target schema $\mathbf{T}$ and I is a source instance, then the *certain answers* of q with respect to $\mathcal{M}$ are defined as

$$\text{certain}(q, I, \mathcal{M}) = \bigcap\{q(J) : \ J \text{ is a solution for } I \text{ w.r.t. } \mathcal{M}\}$$

Recall that $q(J)\downarrow$ is defined as the answers of q on J that contain only values from Const. Fagin et. al. give us the following result:

Proposition 2.7 ([32])**.** *An instance J is a universal solution for a source instance I w.r.t. a schema mapping $\mathcal{M}$ if and only if, for every conjunctive query q, it holds that* $\text{certain}(q, I, \mathcal{M}) = q(J)\downarrow$.

This gives us a PTIME algorithm for conjunctive query answering in data exchange with respect to a weakly acyclic schema mapping. In Chapter 1, we showed that this gives non-trivial, meaningful answers only when solutions exist for the given source instance.

Database Repairs and Consistent Query Answering

Let Σ be a set of constraints over some relational schema. An *inconsistent* database is a database that violates at least one constraint in Σ. Informally, a *repair* of an inconsistent database I is a consistent database I' that differs from I in a "minimal" way. This notion can be formalized in several different ways [6]. For example, I' may be a copy of I with a minimal number of attribute values changed (an "attribute-based repair"). Alternatively, I' might be a copy of I with a minimal number of tuples added and removed (a "cardinality repair"). We take inspiration mainly from a third type of repair, called a "set-based repair", in which I' is a copy of I with a subset-minimal set of tuples added or removed. Set-based repairs are formulated under the assumption that tuples are indivisible. We will see shortly that this idealogy does not necessitate discarding any information in the case when a constraint is violated. We begin with the definition of three interesting, related variations of set-based repairs.

Definition 2.8. 1. A *symmetric-difference-repair* of I, denoted $\oplus$-repair of I, is an instance I' that satisfies Σ and where there is no instance I'' such that $I \oplus I'' \subset I \oplus I'$ and I'' satisfies Σ. Here, $I \oplus I'$ denotes the set of facts that form the symmetric difference of the instances I and I'.

2. A *subset-repair* of I is an instance I' that satisfies Σ and where there is no instance I'' such that $I' \subset I'' \subseteq I$ and I'' satisfies Σ.

3. A *superset-repair* of I is an instance I' that satisfies Σ and where there is no instance I'' such that $I' \supset I'' \supseteq I$ and I'' satisfies Σ.

Clearly, subset-repairs and superset-repairs are also $\oplus$-repairs; however, a $\oplus$-repair need not be a subset-repair nor a superset-repair. Nonetheless, for certain

classes of constraints, these different types of set-based repairs are known to collapse. For example, given an instance I, if Σ is a set of EGDs, then every $\oplus$-repair of I w.r.t. Σ is a subset-repair [21].

For a particular type of repair and for a set Σ of constraints, a natural problem to consider is the following: given two instances I and I', is I' a repair of I w.r.t. Σ? This is called the REPAIR CHECKING PROBLEM , and it is a useful measure of the inherent complexity of a type of repair. In [3], Afrati and Kolaitis establish that the REPAIR CHECKING PROBLEM for weakly acyclic sets of tgds and egds is coNP-complete in data complexity for both subset- and $\oplus$-repairs (and likewise for inclusion dependencies and egds).

In general, an inconsistent instance I does not have one unique $\oplus$-repair, but rather a set of possible repairs. There may be exponentially many. (The same holds for subset- and superset-repairs). Each repair is just one possible way to modify the database to make it satisfy the constraints. An interactive user might study these options, and pick one. In the non-interactive case, rather than choosing between the possibilities, we compute answers to queries that will hold for all such choices. These are are called the "consistent answers".

Definition 2.9. Given an instance I and a set Σ of constraints, the *symmetric-difference consistent query answers* of I w.r.t. Σ are defined as:

$$\oplus\text{-CQA}(q, I, \Sigma) = \bigcap\{q(I') : \ I' \text{ is a } \oplus\text{-repair of } I \text{ w.r.t. } \Sigma\}$$

Subset- and superset-CQA are defined analogously.

Consistent answers come at a computational cost. As in data exchange, if we consider arbitrary sets of TGDs, query answering may be undecidable for $\oplus$-repairs. However, in contrast with data exchange, even if we restrict Σ to a weakly

acyclic set of TGDs (with or without EGDs), both $\oplus$-CQA and subset-CQA are Σ_2^{P}-complete in data complexity [72]. Nonetheless, there are conditions under which CQA becomes tractable. For a given schema and set of key constraints, and for a boolean conjunctive query q, if there exists a boolean first-order query q' whose answers for every database instance I are the consistent answers to q on I, then we say that q is *first-order rewritable*. In [48], Koutris and Wijsen prove a trichotomy in the complexity of consistent query answering for self-join-free conjunctive queries. Specifically, for a given schema, set of key constraints Σ, and boolean conjunctive query q, there is a polynomial-time algorithm to determine whether q is first-order rewritable w.r.t. Σ, or else whether the data complexity of consistent query answering for q w.r.t. Σ is in PTIME (but LOGSPACE-hard), or is coNP-complete.

Chapter 3

XR-certain Semantics

The traditional semantics of data exchange has, at its core, the assumption that source instances are reliable sources of information. We have seen that the traditional approach is not equipped to handle the situation when a source instance has no solution. To develop a semantics that gives meaningful answers in such circumstances, we will have to relax the assumption that source facts hold true, which we will accomplish by considering source instances that differ minimally from the orginal, but for which solutions do exist. We will also preserve the standard assumption in data exchange that we wish to construct a target instance – that is, we begin with an empty target instance.

Exchange-Repairs

We now introduce our approach, called *exchange repair*.

Definition 3.1 (**source repair, XR-solution, XR-certain answers).** [19] Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a schema mapping, let I be an $\mathbf{S}$-instance.

1. A source instance I' is said to be a *source repair* of I with respect to $\mathcal{M}$ if $I' \subseteq I$ and I' is set-containment-maximal amongst subsets of I that have

a solution w.r.t. $\mathcal{M}$. That is, I' has a solution with respect to $\mathcal{M}$, and no instance I'' with $I' \subsetneq I'' \subseteq I$ has a solution with respect to $\mathcal{M}$.

2. We say that a pair (I', J') is an *exchange-repair solution* (or XR-solution) for I with respect to $\mathcal{M}$ if I' is a source repair of I with respect to $\mathcal{M}$ and J' is a solution for I' with respect to $\mathcal{M}$.

3. For a query q over the target schema $\mathbf{T}$, the *XR-certain answers* to q in I w.r.t. $\mathcal{M}$, denoted XR-certain$(q, I, \mathcal{M})$, is the set

$$\bigcap \{q(J') \mid (I', J') \text{ is an XR-solution for } I \text{ w.r.t. } \mathcal{M}\}$$

We will further qualify the term XR-solution with any of the modifiers describing special classes of solutions (e.g., universal, canonical universal. For example, if I' is a source repair for I w.r.t. $\mathcal{M}$ and J' is a canonical universal solution for I' with respect to $\mathcal{M}$, then we say (I', J') is a canonical universal XR-solution (or just "canonical XR-solution").

$(I, J)'$

Task_Assignments		
person	task	dept
~~peter~~	~~tpsreport~~	~~software~~
~~peter~~	~~spaceout~~	~~software~~
peter	meetbobs	exec

Departments	
person	dept
peter	exec

Tasks	
person	task
peter	meetbobs

$(I, J)''$

Task_Assignments		
person	task	dept
peter	tpsreport	software
peter	spaceout	software
~~peter~~	~~meetbobs~~	~~exec~~

Departments	
person	dept
peter	software

Tasks	
person	task
peter	tpsreport
peter	spaceout

Figure 3.1: Two XR-solutions for I w.r.t. $\mathcal{M}$. The `Stakeholders` tables are omitted for brevity.

Figure 3.1 shows (the only two) XR-solutions for the source instance and schema mapping of Example 1.1. Recall that a materialize-then-repair approach gave unsatisfactory answers (e.g., (peter, lumbergh), even though we can-

not conclude that peter works in software), while an exchange-as-repair approach gave only $\emptyset$. The XR-certain semantics improve on both: if we now evaluate $\texttt{boss}(\text{peter}, b)$ over each target instance, and take the intersection, we obtain that XR-certain$(\texttt{boss}(\text{peter}, b), \text{src}, \mathcal{M}) = \{(\text{peter}, \text{bobs})\}$. In other words, with certainty, peter works for the bobs, even though his assigned department is uncertain. Notice that the shared origins of tuples are reflected in the XR-solutions (e.g., in each, peter performs tasks only for his assigned department, in contrast to Figure 1.2), but the XR-solutions retain more derived target information than the instances in Figure 1.1.

Although the details of the formulation of XR-certain semantics are specific to data exchange, it nonetheless falls into a more general framework of *possible worlds* (the XR-solutions) and *certain answers* (the answers that hold in every possible world). The same can be said of inconsistent databases (in which the possible worlds are the repairs), and also of the traditional semantics of data exchange (in which the possible worlds are those given by different valuations of the nulls).

We will see in the remainder of the chapter that XR-certain semantics is a generalization of both classical data exchange and subset-CQA. Additionally, we will justify the requirement that a source repair be a subset of the original source. Furthermore, we will demonstrate that, in general, XR-certain semantics differs non-trivially from $\oplus$-CQA and also from approaches proposed elsewhere in the literature for inconsistency tolerant data exchange. In several related fields of research (e.g., data integration, ontology-based data access), semantics have been studied that have some connections to exchange repair. These are described in detail in Section 3.1. Finally, in the last sections of the chapter, we will explore fundamental computational properties of XR-certain semantics.

Justification of XR-certain Semantics

Source repairs are repairs of the source instance: they are given by minimal tuple deletions of the source such that it becomes compatible with the schema mapping. This gives us query answering semantics in which the answers given are those that hold under the traditional semantics for source instances very closely related to the original. The next proposition tells us that the XR-certain answers coincide with the certain answers when solutions exist.

Proposition 3.2. *Let* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ *be a schema mapping and let* I *be an* $\mathbf{S}$*-instance. If* I *has a solution w.r.t.* $\mathcal{M}$*, then for every target query* $\text{XR-certain}(q, I, \mathcal{M}) = \text{certain}(q, I, \mathcal{M})$.

Proof. The instance I has a solution w.r.t. $\mathcal{M}$, and therefore it is its own only source repair. Thus every XR-solution for I w.r.t. $\mathcal{M}$ is I along with a solution J for I w.r.t. $\mathcal{M}$. Therefore $\bigcap\{q(J) \mid J \text{ is a solution for } I \text{ w.r.t. } \mathcal{M}\} = \bigcap\{q(J) \mid (I, J) \text{ is an XR-solution for } I \text{ w.r.t. } \mathcal{M}\}$. □

The XR-certain semantics certainly enjoy some appealing semantic properties (and computational properties, as we will see in Section 3.2). However, it's reasonable to wonder whether a subset-based maximality condition might be too restrictive for the setting of data exchange. We will now see why this is not the case. Consider the following alternative semantics, which is a seemingly obvious choice since, for some source instances, both tuple deletions and tuple insertions are necessary in order to find a source and target pair that satisfy certain schema mappings. These semantics are inspired by symmetric-difference repairs of inconsistent databases.

Definition 3.3 (**$\oplus$-exchange-repair solution, $\oplus$-source repair).** Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a schema mapping, let I be an $\mathbf{S}$-instance.

1. A source instance I' is said to be a *symmetric-difference source repair* (or $\oplus$-*source repair*) of I with respect to $\mathcal{M}$ if I' has a solution w.r.t. $\mathcal{M}$, and there is no instance I'' that has a solution w.r.t. $\mathcal{M}$ such that $I \oplus I'' \subset I \oplus I'$.

2. We say that a pair (I', J') is a *symmetric-difference exchange-repair solution* (or $\oplus$-XR-solution) for I with respect to $\mathcal{M}$ if I' is a symmetric-difference source repair of I with respect to $\mathcal{M}$ and J' is a solution for I' with respect to $\mathcal{M}$.

In general, source repairs have different properties from those of standard database repairs. Indeed, as mentioned earlier, a $\oplus$-repair of an inconsistent database need not be a subset repair. In contrast, Theorem 3.5 (below) asserts that the state of affairs is different for $\oplus$-XR-solutions and XR-solutions. We need the following lemma in order to prove Theorem 3.5.

Lemma 3.4. *Let $\mathcal{M}$ be a* GLAV+(GLAV, EGD) *schema mapping. If $I' \supseteq I$ are two source instances, then every solution for I' w.r.t. $\mathcal{M}$ is also a solution for I w.r.t. $\mathcal{M}$; consequently, if I has no solution w.r.t. $\mathcal{M}$ then I' has no solution w.r.t. $\mathcal{M}$.*

Proof. Let $I' \supseteq I$ be two source instances. We will show that if I' has a solution w.r.t. $\mathcal{M}$, then I also has a solution w.r.t. $\mathcal{M}$. Let J be an arbitrary solution for I' w.r.t. $\mathcal{M}$. Let $\phi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$ be an arbitrary TGD in Σ_{st}, and let $h : \mathbf{x} \rightarrow \text{adom}(I)$ be a homomorphism such that $h(\phi(\mathbf{x})) \subseteq I$, and of course $h(\phi(\mathbf{x})) \subseteq I'$ as well. Then h can be extended to some homomorphism h' such that $h'(\psi(\mathbf{x}, \mathbf{y})) \subseteq J$, and therefore (I, J) together satisfy Σ_{st}, and since J satisfies Σ_{t}, we have that J is also a solution for I w.r.t. $\mathcal{M}$. □

Theorem 3.5. *Let $\mathcal{M}$ be a* GLAV+(GLAV, EGD) *schema mapping. Let I be a source instance. Then if (I', J') is a* $\oplus$-XR-solution *of I w.r.t. $\mathcal{M}$, then (I', J') is*

actually an XR-solution *of I w.r.t. $\mathcal{M}$. Consequently, every $\oplus$-source-repair of I is also a source-repair of I.*

Proof. Let (I', J') be an $\oplus$-XR-solution for I w.r.t. $\mathcal{M}$. Suppose $I' \setminus I \neq \emptyset$. Then by Lemma 3.4, J' is also a solution for $I' \cap I$. Since $I' \oplus I \supset (I' \cap I) \oplus I$, we have that (I', J') fails the minimality criterion and thus is not a $\oplus$-XR-solution for I w.r.t. $\mathcal{M}$, which is a contradiction. □

Notice that if $\mathcal{M}$ is a GLAV+(GLAV, EGD) schema mapping, then source-repairs always exist: the pair $(\emptyset, \emptyset)$ trivially satisfies $\mathcal{M}$, so for every source instance I there must exist some maximal subinstance I' of I for which solutions exist.

3.1 Relationship to Other Frameworks

The work reported here builds directly on the work of many others, in particular the foundational work on database repairs and consistent query answering by Arenas, Bertossi, and Chomicki [6], and on data exchange and certain answers by Fagin *et al.* [32]. Inconsistency-tolerant semantics have been studied in several different areas of database management, including inconsistent databases, data integration, and ontology-based data access (OBDA). The common motivation for inconsistency-tolerant semantics is to give non-trivial, meaningful semantics to query answering. We now discuss the relationship between the XR-certain answers and the inconsistency-tolerant semantics of queries in these different contexts.

It is often convenient to consider the special case of the *copy mapping*.

Definition 3.6 (copy mapping). [73] Given a source schema $\mathbf{S}$, let $\mathbf{T}$ be the schema that contains, for every relation R $\in \mathbf{S}$, a corresponding relation R′ of the same arity. The *copy mapping* is a set of source-to-target tgds constructed as

follows: for each relation $R \in \mathbf{S}$, add the tgd $R(\mathbf{x}) \rightarrow R'(\mathbf{x})$. We say an instance J is the *copy* of an instance I if J is the canonical universal solution for I w.r.t. the copy mapping (so it contains the same facts appropriately renamed to match the target schema).

The copy mapping is a source-to-target mapping. When we use the copy mapping, we will typically append target tgds to it to fit the problem at hand.

3.1.1 Connections with Database Repairs

We have already seen that, in general, the problems of data exchange and database repair have different characteristics. We will now explore in more detail how these two settings compare, including some special cases in which they coincide. The following statements are true for arbitrary source instances and schema mappings.

1. Repairs of the target instance obtained by chasing with the tgds of the schema mapping are not necessarily XR-solutions.

2. Repairs of $(I, \emptyset)$ are not necessarily XR-solutions.

For the first statement, consider the pair (I, J') from Figures 1.1 and 1.2, where J' is a $\oplus$-repair of the inconsistent result J of the chase of I. Clearly, (I, J') is not an XR-solution, because J' is not a solution for I. For the second statement, consider the pairs (I_1, J_1), (I_2, J_2), (I_3, J_3) in Figure 1.1, all of which are $\oplus$-repairs of $(I, \emptyset)$. The first two are also XR-solutions of I, but the third one is not.

It can also be shown that XR-solutions are not necessarily $\oplus$-repairs of $(I, \emptyset)$. We now describe an important case in which XR-solutions **are** $\oplus$-repairs of $(I, \emptyset)$. For this, we recall the notion of a *core universal solution* from [32]. By definition, a core universal solution is a universal solution that has no homomorphism to a

proper subinstance. Thus, given an instance I and a schema mapping $\mathcal{M}$, a core universal XR-solution for I w.r.t. $\mathcal{M}$ is a pair (I', J') of instances such that I' is a source repair of I w.r.t. $\mathcal{M}$, and J' is a solution for I' w.r.t. $\mathcal{M}$ such that J' has no homomorphism to any proper subinstance of itself. According to [32], if a universal solution exists, then a core universal solution also exists. Moreover, core universal solutions are unique up to isomorphism.

Proposition 3.7. *Let $\mathcal{M}$ be a* GLAV+(GLAV, EGD) *schema mapping. Given a source instance I, if (I', J') is a core universal* XR-solution *for I w.r.t. $\mathcal{M}$, then (I', J') is a $\oplus$-repair of $(I, \emptyset)$ w.r.t. $\Sigma_{st} \cup \Sigma_t$.*

Proof. Let (I'', J'') be a pair of instances which together satisfy $\Sigma_{\text{st}} \cup \Sigma_{\text{t}}$ and such that $(I'', J'') \oplus (I, \emptyset) \subseteq (I', J') \oplus (I, \emptyset)$. Then $J'' \subseteq J'$. Since (I', J') is an XR-solution for I w.r.t. $\mathcal{M}$, there is no instance I'' such that $I' \subset I'' \subseteq I$ and I'' has a solution w.r.t. $\mathcal{M}$. Therefore it must be that $I'' = I'$. Furthermore, since J' is a core universal solution for I' w.r.t. $\mathcal{M}$, there is no proper subinstance $J'' \subset J'$ that is a solution for I' w.r.t. $\mathcal{M}$, so we have that $J'' \supseteq J'$, and thus $J'' = J'$. Therefore (I', J') is a $\oplus$-repair of $(I, \emptyset)$ w.r.t. $\Sigma_{\text{st}} \cup \Sigma_{\text{t}}$. □

This result tells us that, despite the set of XR-solutions and the set $\oplus$-repairs of an instance being incomparable, we can still find a precise relationship between the two associated query answering semantics. The next proposition does just that.

Proposition 3.8. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a weakly acyclic* GLAV+(GLAV, EGD) *schema mapping and q a conjunctive query over the target schema $\mathbf{T}$. If I is a source instance, then* XR-certain$(q, I, \mathcal{M}) \supseteq \oplus$-CQA$(q, (I, \emptyset), \Sigma_{st} \cup \Sigma_t)$. *Moreover, this containment may be a proper one.*

Proof. Since $\mathcal{M}$ is weakly acyclic, for any instance I for which solutions exist, a core universal solution also exists. Therefore, we have that XR-certain$(q, I, \mathcal{M}) = \bigcap\{q(J') : (I', J')$ is an XR-solution for I w.r.t. $\mathcal{M}$, and J' is a core universal solution for I' w.r.t. $\mathcal{M}$ $\}$. By Proposition 3.7, the set of XR-solutions (I', J') where J' is a core universal solution for I' w.r.t. $\mathcal{M}$ is a subset (maybe proper) of the set of $\oplus$-repairs of $(I, \emptyset)$ w.r.t. $\Sigma_{\text{st}} \cup \Sigma_{\text{t}}$. Therefore XR-certain$(q, I, \mathcal{M}) \supseteq$ $\oplus$-CQA$(q, (I, \emptyset), \Sigma_{\text{st}} \cup \Sigma_{\text{t}})$.

To see that this containment may be a proper one, consider the schema mapping $\mathcal{M}$ and query $\texttt{boss}(\texttt{peter}, b)$ in Example 1.1, and the repairs of $(I, \emptyset)$ in Figure 1.1. It is easy to verify that $\oplus$-CQA$(\texttt{boss}(\texttt{peter}, b), (I, \emptyset), \Sigma_{\text{st}} \cup \Sigma_{\text{t}}) = \emptyset$, while XR-certain$(\texttt{boss}(\texttt{peter}, b), I, \mathcal{M}) = \{(\texttt{peter}, \texttt{bobs})\}$. □

We will now consider a special case in which the relationship between XR-certain answers and consistent answers is even stronger. The following proposition pertains to the case where Σ_{st} is the copy mapping.

Proposition 3.9. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a* GAV+EGD *schema mapping where Σ_{st} is the copy mapping, and let q be a conjunctive query over the target schema $\mathbf{T}$. Then for every instance I, it holds that* XR-certain$(q, I, \mathcal{M}) =$ *subset*-CQA(q, J, Σ_t)*, where J is the copy of I.*

Proof. Since Σ_{st} specifies the copy mapping and Σ_{t} contains only egds, for every source repair I' there is an XR-solution (I', J') where J' is the copy of I'. Furthermore, J' is a universal solution for I' w.r.t. $\mathcal{M}$, so we can write XR-certain$(q, I, \mathcal{M}) = \bigcap\{q(J') \mid (I', J')$ is an XR-solution for I w.r.t. $\mathcal{M}$ and J' is the copy of $I'\}$. Therefore XR-certain$(q, I, \mathcal{M}) = \bigcap\{q(J') \mid J'$ is the copy of a maximal subset of I such that $J' \models \Sigma_{\text{t}}\}$. Let J be the copy of I. Then XR-certain$(q, I, \mathcal{M}) = \bigcap\{q(J') \mid J'$ is a maximal subset of J such that $J' \models \Sigma_{\text{t}}\}$, which is precisely subset-CQA$(q, J, \Sigma_{\text{t}})$. □

3.1.2 Connections with Data Integration

Data integration is a problem closely related to data exchange, and in fact they are formalized very similarly. In data integration, data from multiple source instances is combined into a global schema, to which queries may be posed. The global schema is analogous to a target schema in data exchange. Data integration differs from data exchange primarily in its motivation and in the settings that are typically considered. For example, the global schema in a data integration system is presumed to be virtual (and the sources assumed to be continuously accessible), whereas in data exchange it is presumed that a target instance will be materialized and used in isolation for query answering. While data integration research has grappled with different notions of soundness and completeness (in which either source or target is treated like a view of the other), in data exchange it is usually assumed that sources are reliable (i.e., that all their facts are true), though we relax that assumption in this work. The on-the-fly nature of data integration has motivated the use of fairly restrictive source-to-target constraint languages (e.g., LAV, GAV) in most data integration research, as opposed to the once-and-done nature of data exchange, which lends itself to the study of richer constraint languages. Target constraints have been studied in both scenarios. A data integration system $\mathcal{I}$ is formalized as triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{G}$ is a global schema, $\mathcal{S}$ is a source schema, and $\mathcal{M}$ is a mapping from $\mathcal{S}$ to $\mathcal{G}$. Target constraints, if any, are implicit in $\mathcal{G}$ (that is, the language used to specify $\mathcal{G}$ may include constraints). In [17] and [51], the authors introduce and study the notion of *loosely-sound* semantics for queries in a data integration setting. There are two main differences between their approach and ours. First, they consider scenarios in which the source-to-target mapping is restricted to GAV, and the target is subject only to key constraints and inclusion dependencies. More im-

portantly perhaps, the loosely-sound semantics are, in general, different from the XR-certain answers semantics. Specifically, given a source instance I, the loosely-sound semantics are obtained by first computing the result J of the chase of I with the source-to-target constraints, and then considering as "repairs" all instances J' that satisfy the target constraints and are inclusion maximal in their intersection with J. If all target constraints are egds (in particular, if all target constraints are key constraints), then it is easy to show that, for target conjunctive queries, the loosely-sound semantics coincide with the consistent answers of queries with respect to subset repairs of J. Thus, in this case, the loosely-sound semantics give the same unsatisfactory answers as the materialize-then-repair approach seen in Figure 1.2: concretely, the undesirable instance J' in Figure 1.2 is a possible "repair" using loosely-sound semantics.

3.1.3 Connections with Ontology-Based Data Access

Ontology-based data access (OBDA), originally introduced in [18], is a framework for answering queries over knowledge bases. In that framework, a knowledge base over a schema $\mathbf{T}$ is a pair $\langle \Sigma, D \rangle$, where D is a $\mathbf{T}$-instance and Σ is a set of constraints expressed in some logical formalism over $\mathbf{T}$. The instance D represents extensional knowledge given by the facts of D, and is called the ABox. The set Σ of constraints represents intensional knowledge, and is called the TBox. In most scenarios, the schema $\mathbf{T}$ consists of unary relation symbols, called *concepts*, and of binary relation symbols, called *roles*. Moreover, Σ typically consists of sentences in some description logic. An inconsistency-tolerant semantics in the context of OBDA was first investigated in [54]; this semantics is based on the notion of *AR-repairs* (ABox-repairs) and has become known as *AR-semantics*. Subsequent investigations of AR-semantics were carried out in a number of papers, includ-

ing (in chronological order) [52, 69, 11, 13, 12, 58]. These papers have analyzed the computational complexity of consistent query answering in OBDA and have also considered several variants of the AR-semantics in the OBDA framework. We investigate the relevance to data exchange of several of these variants in Chapter 7.

Data exchange and OBDA are different frameworks that aim to formalize different aspects of data inter-operability. In data exchange there are two schemas, the source schema and the target schema, with no restrictions on the type of relation symbols they contain, while in OBDA there is a single schema that typically contains only unary and binary relation symbols. Moreover, we have seen that the constraints typically used in data exchange are quite different from those typically used in OBDA. One notable exception to this is the work reported in [58], where the OBDA framework studied allows for tuple-generating dependencies (it also allows for *negative constraints*, but not for equality-generating dependencies). In spite of these differences, it turns out that there are close connections between data exchange and OBDA. In what follows, we spell out these connections in detail and show that, regarding consistent query answering, each of these two frameworks can simulate the other.

We first introduce some basic concepts and terminology for OBDA; for the most part, we follow [58]. Let $\mathbf{T}$ be a schema and let $\langle \Sigma, D \rangle$ be a knowledge base over $\mathbf{T}$. A *model* of $\langle \Sigma, D \rangle$ is a $\mathbf{T}$-instance J such that $D \subseteq J$ and $J \models \Sigma$. We write $\mathrm{mod}(\langle \Sigma, D \rangle)$ for the set of all models of $\langle \Sigma, D \rangle$.

An *AR-repair* of $\langle \Sigma, D \rangle$ is a $\mathbf{T}$-instance D' with the following properties: (i) $D' \subseteq D$; (ii) $\mathrm{mod}(\langle \Sigma, D' \rangle) \neq \emptyset$; (iii) D' is an inclusion maximal sub-instance of D having the second property, i.e., there is no $\mathbf{T}$-instance D'' such that $D' \subset D'' \subseteq D$ and $\mathrm{mod}(\langle \Sigma, D'' \rangle) \neq \emptyset$. We write $\mathrm{drep}(\langle \Sigma, D \rangle)$ for the set of all AR-repairs of $\langle \Sigma, D \rangle$.

Next, we introduce the notion of consistent query answering in the context of OBDA. Let q be a Boolean query over the schema $\mathbf{T}$. We say that q is *entailed by* $\langle \Sigma, D \rangle$ *under AR-semantics* if for every AR-repair D' in $\mathrm{drep}(\langle \Sigma, D \rangle)$ and every $\mathbf{T}$-instance $J \in \mathrm{mod}(\langle \Sigma, D' \rangle)$, we have that $J \models q$. If q is a non-Boolean query of arity k over the schema $\mathbf{T}$ and $\mathbf{a}$ is a k-tuple of constants, then we say that $q(\mathbf{a})$ is *entailed by* $\langle \Sigma, D \rangle$ *under AR-semantics* if $q(\mathbf{a})$ is entailed when viewed as a Boolean query; this means that for every AR-repair D' in $\mathrm{drep}(\langle \Sigma, D \rangle)$ and every $\mathbf{T}$-instance $J \in \mathrm{mod}(\langle \Sigma, D' \rangle)$, we have that $\mathbf{a}$ belongs to the result $q(J)$ of evaluating q on J. We write AR-certain$(q, \langle \Sigma, D \rangle)$ to denote the set of all tuples $\mathbf{a}$ such that $q(\mathbf{a})$ is entailed by $\langle \Sigma, D \rangle$ under AR-semantics. By unraveling the definitions, we see that

$$\text{AR-certain}(q, \langle \Sigma, D \rangle) = \bigcap \{ q(J) : J \in \mathrm{mod}(\langle \Sigma, D' \rangle) \text{ and } D' \in \mathrm{drep}(\langle \Sigma, D \rangle) \}.$$

We are now ready to establish the precise connections between the exchange-repairs framework and the OBDA framework.

From OBDA to Exchange Repair

Assume that $\langle \Sigma, D \rangle$ is a knowledge base over a schema $\mathbf{T}$. Let $\mathbf{S}'$ be the schema of the relation symbols occurring in D; note that $\mathbf{S}'$ is a (possibly proper) subschema of $\mathbf{T}$. Let $\mathbf{S}$ be the schema such that, for every relation symbol R' in $\mathbf{S}'$, there is a relation symbol R in $\mathbf{S}$ of the same arity. As shorthand, we will write $D_{\mathbf{S}}$ to refer to instance whose copy is D.

The next proposition shows that the OBDA framework can be simulated by the exchange-repair framework.

Proposition 3.10. *Consider the schema mapping* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma)$*, where* Σ_{st} *is the copy mapping from* $\mathbf{S}$ *to* $\mathbf{S}'$*. The following statements are true.*

1. *For every* $\mathbf{T}$*-instance* $D' \subseteq D$ *and every* $\mathbf{T}$*-instance* J*, we have that* $D' \subseteq J$ *and* $J \models \Sigma$ *if and only if* J *is a solution for* $D'_{\mathbf{S}}$ *w.r.t.* $\mathcal{M}$*.*

2. *For every* $\mathbf{S}$*-instance* $I' \subseteq D_{\mathbf{S}}$ *and every* $\mathbf{T}$*-instance* J*, we have that* J *is a solution for* I' *w.r.t.* $\mathcal{M}$ *if and only if* $I'_{\mathbf{S}^*} \subseteq J$ *and* $J \models \Sigma$*.*

3. *There is a 1-1 correspondence between the AR-repairs of* $\langle \Sigma, D \rangle$ *and the subset source repairs of* $D_{\mathbf{S}}$ *w.r.t.* $\mathcal{M}$*. In fact, the former are the copies of the latter.*

4. *For every query* q *over* $\mathbf{T}$*, we have that* $\text{AR-certain}(q, \langle \Sigma, D \rangle) = \text{XR-certain}(q, D_{\mathbf{S}}, \mathcal{M})$*.*

The proof is straightforward, and it is omitted.

From Exchange-Repair to OBDA

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a schema mapping in which Σ_{st} is a set of s-t tgds and Σ_{t} is a set of arbitrary constraints over $\mathbf{T}$. Recall that the schemas $\mathbf{S}$ and $\mathbf{T}$ have no relation symbols in common. The next proposition tells us that the exchange-repair framework can be simulated by the OBDA framework.

Proposition 3.11. *Let* I *be a source instance. Consider the knowledge base* $\langle \Sigma_{st} \cup \Sigma_t, I \rangle$*. The following statements are true.*

1. *For every source instance* I'*, we have that* I' *is a source repair of* I *w.r.t.* $\mathcal{M}$ *if and only if* I' *is an AR-repair of* $\langle \Sigma_{st} \cup \Sigma_t, I \rangle$*.*

2. *For every query* q *over* $\mathbf{T}$*, we have that* $\text{XR-certain}(q, I, \mathcal{M}) = \text{AR-certain}(q, \langle \Sigma_{st} \cup \Sigma_t, I \rangle)$*.*

Proof. Assume first that I' is a source repair of I w.r.t. $\mathcal{M}$. We have to show that I' is an AR-repair of $\langle \Sigma_{\text{st}} \cup \Sigma_{\text{t}}, I \rangle$. Since I' is a source repair of I w.r.t.

$\mathcal{M}$, we have that $I' \subseteq I$. Moreover, there is a solution J for I' w.r.t. $\mathcal{M}$. Let $J' = I' \cup J$. Clearly, we have that (i) $I' \subseteq J'$ and (ii) $J' \models \Sigma_{\rm st} \cup \Sigma_{\rm t}$, hence $\mathrm{mod}(\langle \Sigma_{\rm st} \cup \Sigma_{\rm t}, I' \rangle) \neq \emptyset$. It remains to show that I' is a maximal sub-instance of I with the preceding properties (i) and (ii). Towards a contradiction, suppose that there is a sub-instance I'' of I such that $I' \subset I'' \subseteq I$ and $\mathrm{mod}(\langle \Sigma_{\rm st} \cup \Sigma_{\rm t}, I'' \rangle) \neq \emptyset$. Consider an instance $J'' \in \mathrm{mod}(\langle \Sigma_{\rm st} \cup \Sigma_{\rm t}, I'' \rangle)$. Then $I'' \subseteq J''$ and $J'' \models \Sigma_{\rm st} \cup \Sigma_{\rm t}$. Let $J''\big|_{\mathbf{T}}$ be the restriction of J'' to the target schema $\mathbf{T}$, that is, $J''\big|_{\mathbf{T}}$ is the sub-instance of J'' consisting of the facts of J'' that involve relation symbols in $\mathbf{T}$ only. We claim that $J''\big|_{\mathbf{T}}$ is a solution for I'' w.r.t. $\mathcal{M}$. Indeed, $J''\big|_{\mathbf{T}} \models \Sigma_{\rm t}$, since all formulas in $\Sigma_{\rm t}$ contain atomic formulas from $\mathbf{T}$ only. Moreover, since $I'' \subseteq J''$ and since $J'' \models \Sigma_{\rm st}$, we have $(I'', J''\big|_{\mathbf{T}}) \models \Sigma_{\rm st}$. This is so because, since $\Sigma_{\rm st}$ consists of s-t tgds, the $\mathbf{S}$-facts in $J'' \setminus I''$ play no role in satisfying $\Sigma_{\rm st}$; we note that this may not hold if, say, $\Sigma_{\rm st}$ contained target-to-source tgds. It follows that I' is not a source repair for I w.r.t. $\mathcal{M}$, which is a contradiction.

Next, assume that I' is an AR-repair of $\langle \Sigma_{\rm st} \cup \Sigma_{\rm t}, I \rangle$. We have to show that I' is a source repair of I w.r.t. $\mathcal{M}$. Since I' is an AR-repair of $\langle \Sigma_{\rm st} \cup \Sigma_{\rm t}, I \rangle$, we have that $I' \subseteq I$ and $\mathrm{mod}(\langle \Sigma_{\rm st} \cup \Sigma_{\rm t}, I' \rangle) \neq \emptyset$. Let J' be a member of $\mathrm{mod}(\langle \Sigma_{\rm st} \cup \Sigma_{\rm t}, I' \rangle)$. Hence, $I' \subseteq J'$ and $J' \models \Sigma_{\rm st} \cup \Sigma_{\rm t}$. If $J'\big|_{\mathbf{T}}$ is the restriction of J' to the target schema $\mathbf{T}$, then $(I', J'\big|_{\mathbf{T}}) \models \Sigma_{\rm st}$ (because $\Sigma_{\rm st}$ consists of s-t tgds) and $J'\big|_{\mathbf{T}} \models \Sigma_{\rm t}$. Thus, there is a solution for I' w.r.t. $\mathcal{M}$. Moreover, we claim that I' is a maximal sub-instance of I for which there exists a solution w.r.t. $\mathcal{M}$. Indeed, if I'' is such that $I' \subset I'' \subseteq I$ and a solution J'' for I'' w.r.t. $\mathcal{M}$ exists, then $I'' \cup J'' \models \Sigma_{\rm st} \cup \Sigma_{\rm t}$. It follows that I' is an AR-repair of $\langle \Sigma_{\rm st} \cup \Sigma_{\rm t}, I \rangle$, which is a contradiction.

Finally, if q is a query over $\mathbf{T}$, then, using the first part of the proposition, it is easy to show that XR-certain$(q, I, \mathcal{M})$ = AR-certain$(q, \langle \Sigma_{\rm st} \cup \Sigma_{\rm t}, I \rangle)$. $\square$

3.1.4 Connections with Prior Work on Inconsistency-Tolerance in Data Exchange

Although several data exchange systems exist that, to some degree, handle target constraints, most do not consider carefully the case where a source instance has no solution. One exception is the peer data exchange work of Grahne and Onet [42], in which a collection of peer instances are treated like a single inconsistent database instance over the union of the schemas. As noted in Chapter 1, this exchange-as-repair approach has shortcomings in the context of source to target (that is, not peer) data exchange.

Another notable inconsistency-tolerant data exchange framework is the mapping and cleaning facility of the Llunatic project [38], in which the concerns of source-to-target and target constraints are studied together. In that work, Geerts *et al.* define a *mapping and cleaning scenario* as a tuple $\mathcal{M} = (\mathbf{S}, \mathbf{S_a}, \mathbf{T}, \Sigma_{\text{tgd}}, \Sigma_{\text{egd}}, \prec_p, \text{User})$, where $\mathbf{S} \cup \mathbf{S_a}$ is the source schema, $\mathbf{S_a}$ is an *authoritative* portion of the source schema, $\mathbf{T}$ is the target schema, Σ_{tgd} is a set of *extended tgds*, Σ_{egd} is a set of *cleaning egds*, $\prec_p$ is a preference order over values, and User is an oracle representing user interactions. An *extended tgd* (or *cleaning egd*) is a tgd (egd) in which the premise may contain atoms from both the source and target schemas. Cleaning egds can be thought of as editing rules, while extended tgds are a mechanism to derive additional target tuples with input directly from authoritative source relations. Define a *cell group* $g = \langle v \rightarrow occ(g)$ by $just(g)\rangle$ as a collection $occ(g)$ of target positions, a value v to place in those positions, and possibly a justification $just(g)$, which is a list of authoritative source positions. Then a repair for an input instance (I, J) in a mapping and cleaning scenario is a set *Rep* of cell groups such that:

- *Rep upgrades* the initial instance J, denoted $J \prec_{p,\text{User}} Rep(J)$. In other

words, there is a mapping h from tuples in J to tuples in $Rep(J)$ (and maintaining positions within those tuples) such that for every constant c appearing in positions $\mathbf{C}$ in J, there is a cell group $g = \langle v \rightarrow occ(g)$ by $just(g)\rangle$ in $Rep(J)$ where $h(\mathbf{C}) \subseteq occ(g)$ and $c \preceq_p v$.

- $(I, Rep(J))$ *satisfies after repairs* the constraints of Σ_{tgd} and Σ_{egd} under $\prec_{p,User}$. In other words, for every egd $\phi(\mathbf{x}) \rightarrow x_i = x_j$, when there is a homomorphism h from $\phi(\mathbf{x})$ into $Rep(J)$, either $h(x_i) = h(x_j)$ or else $h(x_i) \prec_{p,User} h(x_j)$ (or vice-versa). Similarly, for every tgd, either the tgd is satisfied in the traditional sense, or else positions exist whose values constitute an upgrade over the canonical repair.

Repairs in this context are constrained by the notion of satisfaction after repairs, and they are ordered by the notion of upgrades. Repairs of interest are further constrained by a minimality condition, which, intuitively, is that they do not contain spurious additions of tuples nor spurious upgrades of values. The authors make this precise with a containment condition or, failing that, the existence of a surjective non-injective mapping from the tuples of one repair to the tuples of another.

It should be clear that the Llunatic approach to mapping and cleaning assumes that constraints have been written to express preferences among values, and that their notion of repair is essentially the lifting of these value preferences to the level of entire instances. The authors proceed to describe a chase procedure for computing minimal repairs, and then introduce the notion of a "cost manager", which is a heuristic function to prune the space of repairs to consider. The cost manager may, for example, refuse to consider new repairs once it has found some constant number N of repairs. Other cost manager strategies are more dynamic.

Llunatic is a drastically different approach to data exchange than XR-certain

semantics. While XR-certain semantics give query answers for which there is no uncertainty, Llunatic makes a best-effort to compute a small set of subjectively "good" repairs for query answering. While XR-certain semantics are based on the assumption that there is no known preference order among values, Llunatic relies on such an order to produce good repairs. Although XR-certain semantics and Llunatic approach the problem of data exchange in significantly different ways, there are circumstances appropriate for each of them. For example, when preference information is available and absolute certainty is not necessary, Llunatic may be a good algorithm choice for data exchange. On the other hand, when certainty is needed, or more predictable behavior is desired, XR-certain may be the right approach.

3.2 Computational Properties

We have seen that XR-certain semantics coincides with the traditional semantics when the source instance has a solution, but that it uses ideas from consistent query answering to also give meaningful answers to target queries otherwise. We will now see that XR-certain semantics compares favorably against $\oplus$-repairs and CQA in terms of computational complexity.

Computational Complexity

We consider two natural decision problems in the exchange-repair framework, and give tight bounds for their computational complexity.

- Source-Repair Checking: Fix a schema mapping $\mathcal{M}$. Given two source instances I and I', is I' a source-repair of I w.r.t. $\mathcal{M}$?

- XR-certain Query Answering: Fix a schema mapping $\mathcal{M}$ and a boolean

target query q. Given a source instance I, does XR-certain$(q, I, \mathcal{M})$ evaluate to TRUE?

Theorem 3.12. *Let $\mathcal{M}$ be a weakly acyclic* GLAV+(GLAV, EGD) *schema mapping.*

1. *The source-repair checking problem is in* PTIME.
2. *Let q be a union of conjunctive queries over the target schema. The* XR-certain *query answering problem for q is in* coNP.

Moreover, there is a schema mapping specified by copy s-t tgds and target egds, and a boolean conjunctive query for which the XR-certain *query answering problem is* coNP*-complete. Thus, the data complexity of the* XR-certain *answers for boolean conjunctive queries is* coNP*-complete.*

Proof. For the first part, the following is a polynomial time algorithm to check if $I' \subseteq I$ is a source repair of I w.r.t. $\mathcal{M}$:

> Use the chase procedure to check that I' has a solution w.r.t. $\mathcal{M}$ [32]. For every tuple $t \in I \setminus I'$, use the chase procedure to check that $I' \cup \{t\}$ does *not* have a solution w.r.t. $\mathcal{M}$.

The first step ensures that I' has a solution, and by Lemma 3.4, the second step is sufficient to ensure that I' is a maximal such subset of I. Since $\mathcal{M}$ is weakly acyclic, this algorithm runs in time which is polynomial in the size of I.

For the second part, the following is an NP algorithm to check if XR-certain$(q, I, \mathcal{M})$ is false:

> Guess a subset I' of I. Using the algorithm from the first part, check that I' is a source repair of I. If so, check that $q(\text{chase}(I')) = \text{FALSE}$.

For the matching lower bound, consider the schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ and target conjunctive query q, where $\mathbf{S} = \{\text{P}(x, y), \text{Q}(x, y)\}$, $\mathbf{T} =$

$\{\mathrm{P}'(x,y), \mathrm{Q}'(x,y)\}$, $\Sigma_{\mathrm{st}} = \{\mathrm{P}(x,y) \rightarrow \mathrm{P}'(x,y), \mathrm{Q}(x,y) \rightarrow \mathrm{Q}'(x,y)\}$, and $\Sigma_{\mathrm{t}} = \{\mathrm{P}'(x,y) \wedge \mathrm{P}'(x,y') \rightarrow y = y', \mathrm{Q}'(x,y) \wedge \mathrm{Q}'(x,y') \rightarrow y = y'\}$, and $q = \exists x \exists y \exists x' \mathrm{P}'(x,y) \wedge \mathrm{Q}'(x',y)$. Note that Σ_{st} is the *copy* mapping, therefore, we have XR-certain$(q, I, \mathcal{M}) = \oplus$-CQA$(q, J, \Sigma_{\mathrm{t}})$ where J is merely a copy of I. For the given target query and target constraints, the latter is known to be coNP-hard in data complexity [21, 36]. □

Theorem 3.12 implies that the algorithmic properties of exchange-repairs are quite different from those of $\oplus$-repairs. Indeed, as shown in [3, 28], for weakly acyclic GLAV+(GLAV, EGD) schema mappings, the $\oplus$-repair-checking problem is in coNP (and can be coNP-complete), and the data complexity of the consistent answers of Boolean conjunctive queries is Π_2^{P}-complete [21]. This drop in complexity can be directly attributed to Theorem 3.5.

Interestingly, the data complexity of XR-certain query answering remains coNP-complete even for very restricted constraint languages. In the proof of Theorem 3.12, only copy s-t tgds and target key constraints were necessary to show hardness. Copy tgds are both GAV and LAV, and key constraints are one of the smallest interesting restrictions of EGDs. Moreover, the proof didn't make use of any target tgds at all.

Expressiveness

Perhaps unsurprisingly, XR-certain query answering with projection-free atomic queries is just as expressive as XR-certain query answering with unions of conjunctive queries, as long as our schema mapping language includes GAV target constraints. We now make this fact precise.

Lemma 3.13. *Given a* GLAV+(GLAV, EGD) *schema mapping* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ *and a union of conjunctive queries* $q(\mathbf{x})$:- $(\exists \mathbf{y})\phi_1(\mathbf{x}, \mathbf{y}) \vee \ldots \vee \phi_n(\mathbf{x}, \mathbf{y})$, *de-*

fine $\{t_1, \ldots, t_n\}$ *to be a set of new* GAV *tgds where each* t_k *is* $\phi_k(\mathbf{x}, \mathbf{y}) \rightarrow \mathrm{q}(\mathbf{x})$. *Let* $\mathcal{M}' = (\mathbf{S}, \mathbf{T} \cup \{\mathrm{q}\}, \Sigma_{st}, \Sigma_t \cup \{t_1, \ldots, t_n\})$, *and let* $q'(\mathbf{x}) := \mathrm{q}(\mathbf{x})$. *Then for every source instance* I, *we have that* XR-certain$(q'(\mathbf{x}), I, \mathcal{M}') =$ XR-certain$(q(\mathbf{x}), I, \mathcal{M})$.

Proof. Let (I', J') be an XR-solution for an instance I w.r.t. $\mathcal{M}$. Let J'' be the result of chasing J' with $\{t_1, \ldots, t_n\}$. Then the q-facts in J'' are the answers to q over J. It is easy to see that (I', J'') is an XR-solution for I w.r.t. $\mathcal{M}'$, and thus that XR-certain$(q'(\mathbf{x}), I, \mathcal{M}') =$ XR-certain$(q(\mathbf{x}), I, \mathcal{M})$. □

This result blurs the distinction between XR-certain query answers and facts that are members of every XR-solution. Notice that this result is a reduction, which gives us the following corollary.

Corollary 3.14. *Let* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ *be a weakly acyclic* GLAV+(GLAV, EGD) *schema mapping. Then given an* $\mathbf{S}$*-instance* I, *it is in* coNP *(data complexity) to determine if a fact* f *is contained in the intersection of exchange repair solutions for* I *w.r.t.* $\mathcal{M}$. *Furthermore, there is a schema mapping specified by copy s-t tgds and target* GAV *tgds and key constraints such that, given an* $\mathbf{S}$*-instance* I, *it is* coNP*-complete to determine if a fact* f *is contained in the intersection of exchange repair solutions for* I *w.r.t.* $\mathcal{M}$.

3.3 Discussion

Exchange-repair provides non-trivial, meaningful answers for the case of a source instance with no solution w.r.t. a schema mapping. We have seen that these semantics are suitable for the data exchange problem. In addition to exhibiting the basic properties for inconsistency-tolerant data exchange described in Section 3, XR-certain query answering also has lower data complexity than naïve approaches

based on CQA. We will see in the upcoming chapters that there are several ways to compute XR-certain answers, that doing so may be feasible despite its inherent complexity, and that existing approximation techniques for similar semantics in other problem domains do not apply to data exchange.

Chapter 4

Query Answering Under XR-certain Semantics with GAV tgds

In this chapter, we give algorithms for query answering under the XR-certain semantics. First, we explore the depth of the relationship between exchange-repair and consistent query answering by giving sufficient conditions for use of a CQA solver to solve instances of the XR-certain problem. Then, we show that for GAV+(GAV, EGD) schema mappings, it is possible to use a disjunctive logic program solver to compute XR-certain answers.

4.1 Computing XR-certain Answers with a CQA Solver

In Section 3.1.1, we showed that for the special case of the copy mapping with target egds, the XR-certain answers are equal to the consistent query answers

over a renamed copy of the source instance w.r.t. the target constraints. We will now prove a related but much stronger result which significantly extends the applicability of CQA solvers to XR-certain. We will show that for any GAV+EGD schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$, it is possible to construct a set of egds Σ_s over $\mathbf{S}$ such that an $\mathbf{S}$-instance I is consistent with Σ_s if and only if I has a solution w.r.t. $\mathcal{M}$. We use this to show that XR-certain$(q, I, \mathcal{M})$ for a conjunctive query q coincides with subset-CQA(q_s, I, Σ_s) for a union of conjunctive queries q_s. Thus, we can employ tools for consistent query answering with respect to egds in order to compute XR-certain answers for GAV+EGD schema mappings.

We will use the well-known technique of *GAV unfolding* (see, e.g., [55]).

Definition 4.1 (GAV unfolding)**.** Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a GAV+EGD schema mapping. For each k-ary target relation $T \in \mathbf{T}$, let q_{T} be the set of all conjunctive queries $q(x_1, \ldots, x_k) = \exists \mathbf{y}(\phi(\mathbf{y}) \wedge x_1 = y_{i_1} \wedge \cdots \wedge x_k = y_{i_k})$, for $\phi(\mathbf{y}) \rightarrow T(y_{i_1}, \ldots, y_{i_k})$ a GAV tgd belonging to Σ_{st}.

A *GAV unfolding* of a conjunctive query $q(\mathbf{z}) = T_1(\mathbf{z_1}) \wedge \cdots \wedge T_n(\mathbf{z_n})$ over $\mathbf{T}$ w.r.t. Σ_{st} is a conjunctive query $q'(\mathbf{z}) = \exists \mathbf{y_1}, \ldots \mathbf{y_n}(\phi_1(\mathbf{y_1}) \wedge z_{1_1} = y_{j_{1_1}} \wedge \cdots \wedge z_{1_m} = y_{j_{1_m}} \wedge \cdots \wedge \phi_n(\mathbf{y_n}) \wedge z_{n_1} = y_{j_{n_1}} \wedge \cdots \wedge z_{n_m} = y_{j_{n_m}})$, where each ϕ_i is the conjunction of relational atoms from one of the queries in q_{T_i}.

Similarly, we define a *GAV unfolding* of an egd $T_1(\mathbf{z_1}) \wedge \cdots \wedge T_n(\mathbf{z_n}) \rightarrow z_k = z_l$ over $\mathbf{T}$ w.r.t. Σ_{st} to be an egd over $\mathbf{S}$ given by $\forall \mathbf{y_1}, \ldots \mathbf{y_n}(\phi_1(\mathbf{y_1}) \wedge z_{1_1} = y_{j_{1_1}} \wedge \cdots \wedge z_{1_m} = y_{j_{1_m}} \wedge \cdots \wedge \phi_n(\mathbf{y_n}) \wedge z_{n_1} = y_{j_{n_1}} \wedge \cdots \wedge z_{n_m} = y_{j_{n_m}})$, where each ϕ_i is the conjunction of relational atoms from one of the queries in q_{T_i}.

The *GAV unfolding* of a GAV+EGD schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ is the constraint set Σ_{s} over $\mathbf{S}$ that contains all of the GAV unfoldings of all of the egds in Σ_{t} w.r.t. Σ_{st}.

Figure 4.1 shows the GAV unfolding of the schema mapping and query from

$$\Sigma_{\mathrm{s}} = \left\{ \texttt{Task\_Assignments}(p,t,d) \wedge \texttt{Task\_Assignments}(p,t',d') \rightarrow d = d' \right\}$$

$$\begin{aligned}\texttt{boss}_s(person, stakeholder) = (\exists task, department)& \\ \texttt{Task\_Assignments}(person, task, department)& \\ \wedge\, \texttt{Stakeholders\_old}(task, stakeholder)&\end{aligned}$$

Figure 4.1: The GAV unfolding of the schema mapping and query given in Example 1.1.

Example 1.1. Note that, in this example, both the query and the constraint happen to have exactly one GAV unfolding.

Theorem 4.2. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a* GAV+EGD *schema mapping, and let Σ_s be the set of all GAV unfoldings of egds in Σ_t w.r.t. Σ_{st}. Let I be an $\mathbf{S}$-instance. Then the following are equivalent:*

1. *I satisfies Σ_s if and only if I has a solution w.r.t. $\mathcal{M}$.*
2. *The subset-repairs of I w.r.t. Σ_s are the source repairs of I w.r.t. $\mathcal{M}$.*
3. *For each conjunctive query q over $\mathbf{T}$, we have that* XR-certain$(q, I, \mathcal{M})$ = *subset-*CQA(q_s, I, Σ_s)*, where q_s is the union of* GAV*-unfoldings of q w.r.t. Σ_{st}.*

Proof.

1. Let I be an $\mathbf{S}$-instance which does not satisfy Σ_s. Then there is an egd $\phi_1(\mathbf{x}) \wedge \cdots \wedge \phi_k(\mathbf{x}) \rightarrow x_i = x_j \in \Sigma_{\mathrm{s}}$ which is violated in I by some image $\phi_1(\mathbf{a}) \wedge \cdots \wedge \phi_k(\mathbf{a})$. By the definition of Σ_{s}, there is an egd $T_1(\mathbf{x}) \wedge \cdots \wedge T_k(\mathbf{x}) \rightarrow x_i = x_j$ in Σ_{t}, and tgds $\phi_1(\mathbf{x}) \rightarrow T_1(\mathbf{x}), \ldots, \phi_k(\mathbf{x}) \rightarrow T_k(\mathbf{x})$ in Σ_{st}. Then for any instance J where (I, J) together satisfy Σ_{st}, it holds that J contains the image $T_1(\mathbf{a}) \wedge \cdots \wedge T_k(\mathbf{a})$ and therefore violates Σ_{t}. The proof of the converse is similar.

2. Consider that the source repairs are the maximal subsets of I for which solutions exist. Using the above, we have that these are also the maximal subsets of I which satisfy Σ_{s}, and therefore they are also the subset repairs of I w.r.t. Σ_{s}.

3. By definition XR-certain$(q, I, \mathcal{M})$ is the intersection over $q(J')$ for all XR-solutions (I', J') w.r.t. $\mathcal{M}$ (or in other words, for all source repairs I' and solutions J' for I' w.r.t. $\mathcal{M}$). Observe that this is the intersection of certain$(q, I', \mathcal{M})$ over all source repairs I' w.r.t. $\mathcal{M}$. We will now show that certain$(q, I', \mathcal{M}) = q_s(I')$:

   Let J' be the solution for I' w.r.t. $\mathcal{M}$. Suppose $\mathbf{a}$ is a tuple in $q(J')$. Then there is some image $T_1(\mathbf{a}, \mathbf{b}) \wedge \cdots \wedge T_k(\mathbf{a}, \mathbf{b})$ of q in J', and there are some tgds $\phi_1(\mathbf{x}, \mathbf{y}) \rightarrow T_1(\mathbf{x}, \mathbf{y}), \ldots, \phi_k(\mathbf{x}, \mathbf{y}) \rightarrow T_k(\mathbf{x}, \mathbf{y})$ in Σ_{st} where the image $\phi_1(\mathbf{a}, \mathbf{b}) \wedge \cdots \wedge \phi_k(\mathbf{a}, \mathbf{b})$ is in I'. By definition the clause $\exists y \phi_1(\mathbf{x}, \mathbf{y}) \wedge \cdots \wedge \phi_k(\mathbf{x}, \mathbf{y})$ is in q_s, so $\mathbf{a}$ is in $q_s(I')$. The proof of the converse is similar.

   We now have that XR-certain$(q, I, \mathcal{M})$ is the intersection over $q_s(I')$ for all source repairs I' of I w.r.t. $\mathcal{M}$. By the second item of the theorem, this gives the intersection over $q_s(I')$ for all subset repairs I' of I w.r.t. Σ_{s}, which is simply subset-CQA$(q_s, I, \Sigma_{\text{s}})$.

□

The next result tells us that Theorem 4.2 cannot, in general, be extended to schema mappings containing LAV s-t tgds. First, recall notion of *undirected reachability* in a graph defined by an edge relation E: a vertex a_k is said to be reachable from a_1 if there exists a sequence $a_1, \ldots, a_k$ such that for each (a_i, a_{i+1}) in the sequence, we have that either $(a_i, a_{i+1}) \in E$ or $(a_{i+1}, a_i) \in E$.

Theorem 4.3. *Consider the* LAV+EGD *schema mapping* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$, *where*

- $\mathbf{S} = \{R\}$ *and* $\mathbf{T} = \{T\}$,
- $\Sigma_{st} = \{R(x, y) \rightarrow \exists u\ T(x, u) \wedge T(y, u)\}$, *and*
- $\Sigma_t = \{T(x, y) \wedge T(x, z) \rightarrow y = z\}$.

Consider the query $q(x, y) = \exists z.\ T(x, z) \wedge T(y, z)$ *over* $\mathbf{T}$. *There does not exist a UCQ* q_s *over* $\mathbf{S}$ *and a set of universal first-order sentences (in particular, egds)* Σ_s *such that, for every instance* I, *we have that* XR-certain$(q, I, \mathcal{M}) =$ *subset*-CQA(q_s, I, Σ_s).

Proof of Theorem 4.3. We start by observing that certain$(q, I, \mathcal{M})$ expresses undirected reachability along the relation R:

Claim: For every $\mathbf{S}$-instance I, certain$(q, I, \mathcal{M}) = \{(a, b) \in adom(I) \mid b$ is reachable from a by an undirected R-path$\}$.

The left-to-right inclusion can be proved by induction on the length of the shortest undirected path from a to b, while, for the right-to-left inclusion, it is enough to consider the solution J that contains a null value for each connected component of I, and such that J contains all facts of the form $T(a, N)$ for $a \in adom(I)$, where N is the null value associated to the connected component of I to which a belongs.

Now, suppose for the sake of a contradiction that q_s and Σ_s as described in the statement of the proposition exist. Let k be the number of variables in q_s. Let I be an instance that consists of a directed path of length $k + 1$ from a to b. It follows from the above claim, and from our assumption on q_s and Σ_s, that $(a, b) \in$ subset-CQA(q_s, I, Σ_s), and for every proper subinstance I' of I, we have that $(a, b) \notin$ certain(q, I', M).

Claim: The instance I is consistent with Σ_s.

Suppose for the sake of a contradiction that the above claim does not hold. Let I' be any subset-repair of I with respect to Σ_s. Since I' is a proper sub-instance of I, we have that $(a, b) \notin \text{certain}(q, I', \Sigma)$. In particular, since I' satisfies Σ_s, we have that $(a, b) \notin q_s(I')$. But since I' is a repair of I, this means that $(a, b) \notin \text{subset-CQA}(q_s, I, \Sigma_s)$, a contradiction.

Since $(a, b) \in \text{subset-CQA}(q_s, I, \Sigma_s)$ and I is consistent with Σ_s we have that $(a, b) \in q_s(I)$. That is, there is a homomorphism h from q_s to I. Let I'' be the sub-instance of I consisting of the facts involving only values that are in the image of h. Since I contains k facts and q contains $k+1$ facts, I'' is a proper sub-instance of I. Moreover, since universal first-order sentences are preserved under taking induced sub-instances, every egd true in I is also true in I'' and therefore, I'' is consistent with Σ_s. Finally, by construction, $q_s(I'') = \mathit{true}$. Therefore, $(a, b) \in \text{subset-CQA}(q_s, I'', \Sigma_s)$. This contradicts the fact that $(a, b) \notin \text{certain}(q, I'', M)$. □

The following result tells us that Theorem 4.2 also cannot, in general, be extended to schema mappings containing GAV target tgds. First, recall notion of *directed reachability* in a graph defined by an edge relation E: a vertex a_k is said to be reachable from a_1 if there exists a sequence $a_1, \ldots, a_k$ such that for each (a_i, a_{i+1}) in the sequence, we have that $(a_i, a_{i+1}) \in E$.

Theorem 4.4. *Consider the* GAV+(GAV, EGD) *schema mapping* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$*, where*

- $\mathbf{S} = \{R\}$ *and* $\mathbf{T} = \{T\}$*,*
- $\Sigma_{st} = \{R(x, y) \rightarrow T(x, y)\}$*, and*

- $\Sigma_t = \{T(x,y) \wedge T(y,z) \rightarrow T(x,z)\}$.

Consider the query $q(x,y) = T(x,y)$ *over* $\mathbf{T}$. *There does not exist a UCQ* q_s *over* $\mathbf{S}$ *and a set of universal first-order sentences (in particular, egds)* Σ_s *such that, for every instance* I, *we have that* XR-certain$(q, I, \mathcal{M})$ = *subset*-CQA(q_s, I, Σ_s).

Proof. We start by observing that certain$(q, I, \mathcal{M})$ expresses directed reachability along the relation R: for every $\mathbf{S}$-instance I, certain$(q, I, \mathcal{M}) = \{(a,b) \in adom(I) \mid b$ is reachable from a by a directed R-path$\}$. The claim is proved by induction on the length of the path. The remainder of the proof is identical to that of Theorem 4.3 (the difference between directed paths and undirected paths is inessential to the argument). □

It is worth noting that the schema mappings in the statements of Theorems 4.3 and 4.4 are such that every source instance has a solution, and hence "XR-certain" could be replaced by "certain" in the statements.

4.2 Computing XR-certain Answers with Disjunctive Logic Programming

The approach to computing XR-certain answers described in Section 4.1 applies for GAV schema mappings without target tgds. In this section, we show that XR-certain answers coincide with the cautious answers over stable models of a disjunctive logic program. This approach is suitable for all GAV+(GAV, EGD) schema mappings Disjunctive logic programming is a general-purpose formalism for expressing combinatorially complex problems, for which high quality industrial-strength solvers are available. This makes it a good candidate for computing XR-certain answers.

Disjunctive Logic Programming

We briefly review the basic notions and results of disjunctive logic programming. A *disjunctive logic program* (DLP) Π over a schema $\mathbf{R}$ is a finite collection of rules of the form

$$\alpha_1 \vee \ldots \vee \alpha_n \leftarrow \beta_1, \ldots, \beta_m, \neg\gamma_1, \ldots, \neg\gamma_k.$$

where $n, m, k \geq 0$ and $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m, \gamma_1, \ldots, \gamma_k$ are atoms formed from the relations in $\mathbf{R} \cup \{=\}$, using the constants in Const and first-order variables. A DLP is said to be *positive* if it consists of rules that do not contain negated atoms except possibly for inequalities. A DLP is said to be *ground* if it consists of rules that do not contain any first-order variables. A *model* of Π is an $\mathbf{R}$-instance I over domain Const that satisfies all rules of Π (viewed as universally quantified first-order sentences). A rule in which $n = 0$ is called a constraint, and is satisfied only if its body is not satisfied.

The *stable model* semantics is a widely used semantics of DLPs that are not necessarily positive. For positive DLPs, it coincides with the minimal model semantics. A *minimal model* of Π is a model M of Π such that there does not exist a model M' of Π where the facts of M' form a strict subset of the facts of M. For a ground DLP Π over a schema $\mathbf{R}$ and an $\mathbf{R}$-instance M over the domain Const, the *reduct* Π^M of Π with respect to M is the DLP containing, for each rule $\alpha_1 \vee \ldots \vee \alpha_n \leftarrow \beta_1, \ldots, \beta_m, \neg\gamma_1, \ldots, \neg\gamma_k$, with $M \not\models \gamma_i$ for all $i \leq k$, the rule $\alpha_1 \vee \ldots \vee \alpha_n \leftarrow \beta_1, \ldots, \beta_m$. A *stable model* of a ground DLP Π is an $\mathbf{R}$-instance M over the domain Const such that M is a minimal model of the reduct Π^M. See [39] for more details.

It was shown in [30] that the data complexity of disjunctive logic program-

ming under stable model semantics is Π_2^{P}-complete. These semantics were later extended with classical negation by Gelfond and Lifschitz [40], who defined the *answer set semantics* as a suitable extension of stable model semantics for these programs, which coincides with the stable model semantics for programs without classical negation. They found that the data complexity of the answer set semantics is Π_2^{P}-complete as well. Answer set semantics, also called *answer set programming*, has become a popular declarative formalism for expressing problems with high combinatorial complexity. As a result, several distinct groups of researchers and practitioners have developed answer set solvers. The two most applicable such solvers are DLV [57] and clingo [37]. These are high-quality optimized algorithms for computing answer sets from a program and query specified using a precise syntax. The availability of these solvers makes disjunctive logic programming an attractive formalism, for practical purposes, for representing XR-certain answers.

DLP for GAV+(GAV, EGD) Schema Mappings

We will now give an encoding into DLP for GAV+(GAV, EGD) schema mappings which, along with an encoding of an instance and a query, will serve as a part of the representation of XR-certain answers.

Definition 4.5 (DLP encoding for $\mathcal{M}$)**.** Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$ be a GAV+(GAV, EGD) schema mapping. Then the *DLP encoding for* $\mathcal{M}$, denoted $\Pi_{\mathcal{M}}$, is the DLP program obtained from $\mathcal{M}$ using the construction in Figure 4.2.

The following observations give some intuition about the structure of program $\Pi_{\mathcal{M}}$, which describes the canonical XR-solutions of a source instance. Suppose (I', J') is a canonical XR-solution for some source instance I w.r.t. a GAV+(GAV, EGD) schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$, and let J be the canonical universal solution for I w.r.t. the tgds of $\mathcal{M}$. The program introduces three

For each tgd $(\mathrm{R1}(\mathbf{x},\mathbf{y}) \wedge \cdots \wedge \mathrm{R}n(\mathbf{x},\mathbf{y}) \rightarrow \mathrm{T}(\mathbf{x})) \in (\Sigma_{\mathrm{st}} \cup \Sigma_{\mathrm{t}})$, construct a *chase rule*, *deletion rule*, and *target remainder rule* respectively:

$$\begin{array}{lcl}
\mathtt{T}(\mathbf{x}) & \leftarrow & \mathtt{R1}(\mathbf{x},\mathbf{y}),\ldots,\mathtt{R}n(\mathbf{x},\mathbf{y}). \\
\mathtt{R1_d}(\mathbf{x},\mathbf{y}) \vee \cdots \vee \mathtt{R}n_{\mathtt{d}}(\mathbf{x},\mathbf{y}) & \leftarrow & \mathtt{T_d}(\mathbf{x}),\mathtt{R1}(\mathbf{x},\mathbf{y}),\ldots,\mathtt{R}n(\mathbf{x},\mathbf{y}), \\
 & & \quad \neg\mathtt{R1_i}(\mathbf{x},\mathbf{y}),\ldots,\neg\mathtt{R}n_{\mathtt{i}}(\mathbf{x},\mathbf{y}). \\
\mathtt{T_r}(\mathbf{x}) & \leftarrow & \mathtt{R1_r}(\mathbf{x},\mathbf{y}),\ldots,\mathtt{R}n_{\mathtt{r}}(\mathbf{x},\mathbf{y}).
\end{array}$$

For each egd $(\mathrm{R1}(\mathbf{x}) \wedge \cdots \wedge \mathrm{R}n(\mathbf{x}) \rightarrow x_i = x_j) \in \Sigma_{\mathrm{t}}$, construct a *deletion rule*:

$$\begin{array}{lcl}
\mathtt{R1_d}(\mathbf{x}) \vee \cdots \vee \mathtt{R}n_{\mathtt{d}}(\mathbf{x}) & \leftarrow & \mathtt{R1}(\mathbf{x}),\ldots,\mathtt{R}n(\mathbf{x}), x_i \neq x_j, \\
 & & \quad \neg\mathtt{R1_i}(\mathbf{x}),\ldots,\neg\mathtt{R}n_{\mathtt{i}}(\mathbf{x}).
\end{array}$$

For each relation $\mathrm{R} \in \mathbf{S}$, construct a *source remainder rule*:

$$\mathtt{R_r}(\mathbf{x}) \leftarrow \mathtt{R}(\mathbf{x}), \neg\mathtt{R_d}(\mathbf{x})$$

For each relation $\mathrm{R} \in \mathbf{T}$, construct an *incidental deletion rule*, and the *one-of-three rules*:

$$\begin{array}{lcl}
\mathtt{R_i}(\mathbf{x}) & \leftarrow & \mathtt{R}(\mathbf{x}), \neg\mathtt{R_r}(\mathbf{x}), \neg\mathtt{R_d}(\mathbf{x}) \\
\bot & \leftarrow & \mathtt{R_r}(\mathbf{x}), \mathtt{R_d}(\mathbf{x}) \\
\bot & \leftarrow & \mathtt{R_r}(\mathbf{x}), \mathtt{R_i}(\mathbf{x}) \\
\bot & \leftarrow & \mathtt{R_d}(\mathbf{x}), \mathtt{R_i}(\mathbf{x})
\end{array}$$

Figure 4.2: Procedure to construct the disjunctive logic program $\Pi_{\mathcal{M}}$

predicates for each source relation, one with the original name, meant to contain the facts of I; one subscripted with `d` ("deleted"), meant to contain the facts of $I \setminus I'$; and one subscripted with `r` ("remains"), meant to contain the facts of I'. The program also introduces these same three predicates for each target relation, plus a fourth, subscripted with `i` ("incidentally deleted"), meant to contain the target facts in $J \setminus J'$ that are also contained in some subset $J'' \supseteq J'$ of J that is consistent with Σ_{t} (that is, they are not in a canonical XR-solution but they appear in some other XR-solution).

For every stable model $\mathfrak{M}$ of $\Pi_{\mathcal{M}}$, we denote by $I^{\mathfrak{M}}$ and $J^{\mathfrak{M}}$ the $\mathbf{S}$-instance and $\mathbf{T}$-instance consisting of those facts $R(\mathbf{a})$ in the relevant schema for which

$R_r(\mathbf{a}) \in \mathfrak{M}$.

We now introduce some further notation to aid in the discussion of $\Pi_{\mathcal{M}}$.

Definition 4.6. Let s be an instance over the schema of $\Pi_{\mathcal{M}}$. We denote by $s\big|_{\mathbf{S}}$ the $\mathbf{S}$-restriction of s. Let $r(s) = \{R(\mathbf{a}) \mid \mathtt{R_r}(\mathbf{a}) \in s\}$. Let $d(s) = \{R(\mathbf{a}) \mid \mathtt{R_d}(\mathbf{a}) \in s\}$. Let $i(s) = \{R(\mathbf{a}) \mid \mathtt{R_i}(\mathbf{a}) \in s\}$. That is, $r(s)$, $d(s)$, and $i(s)$ are the $\mathbf{S} \cup \mathbf{T}$-instances given by the remainder part, deleted part, and incidentally deleted part of s, respectively.

Lemma 4.7. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a* GAV+(GAV, EGD) *schema mapping and I a source instance. Let $\Pi_{\mathcal{M}}$ be the disjunctive logic program for $\mathcal{M}$ obtained by the construction in Figure 4.2. Given a canonical* XR-solution *(I', J') for I w.r.t. $\mathcal{M}$, we can construct a stable model s of the program $\Pi \cup I$ such that $r(s)\big|_{\mathbf{S}} = I'$ and $r(s)\big|_{\mathbf{T}} = J'$ (that is, it contains (I', J'), up to renaming).*

Proof. Let J be the canonical universal solution for I w.r.t. the tgds of $\mathcal{M}$. Choose J'' to be a subset repair of J w.r.t. Σ_{t} such that $J' \subseteq J'' \subseteq J$. Since J' is a consistent subset of J w.r.t. Σ_{t}, some such J'' must exist - possibly many. Now construct the instance s as follows: For every atom $R(\mathbf{a}) \in I$, the instance s contains $\mathtt{R}(\mathbf{a})$. For every atom $R(\mathbf{a}) \in I'$, the instance s contains $\mathtt{R_r}(\mathbf{a})$. For every atom $R(\mathbf{a}) \in I \setminus I'$, the instance s contains $\mathtt{R_d}(\mathbf{a})$. For every atom $T(\mathbf{a}) \in J$, the instance s contains $\mathtt{T}(\mathbf{a})$. For every atom $T(\mathbf{a}) \in J'$, the instance s contains $\mathtt{T_r}(\mathbf{a})$. For every atom $T(\mathbf{a}) \in J \setminus J''$, the instance s contains $\mathtt{T_d}(\mathbf{a})$. For every atom $T(\mathbf{a}) \in J'' \setminus J'$, the instance s contains $\mathtt{T_i}(\mathbf{a})$. In other words,

$$\begin{array}{rclcrcl}
s\big|_{\mathbf{S}} & = & I & \textit{and} & s\big|_{\mathbf{T}} & = & J \\
r(s)\big|_{\mathbf{S}} & = & I' & \textit{and} & r(s)\big|_{\mathbf{T}} & = & J' \\
d(s)\big|_{\mathbf{S}} & = & I \setminus I' & \textit{and} & d(s)\big|_{\mathbf{T}} & = & J \setminus J'' \\
i(s)\big|_{\mathbf{S}} & = & \emptyset & \textit{and} & i(s)\big|_{\mathbf{T}} & = & J'' \setminus J'
\end{array}$$

Claim 1. *s is a model of $\Pi \cup I$.*

Since I is a source instance and J is its canonical universal solution w.r.t. the tgds of $\mathcal{M}$, we have that s satisfies the tgd *chase rules* (cf. Figure 4.2). Since I' is a source instance and J' is its canonical universal solution w.r.t. $\mathcal{M}$, we have that s satisfies the *target remainder rules*. Since I' and $I \setminus I'$ are disjoint, and since $I' \subseteq I$, we have that s satisfies the *source remainder rules*. Since J' and $J'' \setminus J'$ and $J \setminus J''$ are disjoint, and since $J' \subseteq J'' \subseteq J$, we have that s satisfies the *incidental deletion rules* and the *one-of-three rules*.

Consider an egd $R1(\mathbf{x}) \wedge \cdots \wedge Rn(\mathbf{x}) \rightarrow x_i = x_j$. If $R1(\mathbf{a}) \wedge \cdots \wedge Rn(\mathbf{a}) \subseteq J$ where $a_i \neq a_j$, then since J'' is consistent, at least one of $R1(\mathbf{a}), \ldots, Rn(\mathbf{a})$ is in $J \setminus J''$, so s satisfies the egd *deletion rules*. Consider a tgd $R1(\mathbf{x}, \mathbf{y}) \wedge \cdots \wedge Rn(\mathbf{x}, \mathbf{y}) \rightarrow T(\mathbf{x})$. If $R1(\mathbf{a}, \mathbf{b}) \wedge \cdots \wedge Rn(\mathbf{a}, \mathbf{b}) \subseteq I \cup J$ and $T(\mathbf{a}) \in J \setminus J''$. For a target tgd, J'' is consistent w.r.t. Σ_{t}, so at least one of $R1(\mathbf{a}, \mathbf{b}), \ldots, Rn(\mathbf{a}, \mathbf{b})$ is in $J \setminus J''$. For a source-to-target tgd, I', J' is consistent w.r.t. Σ_{st}, so at least one of $R1(\mathbf{a}, \mathbf{b}), \ldots, Rn(\mathbf{a}, \mathbf{b})$ is in $I \setminus I'$. Therefore, s satisfies the tgd *deletion rules*.

Claim 2. *s is stable w.r.t. $\Pi \cup I$.*

Consider the reduct $(\Pi \cup I)^s$ of our program, which contains the tgd *chase rules* and *target remainder rules*, the *one-of-three rules*, the source instance I, the tgd and egd *deletion reduct rule* (deletion rules without $_i$ atoms where none of the disjunct atoms is in $J'' \setminus J'$), the *source remainder reduct rule* $\mathtt{R_r}(\mathbf{a}) \leftarrow \mathtt{R}(\mathbf{a})$ for each $R(\mathbf{a}) \in I'$, and the *incidental deletion reduct rule* $\mathtt{R_i}(\mathbf{a}) \leftarrow \mathtt{R}(\mathbf{a})$ for each $R(\mathbf{a}) \in J'' \setminus J'$. We need to show that s is a minimal model of $(\Pi \cup I)^s$.

Suppose $s' \subsetneq s$ is a model of $(\Pi \cup I)^s$. Note that $I \subseteq s'$. By the *source remainder reduct rules*, for every atom $R(\mathbf{a}) \in I'$, we have that s' contains the atom $\mathtt{R_r}(\mathbf{a})$. By the *target remainder rules*, for every atom $T(\mathbf{a}) \in J'$, we have that s' contains the atom $\mathtt{T_r}(\mathbf{a})$. By the *chase rules*, we have that s' contains J.

By the *incidental deletion reduct rules*, for every atom $R(\mathbf{a}) \in J'' \setminus J'$, we have that s' contains the atom $\mathtt{R_i}(\mathbf{a})$. Therefore, s and s' may differ only in the contents of the deletion predicates, so $d(s \setminus s')$ is nonempty.

Note that by the construction of s we have that $d(s \setminus s')$ is disjoint from J''. Let $\hat{J} = J'' \cup d(s \setminus s')$. If $d(s \setminus s')$ is nonempty, then $\hat{J} \supsetneq J''$, and since J'' is defined as a maximal consistent subset of J, it holds that $\hat{J}$ is inconsistent. Therefore one of the following holds true:

- there is a tgd $R1(\mathbf{x}, \mathbf{y}) \wedge \cdots \wedge Rn(\mathbf{x}, \mathbf{y}) \rightarrow T(\mathbf{x})$ such that $R1(\mathbf{a}, \mathbf{b}) \wedge \cdots \wedge Rn(\mathbf{a}, \mathbf{b}) \subseteq \hat{J}$ but $T(\mathbf{a}) \notin \hat{J}$, so s' violates a tgd *deletion reduct rule*; OR
- there is an egd $R1(\mathbf{x}, \mathbf{y}) \wedge \cdots \wedge Rn(\mathbf{x}, \mathbf{y}) \rightarrow x_i = x_j$ such that $R1(\mathbf{a}, \mathbf{b}) \wedge \cdots \wedge Rn(\mathbf{a}, \mathbf{b}) \subseteq \hat{J}$ but $a_i \neq a_j$, so s' violates an egd *deletion reduct rule*.

In both cases, we find that s' is not actually a model of the reduct $(\Pi \cup I)^s$, which is a contradiction. □

Lemma 4.8. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a* GAV+(GAV, EGD) *schema mapping and I a source instance. Let $\Pi_{\mathcal{M}}$ be the disjunctive logic program for $\mathcal{M}$ given in Figure 4.2. Given a stable model s' of the program $\Pi \cup I$, we have that $(r(s')\big|_{\mathbf{S}}, r(s')\big|_{\mathbf{T}})$ is a canonical* XR-solution *for I w.r.t. $\mathcal{M}$ (that is, $r(s')\big|_{\mathbf{S}}$ is a source repair for I w.r.t. $\mathcal{M}$, and $r(s')\big|_{\mathbf{T}}$ is the universal solution obtained from $r(s')\big|_{\mathbf{S}}$ by chasing with $\mathcal{M}$).*

Proof. Let $I' = r(s')\big|_{\mathbf{S}}$ and let $J' = r(s')\big|_{\mathbf{T}}$. Additionally, let $J = s'\big|_{\mathbf{T}}$ and let $J'' = J' \cup i(s')\big|_{\mathbf{T}}$, or in other words, J'' is the set of remainder and incidentally deleted target facts. Since the *target remainder rules* directly encode the tgds of $\mathcal{M}$, $\mathcal{M}$ is GAV, and s' is minimal w.r.t. its reduct, we have that J' is a canonical universal solution for I' w.r.t. the tgds of $\mathcal{M}$. Suppose towards a contradiction that I' is not maximal. Then there exists a maximal subset $\dot{I} \subseteq I$ that strictly

contains I' and has a solution w.r.t. $\mathcal{M}$. Let $\dot{J}$ be the canonical universal solution for $\dot{I}$ w.r.t. $\mathcal{M}$, and note that $(\dot{I}, \dot{J})$ is thus an XR-solution. Since $\dot{I} \supset I'$ and $\mathcal{M}$ is GAV, we have that $\dot{J} \supseteq J'$. Then by Lemma 4.7 there must be a stable model $\dot{s}$ where

$$\begin{array}{rclcrcl}
\dot{s}\big|_{\mathbf{S}} & = & I & \textit{and} & \dot{s}\big|_{\mathbf{T}} & = & J \\
r(\dot{s})\big|_{\mathbf{S}} & = & \dot{I} & \textit{and} & r(\dot{s})\big|_{\mathbf{T}} & = & \dot{J} \\
d(\dot{s})\big|_{\mathbf{S}} & = & I \setminus \dot{I} & \textit{and} & d(\dot{s})\big|_{\mathbf{T}} & = & J \setminus \ddot{J} \\
i(\dot{s})\big|_{\mathbf{S}} & = & \emptyset & \textit{and} & i(\dot{s})\big|_{\mathbf{T}} & = & \ddot{J} \setminus \dot{J}
\end{array}$$

where $\ddot{J}$ is a subset repair of J w.r.t. Σ_{t} that contains $\dot{J}$.

Suppose $J'' \setminus \ddot{J}$ is nonempty. Consider the instance $\hat{s}$ obtained from $\dot{s}$ by switching the facts in $J'' \setminus \ddot{J}$ from deleted to incidentally deleted. It should be clear that the deletion rules are satisfied in $\hat{s}$, and since no fact in $J'' \setminus \ddot{J}$ appears in $\dot{J}$, neither do they appear in any remainder predicate. Therefore the new model $\hat{s}$ is indeed a model, which tells us that there is no deletion rule violated in s' if we omit the deletion facts from $(\dot{I}, \dot{J}) \setminus (I', J')$, and thus s' is not minimal w.r.t. its reduct, which is a contradiction.

In the case where $J'' \setminus \ddot{J}$ is empty, we simply let $\hat{s} = \dot{s}$, and the argument still holds. □

The following theorem and corollary follow from Lemmas 4.7 and 4.8.

Theorem 4.9. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a* GAV+(GAV, EGD) *schema mapping and I a source instance. Let $\Pi_{\mathcal{M}}$ be the disjunctive logic program for $\mathcal{M}$ given in Figure 4.2. The XR-solutions of I w.r.t. $\mathcal{M}$ are precisely those pairs of instances that are of the form (I^M, J^M) for some stable model M of $\Pi_{\mathcal{M}} \cup I$.*

Corollary 4.10. *Let $\mathcal{M}$ be a* GAV+(GAV, EGD) *schema mapping. For every union*

of conjunctive queries q and for every source instance I,

$$\text{XR-certain}(q, I, \mathcal{M}) = \bigcap \{q_r(s) \mid s \textit{ a stable model of } \Pi_{\mathcal{M}} \cup I\}$$

where q_r is the query formed from q by replacing every occurrence of a relation R with R_r.

The disjunctive logic program $\Pi_{\mathcal{M}} \cup I$ has size linear in the combined size of the source instance and schema mapping, and it can be generated in polynomial time. Thus, we have an efficient way to generate a representation of the XR-certain answers of q on I w.r.t. $\mathcal{M}$ as the cautious answers over the stable models of a disjunctive logic program, which is a suitable input for modern answer set solvers such as DLV or clingo. While the GAV unfolding approach discussed in Section 4.1 applies only to GAV+EGD schema mappings, the translation to DLP described here is suitable for the broader class of GAV+(GAV, EGD).

4.3 Discussion

In this chapter, we have seen that XR-certain query answering for GAV+EGD schema mappings can be performed by reduction to consistent query answering. However, that approach does not extend to LAV+EGD schema mappings, nor to GAV+GAV schema mappings. An alternative technique using disjunctive logic programming is appropriate for the encompassing class of GAV+(GAV, EGD) schema mappings. In the next chapter, we will show that the results in this section can be extended to the class of weakly acyclic GLAV+(GLAV, EGD) schema mappings.

Chapter 5

Query Answering for Weakly Acyclic Schema Mappings Under XR-certain Semantics

In this chapter, we give a general technique for extending results pertaining to GAV+(GAV, EGD) schema mappings to the broader class of weakly acyclic GLAV+(GLAV, EGD) schema mappings. We will make use of this result throughout the rest of the chapter.

5.1 Rewriting Weakly Acyclic Schema Mappings Using GAV Constraints

The main theorem of this section serves to extend the applicability of the techniques of Section 4.2 from GAV+(GAV, EGD) schema mappings to the broader class of weakly acyclic GLAV+(GLAV, EGD) schema mappings.

We begin with some notation for expressing the central theorem of this section.

Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be schema mappings with the same source schema. We will write $\mathcal{M}_1 \rightsquigarrow_{\text{UCQ}} \mathcal{M}_2$ if for every UCQ q over the target schema of $\mathcal{M}_1$, there is a UCQ q' over the target schema of $\mathcal{M}_2$ such that for all source instances I, we have XR-certain$(q, I, \mathcal{M}_1)$ = XR-certain$(q', I, \mathcal{M}_2)$.

Theorem 5.1. *For every weakly acyclic* GLAV+(GLAV, EGD) *schema mapping* $\mathcal{M}$ *we can construct a* GAV+(GAV, EGD) *schema mapping* $\mathcal{M}'$ *such that* $\mathcal{M} \rightsquigarrow_{\text{UCQ}} \mathcal{M}'$.

Theorem 5.1 tells us that any algorithm for XR-certain query answering with respect to GAV+(GAV, EGD) schema mappings can be used to compute XR-certain query answers for weakly acyclic GLAV+(GLAV, EGD) schema mappings as well. In particular, the algorithm developed in Section 4.2 can be used to compute XR-certain for weakly acyclic GLAV+(GLAV, EGD) schema mappings. Although Theorem 5.1 is sufficient to make this possible, we will in fact prove a stronger result, given in Theorem 5.5, that applies to schema mappings defined by *second-order tgds*. Second-order tgds serve not only to strengthen the result, but also to make its proof more natural. Given the prevalence of second-order tgds in certain areas of data interoperability (e.g., composition of schema mappings), this strengthening may also serve to extend the applicability of the technique presented here.

5.1.1 Second-order TGDs

Second-order tgds are a natural extension of tgds that was introduced in [34] in the context of schema mapping composition. We recall that definition:

Let $\mathbf{f}$ be a collection of function symbols, each having a designated arity. A *simple term* is a constant from Const or a variable. A *compound term* is a function symbol applied to a list of terms, such that the arity of the function symbol is respected. By an $\mathbf{f}$*-term*, we mean either a simple term, or a compound term built

up from variables and/or constants using the function symbols in $\mathbf{f}$. We will omit $\mathbf{f}$ from the notation when it is understood from context. The *depth* of a term is the maximal nesting of function symbols, with $depth(e) = 0$ when e is a simple term. A *ground term* is a term in which no variables appear.

A *second-order tgd* (SO tgd) over a schema $\mathbf{R}$ is an expression of the form

$$\sigma = \exists \mathbf{f} \, (\forall \mathbf{x_1}(\phi_1 \rightarrow \psi_1) \wedge \cdots \wedge \forall \mathbf{x_n}(\phi_n \rightarrow \psi_n))$$

where $\mathbf{f}$ is a collection of function symbols, and

1. each ϕ_i is a conjunction of (a) atoms $S(y_1, \ldots, y_k)$ where $S \in \mathbf{R}$ and $y_1, \ldots, y_k$ are variables from $\mathbf{x_i}$; and (b) equalities of the form $t_1 = t_2$ where t_1, t_2 are terms over $\mathbf{x_i}$ and $\mathbf{f}$.

2. each ψ_i is a conjunction of atoms $S(t_1, \ldots, t_k)$ where $S \in \mathbf{R}$ and $t_1, \ldots, t_k$ are $\mathbf{f}$-terms built from $\mathbf{x_i}$.

3. each variable in $\mathbf{x_i}$ occurs in a relational atom in ϕ_i.

We say that an $\mathbf{R}$-instance I satisfies σ if there exists a collection of functions $\mathbf{f^0}$ (whose domain and range are $\mathrm{Const} \cup \mathrm{Nulls}$) such that each conjunct $\forall \mathbf{x_i}(\phi_i \rightarrow \psi_i)$ of σ is satisfied in I where each function symbol in $\mathbf{f}$ is interpreted by the corresponding function in $\mathbf{f^0}$. We will write $I \models \sigma$ when this is the case, or, if we wish to make $\mathbf{f^0}$ explicit in the notation, $I \models \sigma \; [\mathbf{f} \mapsto \mathbf{f^0}]$.

A *source-to-target* SO tgd for source schema $\mathbf{S}$ and target schema $\mathbf{T}$ is an SO tgd over $\mathbf{S} \cup \mathbf{T}$, of the above form, where each ϕ_i contains only relation symbols from $\mathbf{S}$ and each ψ_i contains only relation symbols from $\mathbf{T}$. In [34], the term *SO tgd* was defined as always being source-to-target, and target SO tgds were not considered.

An *equality-free* SO tgd (EF SOTGD) is an SO tgd that does not contain term equalities. We denote by SOTGD+(SOTGD, EGD) the class of schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ where Σ_{st} is a set of source-to-target SO tgds over $\mathbf{S}$ and $\mathbf{T}$, and Σ_t is a set of SO tgds and/or egds over $\mathbf{T}$. Other classes of schema mappings, such as SOTGD+SOTGD and EF SOTGD+EF SOTGD, are defined analogously. Note that an important subclass of equality-free SO tgds are the *plain* SO tgds, introduced in [8], in which no terms contain nested functions.

The following proposition allows us to move freely between discussions involving a single SO tgd (which is the usual presentation), and those involving a set of SO tgds (which we allow here for parity with the other classes of constraints used in this work).

Proposition 5.2. *Let Σ be a set of SO tgds $\{\sigma_1, \ldots, \sigma_k\}$. Then $\bigwedge \Sigma$ is logically equivalent to a single SO tgd.*

Proof. Assume without loss of generality that the SO tgds $\sigma_1, \ldots, \sigma_k$ utilize disjoint sets of function symbols. Then $\bigwedge \Sigma$ is logically equivalent to $\exists \mathbf{f}\,(\tau_1 \wedge \cdots \wedge \tau_m)$, where $\mathbf{f}$ is the set of all function symbols appearing in $\sigma_1, \ldots, \sigma_k$, and $\tau_1, \ldots, \tau_m$ is the collection of all clauses of the form $\forall \mathbf{x_1}(\phi_1 \rightarrow \psi_1)$ appearing in $\sigma_1, \ldots, \sigma_k$. □

Proposition 5.3 ([34])**.** *Every tgd is logically equivalent to an equality-free SO tgd, which can be obtained from it by* skolemization.

Proof. Let σ be the arbitrary tgd $\forall x_1, \ldots, x_m \Phi(x_1, \ldots, x_m) \rightarrow \exists y_1, \ldots, y_n \Psi(x_1, \ldots, x_m, y_1, \ldots, y_n)$. Then it is logically equivalent to its skolemization $\exists f_1, \ldots, f_n(\forall x_1, \ldots, x_m \Phi(x_1, \ldots, x_m) \rightarrow \Psi(x_1, \ldots, x_m, f_1(x_1, \ldots, x_m), \ldots, f_n(x_1, \ldots, x_m)))$, which is an equality-free SO tgd. □

In [34], this result was stated only for the case of source-to-target tgds. Figure 5.2 shows the skolemization of the example schema mapping in Figure 5.1, and Figure 5.3 shows a minor logically equivalent transformation of the same, in which the right-hand side of each clause of each SO tgd consists of a single atom.

Moreover, if we adapt the concept of weak acyclicity to SO tgds in the appropriate way, then every weakly acyclic set of tgds is logically equivalent to a weakly acyclic SO tgd.

More precisely, we say that a set Σ of SO tgds is weakly acyclic if there is no cycle in its dependency graph containing a special edge, where the dependency graph associated to a set of SO tgds is defined as follows:

1. the directed graph whose nodes are positions (R, i) where R is a relation symbol and i is an attribute position of R (as before)

2. there is a normal edge from (R, i) to (S, j) if Σ contains a SO tgd of the form

$$\sigma = \exists \mathbf{f} \left(\forall \mathbf{x_1}(\phi_1 \rightarrow \psi_1) \wedge \cdots \wedge \forall \mathbf{x_n}(\phi_n \rightarrow \psi_n) \right)$$

   and for some $i \leq n$, ϕ_i contains a variable in position (R, i) and ψ_i contains the same variable as a simple term in position (S, j).

3. there is a special edge from (R, i) to (S, j) if Σ contains a SO tgd of the form

$$\sigma = \exists \mathbf{f} \left(\forall \mathbf{x_1}(\phi_1 \rightarrow \psi_1) \wedge \cdots \wedge \forall \mathbf{x_n}(\phi_n \rightarrow \psi_n) \right)$$

   and for some $i \leq n$, ϕ_i contains a variable in position (R, i) and ψ_i contains a compound term in position (S, j) containing the same variable.

We then have:

Proposition 5.4. *Every weakly acyclic* GLAV+(GLAV, EGD) *schema mapping is*

$$
\begin{array}{lcl}
\mathbf{S} & = & \{\mathrm{R}\} \\
\mathbf{T} & = & \{\mathrm{T}\} \\
\Sigma_{\mathrm{st}} & = & \left\{\ \mathrm{R}(x,y) \ \rightarrow\ \exists u(\mathrm{T}(x,u) \wedge \mathrm{T}(y,u))\ \right\} \\
\Sigma_{\mathrm{t}} & = & \left\{\ \mathrm{T}(x,y) \wedge \mathrm{T}(x,y') \rightarrow y = y'\ \right\} \\
\mathrm{q}(x,y) := \exists u(\mathrm{T}(x,u) \wedge \mathrm{T}(y,u)) & &
\end{array}
$$

Figure 5.1: An example schema mapping and query.

$$
\begin{array}{lcl}
\mathbf{S} & = & \{\mathrm{R}\} \\
\mathbf{T} & = & \{\mathrm{T}\} \\
\Sigma_{\mathrm{st}} & = & \left\{\exists f\left(\ \mathrm{R}(x,y) \ \rightarrow\ \mathrm{T}(x,f(x,y)) \wedge \mathrm{T}(y,f(x,y))\ \right)\right\} \\
\Sigma_{\mathrm{t}} & = & \left\{\ \mathrm{T}(x,y) \wedge \mathrm{T}(x,y') \rightarrow y = y'\ \right\} \\
\mathsf{q}(x,y) := \exists u(\mathrm{T}(x,u) \wedge \mathrm{T}(y,u)) & &
\end{array}
$$

Figure 5.2: Result of skolemizing the schema mapping in Figure 5.1.

logically equivalent to a weakly acyclic EF SOTGD+(EF SOTGD, EGD) *schema mapping.*

Proof. Let $\mathcal{M}'$ be the EF SOTGD+(EF SOTGD, EGD) schema mapping obtained from $\mathcal{M}$ by skolemization of each of the tgds in $\mathcal{M}$ (and simply copying the egds). Since every tgd is logically equivalent to its skolemization, we have that $\mathcal{M}$ is logically equivalent to $\mathcal{M}'$. Moreover, when a tgd σ in $\mathcal{M}$ gives rise to a normal edge in the dependency graph of $\mathcal{M}$, its skolemization likewise gives rise to a normal edge in the dependency graph of $\mathcal{M}'$ (since neither position is affected by skolemization), and when σ gives rise to a special edge in the dependency graph of $\mathcal{M}$, in the skolemization of σ, the target position is a skolem term whose arguments are the forwarded variables in σ, which gives rise to a special edge also in the dependency graph of $\mathcal{M}'$. Thus, it is easy to see that $\mathcal{M}$ and $\mathcal{M}'$ have the same dependency graph, so $\mathcal{M}'$ is also weakly acyclic. □

In the remainder of this section, we will establish:

$$
\begin{array}{lcl}
\mathbf{S} & = & \{\mathrm{R}\} \\
\mathbf{T} & = & \{\mathrm{T}\} \\
\Sigma_{\mathrm{st}} & = & \left\{ \exists f \left(\begin{array}{lcl} \mathrm{R}(x,y) & \rightarrow & \mathrm{T}(x, f(x,y)) \wedge \\ \mathrm{R}(x,y) & \rightarrow & \mathrm{T}(y, f(x,y)) \end{array} \right) \right\} \\
\Sigma_{\mathrm{t}} & = & \left\{ \; \mathrm{T}(x,y) \wedge \mathrm{T}(x,y') \rightarrow y = y' \; \right\} \\
\multicolumn{3}{l}{\mathtt{q}(x,y) := \exists u(\mathrm{T}(x,u) \wedge \mathrm{T}(y,u))}
\end{array}
$$

Figure 5.3: Result of splitting the right-hand side of tgds in the skolemized schema mapping from Figure 5.2.

Theorem 5.5. *For every weakly acyclic* SOTGD+(SOTGD, EGD) *schema mapping* $\mathcal{M}$ *there is a* GAV+(GAV, EGD) *schema mapping* $\mathcal{M}'$ *such that* $\mathcal{M} \rightsquigarrow_{\mathrm{UCQ}} \mathcal{M}'$.

5.1.2 Eliminating Equalities to Establish Freeness

In this section, we will rewrite our schema mapping to eliminate egds as well as equality conditions in SO tgds. This allows us to work with solutions in which there is a one-to-one correspondence between ground terms (of any depth) and their values. This property, called *freeness* (which we define below), is used in Section 5.1.3. The meaning of the egds in our schema mapping will be captured by a set of tgds, and in a later step the semantics of those egds will be restored. First, we will show that every SOTGD+(SOTGD, EGD) schema mapping is, for the purpose of XR-certain query answering, equivalent to an EF SOTGD+(SOTGD, EGD) schema mapping. This simplifies the proofs for the results that follow.

Proposition 5.6. *For every* SOTGD+(SOTGD, EGD) *schema mapping* $\mathcal{M}$ *we can construct an* EF SOTGD+(SOTGD, EGD) *schema mapping* $\mathcal{M}'$ *such that* $\mathcal{M} \rightsquigarrow_{\mathrm{UCQ}} \mathcal{M}'$.

Proof. Let Σ_{copy} be the copy mapping for $\mathbf{S}$ (see Definition 3.6). Note that the tgds in Σ_{copy} are GAV. Construct a schema mapping $\mathcal{M}' = (\mathbf{S}, \mathbf{T}', \Sigma'_{\mathrm{st}}, \Sigma'_{\mathrm{t}})$, where $\mathbf{T}' = \mathbf{T} \cup \{\mathrm{R}' \mid \mathrm{R} \in \mathbf{S}\}$; and Σ'_{st} is the equality-free SO tgd

$\exists \mathbf{f}\,(\forall \mathbf{x_1}(\phi_1 \rightarrow \psi_1) \wedge \cdots \wedge \forall \mathbf{x_n}(\phi_n \rightarrow \psi_n))$ where $\mathbf{f} = \emptyset$ and there is one conjunct $\forall \mathbf{x_i}(\phi_i \rightarrow \psi_i)$ for each (GAV) tgd $\forall \mathbf{x_i}(\phi_i \rightarrow \psi_i)$ in Σ_{copy}; and $\Sigma'_{\text{t}} = \Sigma_{\text{t}} \cup \{\sigma' \mid \sigma \in \Sigma_{\text{st}}\}$, where σ' is a copy of σ in which every occurrence of a relation $\text{R} \in \mathbf{S}$ is replaced by R'.

Let I be an arbitrary $\mathbf{S}$-instance. We will show that the set of solutions for I w.r.t. $\mathcal{M}$ is equal to the set composed of the $\mathbf{T}$-restriction of each solution for I w.r.t. $\mathcal{M}'$ (we will show containment in both directions). For the forward direction ($\subseteq$), let I be an $\mathbf{S}$-instance and let J be a solution for I w.r.t. $\mathcal{M}$. Let I' be the copy of I. Then since I and J together satisfy $\Sigma_{\text{st}} \cup \Sigma_{\text{t}}$, it is easy to see that (I', J) is a solution for I w.r.t. $\mathcal{M}'$. For the reverse direction ($\supseteq$), let I be an $\mathbf{S}$-instance and let (I', J) be a solution for I w.r.t. $\mathcal{M}'$, where $I' = \{\text{R}' \mid \text{R} \in \mathbf{S}\}\Big|_{(I',J)}$ and $J = \mathbf{T}\Big|_{(I',J)}$. Then since (I', J) satisfies Σ'_{t}, it is easy to see that J is a solution for I w.r.t. $\mathcal{M}$.

We can now conclude the following: (1) If I has a solution w.r.t. $\mathcal{M}$, then I has a solution w.r.t. $\mathcal{M}'$. (2) The set of source repairs for I w.r.t. $\mathcal{M}$ is equal to the set of source repairs for I w.r.t. $\mathcal{M}'$. (1) For every UCQ q on $\mathbf{T}$, we have $\text{certain}(q, I, \mathcal{M}) = \text{certain}(q, I, \mathcal{M}')$. Therefore, for every UCQ q on $\mathbf{T}$, the XR-certain answers to q on I w.r.t. $\mathcal{M}$ are equal to the XR-certain answers to q on I w.r.t. $\mathcal{M}'$. □

We may now restrict attention to EF SOTGD+(SOTGD, EGD) schema mappings.

Definition 5.7 (Equality Singularization)**.** Fix a fresh binary relation symbol Eq.

- The *equality singularization* of a conjunctive query $q(\mathbf{x}) = \exists \mathbf{y} \phi(\mathbf{x}, \mathbf{y})$, denoted by $q^{\text{Eq}}(\mathbf{x})$, is the conjunctive query $\exists \mathbf{y}\mathbf{z} \phi'(\mathbf{x}, \mathbf{y}, \mathbf{z})$ obtained from q as follows: whenever a variable u (free or quantified) occurs more than once in ϕ, we replace each occurence other than the first occurrence by a fresh distinct existentially quantified variable $z \in \mathbf{z}$ and we add the atom $\text{Eq}(u, z)$.

- The *equality singularization* of an egd $\sigma = \forall \mathbf{x}(\phi \to x_i = x_j)$ is the GAV tgd

$$\sigma^{\mathrm{Eq}} = \forall \mathbf{x}\mathbf{z}(\phi' \to \mathrm{Eq}(x_i, x_j))$$

  where $\phi^{\mathrm{Eq}} = \exists \mathbf{z}\phi'$

- The *equality singularization* of an SO tgd $\sigma = \exists \mathbf{f} \bigwedge_{i=1\ldots n} (\forall \mathbf{x}_i(\phi_i \wedge \alpha_i \to \psi_i))$ (where each ϕ_i is a conjunction of relational atoms and each α_i is a conjunction of equalities) is the equality-free SO tgd

$$\sigma' = \exists \mathbf{f} \bigwedge_{i=1\ldots n} (\forall \mathbf{x}_i \mathbf{z}_i(\phi'_1 \wedge \alpha'_i \to \psi_i))$$

  where $\phi_i^{\mathrm{Eq}} = \exists \mathbf{z}_i \phi'$ and α'_i is obtained from α_i by replacing each equality $s = t$ by $\mathrm{Eq}(s, t)$.

- The *equality singularization* of a EF SOTGD+(SOTGD, EGD) schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$ is the EF SOTGD+EF SOTGD schema mapping

$$\mathcal{M}^{\mathrm{Eq}} = (\mathbf{S}, \mathbf{T} \cup \{\mathrm{Eq}\}, \Sigma_{\mathrm{st}}, \{\sigma^{\mathrm{Eq}} \mid \sigma \in \Sigma_{\mathrm{t}}\} \cup \mathrm{eqAx}(\mathbf{T}))$$

  where $\mathrm{eqAx}(\mathbf{T})$ is the set of tgds of the form $\mathrm{T}(x_1, \ldots, x_n) \to \mathrm{Eq}(x_i, x_i)$ for $1 \leq i \leq n$ where T is a relation in $\mathbf{T}$, along with the tgds $\mathrm{Eq}(x_1, x_2) \to \mathrm{Eq}(x_2, x_1)$ and $\mathrm{Eq}(x_1, x_2) \wedge \mathrm{Eq}(x_2, x_3) \to \mathrm{Eq}(x_1, x_3)$.

Figure 5.4 shows equality singularization in action.

Proposition 5.8. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be an* EF SOTGD+(SOTGD, EGD) *schema mapping and let $\mathcal{M}^{\mathrm{Eq}}$ be its equality singularization. For every $\mathbf{S}$-instance I,*

$$
\begin{array}{lcl}
\mathbf{S} & = & \{\mathrm{R}\} \\
\mathbf{T} & = & \{\mathrm{T}, \mathrm{Eq}\} \\
\Sigma_{\mathrm{st}} & = & \left\{ \exists f \left(\begin{array}{rcl} \mathrm{R}(x,y) & \rightarrow & \mathrm{T}(x, f(x,y)) \wedge \\ \mathrm{R}(x,y) & \rightarrow & \mathrm{T}(y, f(x,y)) \end{array} \right) \right\} \\
\Sigma_{\mathrm{t}}^{\mathrm{Eq}} & = & \left\{ \begin{array}{l} \mathrm{T}(x,y) \wedge \mathrm{Eq}(x,x') \wedge \mathrm{T}(x',y') \rightarrow \mathrm{Eq}(y,y') \\ \mathrm{T}(x,y) \rightarrow \mathrm{Eq}(x,x) \\ \mathrm{T}(x,y) \rightarrow \mathrm{Eq}(y,y) \\ \mathrm{Eq}(x_1,x_2) \rightarrow \mathrm{Eq}(x_2,x_1) \\ \mathrm{Eq}(x_1,x_2) \wedge \mathrm{Eq}(x_2,x_3) \rightarrow \mathrm{Eq}(x_1,x_3) \end{array} \right\} \\
\multicolumn{3}{l}{\mathrm{q}^{\mathrm{Eq}}(x,y) := \exists u, u' (\mathrm{T}(x,u) \wedge \mathrm{Eq}(u,u') \wedge \mathrm{T}(y,u'))}
\end{array}
$$

Figure 5.4: Equality singularization of the schema mapping and query from Figure 5.3.

1. *I has a solution w.r.t. $\mathcal{M}$ if and only if I has a solution J' w.r.t. $\mathcal{M}^{\mathrm{Eq}}$ such that there is no pair of distinct constants a, b where $J' \models \mathrm{Eq}(a,b)$.*

2. *If I has a solution w.r.t. $\mathcal{M}$, then for every UCQ q over $\mathbf{T}$, $\mathrm{certain}(q, I, \mathcal{M}) = \mathrm{certain}(q^{\mathrm{Eq}}, I, \mathcal{M}^{\mathrm{Eq}})$.*

Proof of Proposition 5.8. [$\Rightarrow$] Let J be any $\mathbf{T}$-instance that is a solution for I with respect to $\mathcal{M}$. Take J' to be the $\mathbf{T} \cup \{Eq\}$-instance that extends J with all facts of the form $Eq(a,a)$ with $a \in adom(J)$. It is easy to see that J' is a solution for I with respect to $\mathcal{M}^{\mathrm{Eq}}$, and that, for all UCQs q, we have that $q(J)\downarrow = q^{\mathrm{Eq}}(J')\downarrow$, that is, the null-free answers to q on J are equal to the null-free answers to q^{Eq} on J'. Moreover, it is immediate from the construction of J' that there is no pair of distinct constants a, b where $J' \models Eq(a,b)$.

[$\Leftarrow$] Let $\mathbf{f}$ be the collection of function symbols appearing in $\mathcal{M}^{\mathrm{Eq}}$. Let J be a $\mathbf{T} \cup \{\mathrm{Eq}\}$-instance that is a solution for I with respect to $\mathcal{M}^{\mathrm{Eq}}$, such that there is no pair of distinct constants a, b where $J \models \mathrm{Eq}(a,b)$. Note that Eq is an equivalence relation and that each equivalence class contains at most one constant (but possibly many null values). Let $\mathbf{f^0}$ be a witnessing collection of functions, such that $(I,J) \models \mathcal{M}^{\mathrm{Eq}}\ [\mathbf{f} \mapsto \mathbf{f^0}]$. We will construct a $\mathbf{T}$-instance J' and a

collection $\mathbf{f^1}$ of functions, as follows:

- For every Eq-equivalence class, choose a single representative member. If an equivalence class contains a constant, we use that constant as the representative member. For every value $u \in adom(J)$, denote by $\pi(u)$ the representative member of the Eq-equivalence class to which u belongs.
- J' contains, for every fact $\mathrm{T}(v_1, \ldots, v_n)$ of J (where $\mathrm{T} \in \mathbf{T}$), the corresponding fact $\mathrm{T}(\pi(v_1), \ldots, \pi(v_n))$
- $\mathbf{f^1}$ contains, for each function f in $\mathbf{f^0}$, the corresponding function f' given by $f'(\mathbf{u}) = \pi(f(\mathbf{u}))$.

By construction, we have that, for any image $q^{\mathrm{Eq}}(\mathbf{a})$ in J of the equality singularization of a conjunctive query $q(\mathbf{x})$, we have an image $q(\mathbf{a})$ in J', and vice versa. This tells us both that J' is a solution for I w.r.t. $\mathcal{M}$ and that for any UCQ q over $\mathbf{T}$, we have $q(J')\downarrow = q^{\mathrm{Eq}}(J)\downarrow$. Additionally, since each Eq-class is represented by a single member in J', we have that, for each egd in $\sigma \in \Sigma_{\mathrm{t}}$, the fact that J satisfies σ^{Eq} implies that J' satisfies σ. □

The importance of Proposition 5.8 comes from the following observation. Consider any SOTGD+SOTGD schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$. Let $\mathbf{f}$ be the collection of all function symbols occurring in SO tgds in $\Sigma_{st} \cup \Sigma_t$. A function g is called *injective* if $\forall\mathbf{x}\forall\mathbf{y}\,(g(\mathbf{x}) = g(\mathbf{y}) \rightarrow \mathbf{x} = \mathbf{y})$. Two functions g and h have *mutually disjoint ranges* if $\forall\mathbf{x}\forall\mathbf{y}\,(g(\mathbf{x}) \neq h(\mathbf{y}))$. A solution J for a source instance I with respect to $\mathcal{M}$ is said to be a *free* solution if there is a collection of functions $\mathbf{f^0}$ such that $(I, J) \models \Sigma_{st} \cup \Sigma_t\ [\mathbf{f} \mapsto \mathbf{f^0}]$, and such that each function in $\mathbf{f^0}$ is injective and all pairs of functions in $\mathbf{f^0}$ have mutually disjoint ranges. Equivalently, in a free solution, each value in $adom(J)$ is the denotation of exactly one ground term. If, furthermore, we have that each value in $adom(J)$ is the denotation of a (unique)

ground term of depth at most k, then we say that J is a *free solution of rank k*. Note that there is a distinction between the notion of *rank* for a weakly acyclic schema mapping, and the notion of *rank* for a free solution. They are related but the latter may be a finite multiple of the former - see Proposition 5.10.

To give some intuition, we could instantiate each function symbol $f \in \mathbf{f}$ with the function that maps every input to a string containing its own syntactic representation. For example, given binary function symbol f, unary function symbol g, and two constants a and b, the term $g(a)$ is mapped to the syntax string "$g(a)$", and the term $f(g(a), b)$ is mapped to the syntax string "$f(\text{“}g(a)\text{”}, b)$". Clearly, these functions are injective and every pair of them has mutually disjoint ranges.

The following is the definition of a chase step with respect to an equality-free SO tgd.

Definition 5.9. Let σ be an equality-free SO tgd of the form $\exists f((\forall \mathbf{x_1}(\phi_1 \rightarrow \psi_1)) \wedge \cdots \wedge (\forall \mathbf{x_n}(\phi_n \rightarrow \psi_n)))$, and let C_i be the i^{th} conjunct of σ, so C_i is $(\forall \mathbf{x_i}(\phi_i \rightarrow \psi_i))$.

Let K be an instance. Let d be some clause C_j in σ. Let h be a homomorphism from ϕ_j to K such that there is no extension of h to a homomorphism h' from $\phi_j \wedge \psi_j$ to K. We say that *d can be applied to K with homomorphism h*. Let h' be the homomorphism obtained by extending h such that each compound term in ψ_j is mapped to a unique denotation. Let K' be the union of K with the image of the atoms of ψ under h'. We say that *the result of applying d to K with h* is K', and write $K \xrightarrow{d,h} K'$. This is called a *second-order chase step*.

Chase sequences and finite chases for SO tgds are unaltered from the definitions in Chapter 2. Notably, there is no way for the finite chase of an equality-free SO tgd to fail.

Proposition 5.10. *Let $\mathcal{M}$ be a weakly acyclic* EF SOTGD+EF SOTGD *schema mapping. There is a natural number $k \geq 0$ such that every source instance I has*

a free universal solution J of rank k.

Proof. Let $\mathbf{f^0}$ be an arbitrary collection of injective functions where all pairs of functions in $\mathbf{f^0}$ have mutually disjoint ranges. Let J be the result of chasing I with the SO tgds of $\mathcal{M}$ using these functions. A priori, J is potentially infinite. However, we can show that J is always finite, moreover, of finite rank.

Base Case The initial instance J_0 is equal to $\emptyset$, which has rank 0.

Inductive Step Consider a chase step $J_k \xrightarrow{d_k, h_k} J_{k+1}$, where each position (R, i) in J_k has rank at most the maximal number of special edges on an incoming path to (R, i) in the dependency graph *times* the maximal depth of a term occurring on the right-hand side of an SO tgd. For every position (S, j) on the right-hand side of d_k, either:

- There is a normal edge from (R, i) to (S, j), in which case the maximal depth of terms denoted by values in position (S, j) in J_{k+1} remains at most the maximal number of special edges on an incoming path to (S, j) in the dependency graph *times* the maximal depth of a term occurring on the right-hand side of an SO tgd.
- There is a special edge from (R, i) to (S, j), in which case the maximal depth of terms denoted by values in position (R, i) in J_k is at most one less than the maximal number of special edges on an incoming path to (S, j) in the dependency graph *times* the maximal depth of a term occurring on the right-hand side of an SO tgd, and therefore the maximal depth of terms denoted by values in position (S, j) in J_{k+1} is at most the maximal number of special edges on an incoming path to (S, j) in the dependency graph *times* the maximal depth of a term occurring on the right-hand side of an SO tgd.

□

It is not hard to see that the same does not hold in the presence of egds.

5.1.3 The Skeleton Rewriting Step

In this section, we will see how to remove compound terms from the schema mapping, and thus eliminate the need for second-order tgds. This transformation depends upon the schema mapping admitting free universal solutions. Fortunately, Propositions 5.6 and 5.10 tell us that for any SOTGD+(SOTGD, EGD) schema mapping, we can construct a schema mapping that is equivalent (for computing XR-certain query answers) and that admits free universal solutions.

Suppose $\mathcal{M}$ is a weakly acyclic EF SOTGD+SOTGDEGD schema mapping, and $\mathcal{M}^{\text{Eq}}$ is the equality singularization of $\mathcal{M}$. Since $\mathcal{M}^{\text{Eq}}$ admits free universal solutions, we can represent the value of every compound term simply by its syntax. This makes it possible to rewrite $\mathcal{M}^{\text{Eq}}$ in such a way that the syntax of compound terms is captured using specialized relations, and constraints with only simple terms.

The *skeleton* of a term is the expression obtained by replacing all constants and variables by $\bullet$, where $\bullet$ is a fixed symbol that is not a function symbol [7]. Thus, for example, the skeleton of $f(g(x, y), z)$ is $f(g(\bullet, \bullet), \bullet)$. The *arity* of a skeleton s, denoted by $arity(s)$, is the number of occurrences of $\bullet$, and the depth of a skeleton is defined in the same way as for terms. If $s, s'_1, \ldots, s'_k$ are skeletons with $arity(s) = k$, then we denote by $s(s'_1, \ldots, s'_k)$ the skeleton of arity $arity(s'_1) + \cdots + arity(s'_k)$ obtained by replacing, for each $i \leq k$, the i-th occurrence of $\bullet$ in s by s'_i.

Definition 5.11 (skeleton rewriting)**.** Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a weakly acyclic EF SOTGD+EF SOTGD schema mapping with rank r and whose most deeply nested term has depth d. Let Θ be the set of functions appearing in Σ_{t}. Define the

skeleton rewriting of $\mathcal{M}$ as the schema mapping $\mathcal{M}^{\text{skel}} = (\mathbf{S}, \mathbf{T}^{\text{skel}}, \Sigma_{\text{st}}^{\text{skel}}, \Sigma_{\text{t}}^{\text{skel}})$, where:

- For every n-ary relation $\mathrm{T} \in \mathbf{T}$, the schema $\mathbf{T}^{\text{skel}}$ contains all relations of the form $\mathrm{T}_{s_1,\ldots,s_n}$, where $s_1, \ldots s_n$ are skeletons of depth less than or equal to r.

- For every clause $\phi(\mathbf{x}) \rightarrow \mathrm{T}(\tau_1, \ldots, \tau_n)$ of a s-t EF SOTGD in Σ_{st}, the constraint set $\Sigma_{\text{st}}^{\text{skel}}$ contains the s-t tgd $\phi(\mathbf{x}) \rightarrow \mathrm{T}_{s_1,\ldots,s_n}(\bar{\mathbf{x}})$, where $s_1, \ldots, s_n$ are the skeletons for $\tau_1, \ldots, \tau_n$ respectively, and $\bar{\mathbf{x}}$ is the sequence of variables in $\tau_1, \ldots, \tau_n$.

- For every clause $\phi(\mathbf{x}) \rightarrow \mathrm{T}(\tau_1, \ldots, \tau_n)$ of an EF SOTGD in Σ_{t} (where $\mathbf{x} = x_1, \ldots, x_m$), the constraint set $\Sigma_{\text{t}}^{\text{skel}}$ contains the tgd $\phi_{s_1,\ldots,s_m}(\mathbf{y}_1, \ldots, \mathbf{y}_m) \rightarrow \mathrm{T}_{s'_1,\ldots,s'_n}(\bar{\mathbf{y}}_\mathbf{1}, \ldots, \bar{\mathbf{y}}_\mathbf{n})$, where
  - $s_1, \ldots, s_m$ are Θ-skeletons of depth at most $r * d$;
  - each $\mathbf{y}_i$ is a sequence of arity(s_i) fresh variables;
  - $\phi_{s_1,\ldots,s_m}(\mathbf{y}_1, \ldots, \mathbf{y}_m)$ is obtained from ϕ by replacing each atom $\mathrm{R}(x_1, \ldots, x_l)$ by $\mathrm{R}_{s_1,\ldots,s_l}(\mathbf{y}_1, \ldots, \mathbf{y}_l)$;
  - s'_i is a Θ-skeleton of depth at most $r * d$ such that $s'_i = s_k$ (if τ_i is the term x_k) and $s'_i = t_i(s_1, \ldots, s_m)$ (if τ_i is the term $t_i(x_1, \ldots, x_m)$)
  - $\bar{\mathbf{y}}_\mathbf{i} = (\mathbf{y}_{k_1}, \ldots, \mathbf{y}_{k_{arity(s_k)}})$ (if τ_i is the term x_k) and $\bar{\mathbf{y}}_\mathbf{i} = (y_{1_1}, ..., y_{1_{arity(s_1)}}, ..., y_{m_1}, ..., y_{m_{arity(s_m)}})$ (if τ_i is the term $t_i(x_1, \ldots, x_m)$)

In addition, for each conjunctive query $q(\mathbf{x}) = \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$ over $\mathbf{T}$ with $\mathbf{x} = x_1, \ldots, x_n$ and $\mathbf{y} = y_1, \ldots, y_m$, we denote by $q^{\text{skel}}(\mathbf{x})$ the union of conjunctive queries over $\mathbf{T}^{\text{skel}}$ of the form $\exists \mathbf{z}_1 \ldots \mathbf{z}_m \psi_{s_1,\ldots,s_n,s'_1,\ldots,s'_m}(x_1, \ldots, x_n, \mathbf{z}_1, \ldots, \mathbf{z}_m)$,

where $s_1 = \ldots = s_n = \bullet$; $s'_1, \ldots, s'_m$ are Θ-skeletons of depth at most r; and each $\mathbf{z}_i$ is a sequence of fresh variables of length $arity(s'_i)$.

For example, consider the EF SOTGD+EF SOTGD schema mapping $\mathcal{M}$ whose constraints are $\exists f \forall x, y \ P(x, y) \rightarrow Q(f(y), y, y)$ and $\exists g \forall x, y, z \ Q(x, y, z) \rightarrow Q(x, y, g(x, y))$. Then $\mathcal{M}^{\text{skel}}$ will include the s-t tgd $P(x, y) \rightarrow Q_{f(\bullet),\bullet,\bullet}(x, y, y)$, and the target tgds $Q_{\bullet,\bullet,\bullet}(x, y, z) \rightarrow Q_{\bullet,\bullet,g(\bullet,\bullet)}(x, y, x, y)$, and $Q_{f(\bullet),\bullet,\bullet}(x, y, z) \rightarrow Q_{f(\bullet),\bullet,g(f(\bullet),\bullet)}(x, y, x, y)$.

The full skeleton rewriting of our running example schema mapping is given in Figure 5.5.

Remark 5.12. An optimized version of the schema mapping in Figure 5.5 is shown in Figure 5.6. These two schema mappings are *data exchange equivalent*. That is, for every source instance I, they have the same set of universal solutions [33]. Intuitively, some relations in the former are unreachable when chasing.

Proposition 5.13. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a weakly acyclic* EF SOTGD+EF SOTGD *schema mapping. Let $\mathcal{M}^{\text{skel}}$ be the skeleton rewriting of $\mathcal{M}$. For every UCQ q over $\mathbf{T}$ and for every $\mathbf{S}$-instance I, we have* $\text{XR-certain}(q, I, M) = \text{XR-certain}(q^{\text{skel}}, I, M')$.

Hint. First, we will show that there exists a solution J for I with respect to $\mathcal{M}$ if and only if there exists a solution J' for I with respect to $\mathcal{M}^{\text{skel}}$. The construction of J from J' is the inverse of the construction of J' from J, and we will show that for each such pair J and J', the answers to target queries on J are equal to the answers to their skeleton rewritings on J'.

For the forward direction ($\Rightarrow$), let J be a solution for I w.r.t. $\mathcal{M}$. Then for every tuple $R(v_1, \ldots, v_k)$ in J, there is a tuple of the form $R_{s_1,\ldots,s_k}(c_1, \ldots, c_l)$ in J', where $s_1, \ldots, s_k$ are the skeletons of the terms denoted by the values $v_1, \ldots, v_k$),

$$
\begin{array}{lcl}
\mathbf{S} & = & \{\mathrm{R}\} \\
\mathbf{T} & = & \left\{\mathrm{T}_{\bullet,\bullet}, \mathrm{T}_{\bullet,f(\bullet,\bullet)}, \mathrm{T}_{f(\bullet,\bullet),\bullet}, \mathrm{T}_{f(\bullet,\bullet),f(\bullet,\bullet)}, \mathrm{Eq}_{\bullet,\bullet}, \mathrm{Eq}_{\bullet,f(\bullet,\bullet)}, \mathrm{Eq}_{f(\bullet,\bullet),\bullet}, \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}\right\} \\
\Sigma_{\mathrm{st}} & = & \left\{ \begin{array}{lcl} \mathrm{R}(x,y) & \rightarrow & \mathrm{T}_{\bullet,f(\bullet,\bullet)}(x,x,y) \\ \mathrm{R}(x,y) & \rightarrow & \mathrm{T}_{\bullet,f(\bullet,\bullet)}(y,x,y) \end{array} \right\} \\
\Sigma_{\mathrm{t}}^{\mathrm{Eq}} & = & \left\{ \begin{array}{l}
\mathrm{T}_{\bullet,\bullet}(x,y) \wedge \mathrm{Eq}_{\bullet,\bullet}(x,x') \wedge \mathrm{T}_{\bullet,\bullet}(x',y') \rightarrow \mathrm{Eq}_{\bullet,\bullet}(y,y') \\
\mathrm{T}_{\bullet,\bullet}(x,y) \wedge \mathrm{Eq}_{\bullet,\bullet}(x,x') \wedge \mathrm{T}_{\bullet,f(\bullet,\bullet)}(x',y'_1,y'_2) \rightarrow \mathrm{Eq}_{\bullet,f(\bullet,\bullet)}(y,y'_1,y'_2) \\
\mathrm{T}_{\bullet,\bullet}(x,y) \wedge \mathrm{Eq}_{\bullet,f(\bullet,\bullet)}(x,x'_1,x'_2) \wedge \mathrm{T}_{f(\bullet,\bullet),\bullet}(x'_1,x'_2,y') \rightarrow \mathrm{Eq}_{\bullet,\bullet}(y,y') \\
\mathrm{T}_{\bullet,\bullet}(x,y) \wedge \mathrm{Eq}_{\bullet,f(\bullet,\bullet)}(x,x'_1,x'_2) \wedge \mathrm{T}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x'_1,x'_2,y'_1,y'_2) \rightarrow \mathrm{Eq}_{\bullet,f(\bullet,\bullet)}(y,y'_1,y'_2) \\
\mathrm{T}_{\bullet,f(\bullet,\bullet)}(x,y_1,y_2) \wedge \mathrm{Eq}_{\bullet,\bullet}(x,x') \wedge \mathrm{T}_{\bullet,\bullet}(x',y') \rightarrow \mathrm{Eq}_{f(\bullet,\bullet),\bullet}(y_1,y_2,y') \\
\mathrm{T}_{\bullet,f(\bullet,\bullet)}(x,y_1,y_2) \wedge \mathrm{Eq}_{\bullet,\bullet}(x,x') \wedge \mathrm{T}_{\bullet,f(\bullet,\bullet)}(x',y'_1,y'_2) \rightarrow \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(y_1,y_2,y'_1,y'_2) \\
\mathrm{T}_{\bullet,f(\bullet,\bullet)}(x,y_1,y_2) \wedge \mathrm{Eq}_{\bullet,f(\bullet,\bullet)}(x,x'_1,x'_2) \wedge \mathrm{T}_{f(\bullet,\bullet),\bullet}(x'_1,x'_2,y') \rightarrow \mathrm{Eq}_{f(\bullet,\bullet),\bullet}(y_1,y_2,y') \\
\mathrm{T}_{\bullet,f(\bullet,\bullet)}(x,y_1,y_2) \wedge \mathrm{Eq}_{\bullet,f(\bullet,\bullet)}(x,x'_1,x'_2) \wedge \mathrm{T}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x'_1,x'_2,y'_1,y'_2) \rightarrow \\
\qquad\qquad \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(y_1,y_2,y'_1,y'_2) \\
\mathrm{T}_{f(\bullet,\bullet),\bullet}(x_1,x_2,y) \wedge \mathrm{Eq}_{f(\bullet,\bullet),\bullet}(x_1,x_2,x') \wedge \mathrm{T}_{\bullet,\bullet}(x',y') \rightarrow \mathrm{Eq}_{\bullet,\bullet}(y,y') \\
\mathrm{T}_{f(\bullet,\bullet),\bullet}(x_1,x_2,y) \wedge \mathrm{Eq}_{f(\bullet,\bullet),\bullet}(x_1,x_2,x') \wedge \mathrm{T}_{\bullet,f(\bullet,\bullet)}(x',y'_1,y'_2) \rightarrow \mathrm{Eq}_{\bullet,f(\bullet,\bullet)}(y,y'_1,y'_2) \\
\mathrm{T}_{f(\bullet,\bullet),\bullet}(x_1,x_2,y) \wedge \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_1,x_2,x'_1,x'_2) \wedge \mathrm{T}_{f(\bullet,\bullet),\bullet}(x'_1,x'_2,y') \rightarrow \mathrm{Eq}_{\bullet,\bullet}(y,y') \\
\mathrm{T}_{f(\bullet,\bullet),\bullet}(x_1,x_2,y) \wedge \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_1,x_2,x'_1,x'_2) \wedge \mathrm{T}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x'_1,x'_2,y'_1,y'_2) \rightarrow \\
\qquad\qquad \mathrm{Eq}_{\bullet,f(\bullet,\bullet)}(y,y'_1,y'_2) \\
\mathrm{T}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_1,x_2,y_1,y_2) \wedge \mathrm{Eq}_{f(\bullet,\bullet),\bullet}(x_1,x_2,x') \wedge \mathrm{T}_{\bullet,\bullet}(x',y') \rightarrow \mathrm{Eq}_{f(\bullet,\bullet),\bullet}(y_1,y_2,y') \\
\mathrm{T}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_1,x_2,y_1,y_2) \wedge \mathrm{Eq}_{f(\bullet,\bullet),\bullet}(x_1,x_2,x') \wedge \mathrm{T}_{\bullet,f(\bullet,\bullet)}(x',y'_1,y'_2) \rightarrow \\
\qquad\qquad \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(y_1,y_2,y'_1,y'_2) \\
\mathrm{T}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_1,x_2,y_1,y_2) \wedge \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_1,x_2,x'_1,x'_2) \wedge \mathrm{T}_{f(\bullet,\bullet),\bullet}(x'_1,x'_2,y') \rightarrow \\
\qquad\qquad \mathrm{Eq}_{f(\bullet,\bullet),\bullet}(y_1,y_2,y') \\
\mathrm{T}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_1,x_2,y_1,y_2) \wedge \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_1,x_2,x'_1,x'_2) \wedge \mathrm{T}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x'_1,x'_2,y'_1,y'_2) \rightarrow \\
\qquad\qquad \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(y_1,y_2,y'_1,y'_2) \\
\mathrm{T}_{\bullet,\bullet}(x,y) \rightarrow \mathrm{Eq}_{\bullet,\bullet}(x,x) \\
\mathrm{T}_{\bullet,\bullet}(x,y) \rightarrow \mathrm{Eq}_{\bullet,\bullet}(y,y) \\
\mathrm{T}_{\bullet,f(\bullet,\bullet)}(x,y_1,y_2) \rightarrow \mathrm{Eq}_{\bullet,\bullet}(x,x) \\
\mathrm{T}_{\bullet,f(\bullet,\bullet)}(x,y_1,y_2) \rightarrow \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(y_1,y_2,y_1,y_2) \\
\mathrm{T}_{f(\bullet,\bullet),\bullet}(x_1,x_2,y) \rightarrow \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_1,x_2,x_1,x_2) \\
\mathrm{T}_{f(\bullet,\bullet),\bullet}(x_1,x_2,y) \rightarrow \mathrm{Eq}_{\bullet,\bullet}(y,y) \\
\mathrm{T}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_1,x_2,y_1,y_2) \rightarrow \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_1,x_2,x_1,x_2) \\
\mathrm{T}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_1,x_2,y_1,y_2) \rightarrow \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(y_1,y_2,y_1,y_2) \\
\mathrm{Eq}_{\bullet,\bullet}(x_1,x_2) \rightarrow \mathrm{Eq}_{\bullet,\bullet}(x_2,x_1) \\
\mathrm{Eq}_{\bullet,f(\bullet,\bullet)}(x_1,x_{2_1},x_{2_2}) \rightarrow \mathrm{Eq}_{f(\bullet,\bullet),\bullet}(x_{2_1},x_{2_2},x_1) \\
\mathrm{Eq}_{f(\bullet,\bullet),\bullet}(x_{1_1},x_{1_2},x_2) \rightarrow \mathrm{Eq}_{\bullet,f(\bullet,\bullet)}(x_2,x_{1_1},x_{1_2}) \\
\mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_{1_1},x_{1_2},x_{2_1},x_{2_2}) \rightarrow \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_{2_1},x_{2_2},x_{1_1},x_{1_2}) \\
\mathrm{Eq}_{\bullet,\bullet}(x_1,x_2) \wedge \mathrm{Eq}_{\bullet,\bullet}(x_2,x_3) \rightarrow \mathrm{Eq}_{\bullet,\bullet}(x_1,x_3) \\
\mathrm{Eq}_{\bullet,\bullet}(x_1,x_2) \wedge \mathrm{Eq}_{\bullet,f(\bullet,\bullet)}(x_2,x_{3_1},x_{3_2}) \rightarrow \mathrm{Eq}_{\bullet,f(\bullet,\bullet)}(x_1,x_{3_1},x_{3_2}) \\
\mathrm{Eq}_{\bullet,f(\bullet,\bullet)}(x_1,x_{2_1},x_{2_2}) \wedge \mathrm{Eq}_{f(\bullet,\bullet),\bullet}(x_{2_1},x_{2_2},x_3) \rightarrow \mathrm{Eq}_{\bullet,\bullet}(x_1,x_3) \\
\mathrm{Eq}_{\bullet,f(\bullet,\bullet)}(x_1,x_{2_1},x_{2_2}) \wedge \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_{2_1},x_{2_2},x_{3_1},x_{3_2}) \rightarrow \mathrm{Eq}_{\bullet,f(\bullet,\bullet)}(x_1,x_{3_1},x_{3_2}) \\
\mathrm{Eq}_{f(\bullet,\bullet),\bullet}(x_{1_1},x_{1_2},x_2) \wedge \mathrm{Eq}_{\bullet,\bullet}(x_2,x_3) \rightarrow \mathrm{Eq}_{f(\bullet,\bullet),\bullet}(x_{1_1},x_{1_2},x_3) \\
\mathrm{Eq}_{f(\bullet,\bullet),\bullet}(x_{1_1},x_{1_2},x_2) \wedge \mathrm{Eq}_{\bullet,f(\bullet,\bullet)}(x_2,x_{3_1},x_{3_2}) \rightarrow \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_{1_1},x_{1_2},x_{3_1},x_{3_2}) \\
\mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_{1_1},x_{1_2},x_{2_1},x_{2_2}) \wedge \mathrm{Eq}_{f(\bullet,\bullet),\bullet}(x_{2_1},x_{2_2},x_3) \rightarrow \mathrm{Eq}_{f(\bullet,\bullet),\bullet}(x_{1_1},x_{1_2},x_3) \\
\mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_{1_1},x_{1_2},x_{2_1},x_{2_2}) \wedge \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_{2_1},x_{2_2},x_{3_1},x_{3_2}) \rightarrow \\
\qquad\qquad \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_{1_1},x_{1_2},x_{3_1},x_{3_2})
\end{array} \right\} \\
\mathrm{reachable}^{\mathrm{Eq}}(x,y) & := & \exists c,c'(\mathrm{T}_{\bullet,\bullet}(x,c) \wedge \mathrm{Eq}_{\bullet,\bullet}(c,c') \wedge \mathrm{T}_{\bullet,\bullet}(y,c')) \\
& \cup & \exists c,c'_1,c'_2(\mathrm{T}_{\bullet,\bullet}(x,c) \wedge \mathrm{Eq}_{\bullet,f(\bullet,\bullet)}(c,c'_1,c'_2) \wedge \mathrm{T}_{\bullet,f(\bullet,\bullet)}(y,c'_1,c'_2)) \\
& \cup & \exists c_1,c_2,c'(\mathrm{T}_{\bullet,f(\bullet,\bullet)}(x,c_1,c_2) \wedge \mathrm{Eq}_{f(\bullet,\bullet),\bullet}(c_1,c_2,c') \wedge \mathrm{T}_{\bullet,\bullet}(y,c')) \\
& \cup & \exists c_1,c_2,c'_1,c'_2(\mathrm{T}_{\bullet,f(\bullet,\bullet)}(x,c_1,c_2) \wedge \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(c_1,c_2,c'_1,c'_2) \wedge \mathrm{T}_{\bullet,f(\bullet,\bullet)}(y,c'_1,c'_2))
\end{array}
$$

Figure 5.5: Undirected reachability, skolemized, equality singularized, and skeleton rewritten.

and $c_1, \ldots, c_l$ are the constants appearing in those terms. Consider a tgd σ in $\mathcal{M}$. Since σ is satisfied by (I, J), the corresponding atoms in (I, J') satisfy the skeleton rewritings of σ in $\mathcal{M}'$. Additionally, for every boolean UCQ q on $\mathbf{T}$ that is satisfied by J, the atoms in J' corresponding to q's image satisfy a clause of the

$$
\begin{array}{lcl}
\mathbf{S} & = & \{\mathrm{R}\} \\
\mathbf{T} & = & \left\{\mathrm{T}_{\bullet,f(\bullet,\bullet)}, \mathrm{Eq}_{\bullet,\bullet}, \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}\right\} \\
\Sigma_{\mathrm{st}} & = & \left\{ \begin{array}{lcl} \mathrm{R}(x,y) & \rightarrow & \mathrm{T}_{\bullet,f(\bullet,\bullet)}(x,x,y) \\ \mathrm{R}(x,y) & \rightarrow & \mathrm{T}_{\bullet,f(\bullet,\bullet)}(y,x,y) \end{array} \right\} \\
\Sigma_{\mathrm{t}}^{\mathrm{skel}} & = & \left\{ \begin{array}{l} \mathrm{T}_{\bullet,f(\bullet,\bullet)}(x,y_1,y_2) \wedge \mathrm{Eq}_{\bullet,\bullet}(x,x') \wedge \mathrm{T}_{\bullet,f(\bullet,\bullet)}(x',y_1',y_2') \\ \qquad\qquad \rightarrow \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(y_1,y_2,y_1',y_2') \\ \mathrm{T}_{\bullet,f(\bullet,\bullet)}(x,y_1,y_2) \rightarrow \mathrm{Eq}_{\bullet,\bullet}(x,x) \\ \mathrm{T}_{\bullet,f(\bullet,\bullet)}(x,y_1,y_2) \rightarrow \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(y_1,y_2,y_1,y_2) \\ \mathrm{Eq}_{\bullet,\bullet}(x_1,x_2) \rightarrow \mathrm{Eq}_{\bullet,\bullet}(x_2,x_1) \\ \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_{1_1},x_{1_2},x_{2_1},x_{2_2}) \rightarrow \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_{2_1},x_{2_2},x_{1_1},x_{1_2}) \\ \mathrm{Eq}_{\bullet,\bullet}(x_1,x_2) \wedge \mathrm{Eq}_{\bullet,\bullet}(x_2,x_3) \rightarrow \mathrm{Eq}_{\bullet,\bullet}(x_1,x_3) \\ \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_{1_1},x_{1_2},x_{2_1},x_{2_2}) \wedge \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_{2_1},x_{2_2},x_{3_1},x_{3_2}) \\ \qquad\qquad \rightarrow \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(x_{1_1},x_{1_2},x_{3_1},x_{3_2}) \end{array} \right\} \\
\mathrm{reachable}^{\mathrm{skel}}(x,y) & := & (\exists c_1,c_2,c_1',c_2') \\
 & & \mathrm{T}_{\bullet,f(\bullet,\bullet)}(x,c_1,c_2) \\
 & & \wedge \mathrm{Eq}_{f(\bullet,\bullet),f(\bullet,\bullet)}(c_1,c_2,c_1',c_2') \\
 & & \wedge \mathrm{T}_{\bullet,f(\bullet,\bullet)}(y,c_1',c_2')
\end{array}
$$

Figure 5.6: Example schema mapping from Figure 5.1, skolemized, equality singularized, skeleton rewritten, and optimized.

skeleton rewriting of q.

For the reverse direction ($\Leftarrow$), let J' be a solution for I w.r.t. $\mathcal{M}$. Then for every tuple $R_{s_1,\ldots,s_k}(c_1,\ldots,c_l)$ in J', there is a tuple of the form $R(v_1,\ldots,v_k)$ where $v_1,\ldots,v_k$ are the values of the terms obtained by applying the skeletons $s_1,\ldots,s_k$ to the constants $c_1,\ldots,c_l$. Consider a tgd σ in $\mathcal{M}$. Since every skeleton rewriting of σ is satisfied by (I,J'), the corresponding atoms in (I,J) satisfy σ. Additionally, for every boolean UCQ q on $\mathbf{T}$ and every clause q' of its rewriting, if q' is satisfied by J', then the atoms in J corresponding to q''s image satisfy q. □

5.1.4 Proof of Theorem 5.5

We finally can prove Theorem 5.5 by combining the above results: Let $\mathcal{M} = (\mathbf{S},\mathbf{T},\Sigma_{\mathrm{st}},\Sigma_{\mathrm{t}})$ be a weakly acyclic SOTGD+(SOTGD, EGD) schema mapping, and let $\hat{\mathcal{M}}$ be the skeleton rewriting of the equality singularization of $\mathcal{M}$, extended

with the egd

$$\mathrm{Eq}_{\bullet,\bullet}(x, y) \rightarrow x = y \ .$$

Furthermore, for any UCQ q over $\mathbf{T}$, let $\hat{q}$ be the skeleton rewriting of the equality simulation of q. Then we claim that XR-certain$(q, I, M) =$ XR-certain$(\hat{q}, I, \hat{\mathcal{M}})$.

It suffices to show that, for all source instances I,

1. I has a solution with respect to $\hat{\mathcal{M}}$ if and only if I has a solution with respect to $\mathcal{M}$.

2. if I has a solution with respect to $\mathcal{M}$, then, for all UCQs q over $\mathbf{T}$, certain$(\hat{q}, I, \hat{\mathcal{M}}) =$ certain$(q, I, \mathcal{M})$

The first item follows from Proposition 5.8(a) and Proposition 5.13. The second item follows from Proposition 5.8(b) and Proposition 5.13.

5.2 Discussion

In this chapter, we have given a series of schema mapping and query rewritings that, together, constitute a technique for reducing XR-certain query answering w.r.t. weakly acyclic GLAV+(GLAV, EGD) schema mappings to XR-certain query answering w.r.t. GAV+(GAV, EGD) schema mappings. In fact, we prove the stronger result that XR-certain query answering w.r.t. weakly acyclic SOTGD+(SOTGD, EGD) schema mappings can be reduced to XR-certain query answering w.r.t. GAV+(GAV, EGD) schema mappings (Theorem 5.1).

Theorem 5.5 allows us to extend the results from Chapter 4 to weakly acyclic GLAV+(GLAV, EGD) schema mappings. The proof above is based on a method for eliminating the existentially quantified variables. Other methods for eliminating existential quantifiers from tgds have been previously studied in the literature.

An early example is Duschka and Genesereth's inverse rules algorithm [27] for acyclic LAV rules, which inspired our approach. Krotzsch and Rudolph describe an existentially quantified variable elimination procedure for schema mappings composed of GLAV constraints and relational denial constraints (a subset of denial constraints with no equality or inequality atoms) that are jointly-acyclic (a relaxation of weak acyclicity) in [50]. Their approach is similar to ours in that it creates extra attributes to represent skolem terms in place of existentially quantified variables, but our constraint language includes the additional expressiveness of egds, whose careful handling is a primary concern of our approach. Marnette studied termination of the chase for schema mappings with target constraints in [60], where he introduced the oblivious skolem chase, a modification of the chase procedure in which skolem terms are allowed to appear in instances. A similar procedure was used to prove the correctness of a limited form of skeleton rewriting in [19].

Equality singularization for tgds was introduced in [60], where it was referred to simply as "singularization". In [19], a different equality simulation technique was used, based on substitution. In that presentation, the simulation was woven into the skeleton rewriting step.

Theorem 5.5 is related to a result in an unpublished manuscript [61], which can be stated as follows: given any weakly acyclic GLAV+GLAV schema mapping $\mathcal{M}$ and every conjunctive query q, one can compute a Datalog program that, given any source instance as input, computes the certain answers of q with respect to $\mathcal{M}$. Note that, conceptually, a Datalog program can be viewed as a GAV+GAV schema mapping where the source schema consists of the EDB predicates and the target schema consists of the IDB predicates.

In the next chapter, we will make use of the reductions given here as

part of an implementation of the algorithms of Chapter 4 for weakly acyclic GLAV+(GLAV, EGD) schema mappings.

Chapter 6

Empirical Evaluation of Algorithms

In the previous two chapters, we introduced algorithms for computing XR-certain answers for a broad class of schema mappings. Our techniques include reduction to CQA and a simple reduction to DLP. In the upcoming sections, we will introduce an implementation of the simple reduction to DLP. We will not explore an impementation of reduction to CQA (using GAV unfolding) because, notably, CQA solvers that support egds in full generality are themselves (to date) based on reductions to DLP. We consider the simple reduction to DLP to be a reference implementation: the declarative specifications of schema mapping and query are simply transformed into another declarative specification, albeit in a language with broader applicability.

In the sections below, we introduce a novel compelling data exchange/integration benchmark (Section 6.1), explore the performance of the reference implementation (Section 6.2), and finally describe and evaluate a carefully crafted optimized implementation (Section 6.3). We discuss techniques for computing XR-certain answers in practice despite its coNP-complete data complexity. The

optimized implementation is quite specialized to the problem of data exchange; it utilizes the notion of repair envelopes introduced by Eiter *et al.* [29], along with new notions that encapsulate the effects of individual constraint violations in order to enable computation of XR-certain answers using small *focus areas* of source instances. These techniques are both sound and complete: they do not compromise correctness for efficiency. However, assuming P $\neq$ NP, it is not possible to bound the size of the focus areas for a fixed schema mapping $\mathcal{M}$ unless XR-certain query answering for $\mathcal{M}$ is in PTIME. In other words, these techniques to do not universally yield benefits.

6.1 Benchmark: Genome Browser

Scientific data is a potentially important application of XR-certain query answering because research decisions and discoveries are often made based on data drawn from a variety of sources. We believe that data sets from computational sciences, such as computational genomics, are particularly in need of concepts and techniques that, like XR-certain answers, eliminate the unquantifiable uncertainty that arises from constraint violations. The guarantee given by XR-certain query answers – that all possible repairs of the source instance agree – is a natural fit for these circumstances.

Inspired by the UCSC Genome Browser, we present a benchmark that uses real data: a loose simulation of the genome browser data import process. The UCSC Genome Browser database is constructed using a variety of algorithms and public data sources. Our benchmark focuses on the human gene model, a set of genomic sequences which putatively capture the portion of the human genome that encodes proteins. The UCSC Genome Browser algorithms compute these sequences from a reference genome by computing alignments for known/observed

proteins and transcripts from the UniProt and GenBank databases [44]. For our purposes, we treat the set of transcripts as given (that is, as a source instance rather than the result of a computation), and we provide a schema mapping mimicking how this data, plus a significant volume of data from the RefSeq, Entrez Gene, and UniProt databases, are combined and transformed into the UCSC Genome Browser database. Our schema mapping makes several loose approximations of scientific reality which serve to maximize the portion of the genome browser schema that we populate, but which also introduce some inconsistency to the data. It is for this reason that we say our schema mapping merely *mimicks* the true UCSC Genome Browser data import process, even though our target schema is faithful to the real database.

The RefSeq and EntrezGene databases are not available for download in a relational format. The RefSeq database is offered in a text file format, within which the data are arranged in a nested fashion with transcripts as the top-level elements. Every transcript has associated source and reference information (documenting how and by whom it was observed), and each may have subsections specifying what protein it encodes and what gene it is transcribed from. These subsections often link to other databases using external protein and gene identifiers. Our ETL step places transcripts, sources, references, genes, and proteins into five separate respective relations, all keyed by transcript accession identifier. The EntrezGene database is available in ASN.1 format, which we first convert to xml using the `gene2xml` tool from the NCBI ToolBox [65]. From the resulting xml, we extract the desired fields into a single table using the `xtract` tool from NCBI Entrez Direct [66].

Our complete schema mapping is available for download at `https://users.soe.ucsc.edu/~rhalpert/xr-benchmarks/`. Table 6.1 summarizes the data

Table 6.1: Source Instances

Database	Relations	total # of Attributes	total # of Tuples
UCSC*	2	13	165,920
RefSeq	5	38	706,923
EntrezGene	1	3	431,114
UniProt	1	3	4,405,573

*Transcript alignments and crossreference only.

sources.

We represent the given part of UCSC's gene model with two tables, ComputedAlignments, which holds data about the transcripts themselves, and ComputedCrossref, which holds a cross-reference between UCSC "known gene IDs" (referred to here as "transcripts") and the closest corresponding external database transcript identifier (usually a RefSeq accession) and protein identifier (usually a UniProt or RefSeq accession). Our hand-written schema mapping specifies how these tables and the RefSeq, EntrezGene, and UniProt databases are used to populate the target schema. It also applies a key constraint to each target relation, per industry best-practice. Many of the key constraints are specified by the Genome Browser's schema, while some are not specified but are reasonable constraints that are in fact satisfied by the Genome Browser data.

The true Genome Browser's process computes a single coherent truth that may differ from the truth(s) represented in the external databases. Our schema mapping, on the other hand, consolidates these sources into the target schema, and this gives rise to some inconsistency. The key constraints on the knownGene and kgXref tables prove critical in this regard: they enforce that each transcript has exactly one value for the exon count, and one gene symbol, respectively. Since values from both the UCSC gene model and from the other sources are used to populate the relevant attributes, this effectively gives rise to constraint violations

when the UCSC gene model disagrees with RefSeq on the number of exons, and when RefSeq and EntrezGene collectively list more than one gene symbol. These two circumstances are expected to arise a small fraction of the time. The relevant parts of the schema mapping are depicted in Figure 6.1.

The knownIsoforms relation groups transcripts into clusters, where each cluster represents a gene. The Genome Browser computes this relation based on genomic coordinates [44]. Our schema mapping populates the knownIsoforms relation using a naïve simplification of this approach: transcripts that share either an Entrez Gene ID or a gene symbol are made to reside in the same cluster. These two approaches are incomparable. Ours relies on existing gene symbol annotations from Entrez and UniProt, as well as crossreference annotations on UCSC transcripts, all of which have different levels of rigor and completeness. The knownIsoforms table is thus included in our schema mapping primarily to exercise the interaction of existentially quantified values with egds, which is a differentiating feature of weakly acyclic schema mappings versus other types of syntactic restrictions of GLAV+(GLAV, EGD) schema mappings (e.g., GAV+(GAV, EGD) or separable [16] schema mappings). This part of the schema mapping is depicted in Figure 6.1.

6.1.1 Benchmark Data and Queries

We wish to test the scalability of our implementation relative to two factors: the instance size and the proportion of the source tuples involved in egd violations (called "suspect" tuples, which we make precise with the notion of a *source repair envelope* defined in Section 6.3.2). To this end, we define a set of instances having specified sizes and ratios of suspect to total source tuples. In order to construct our test instances, we chase the unmodified source instances with the schema mapping and compute which source tuples are suspect. We call this the *raw*

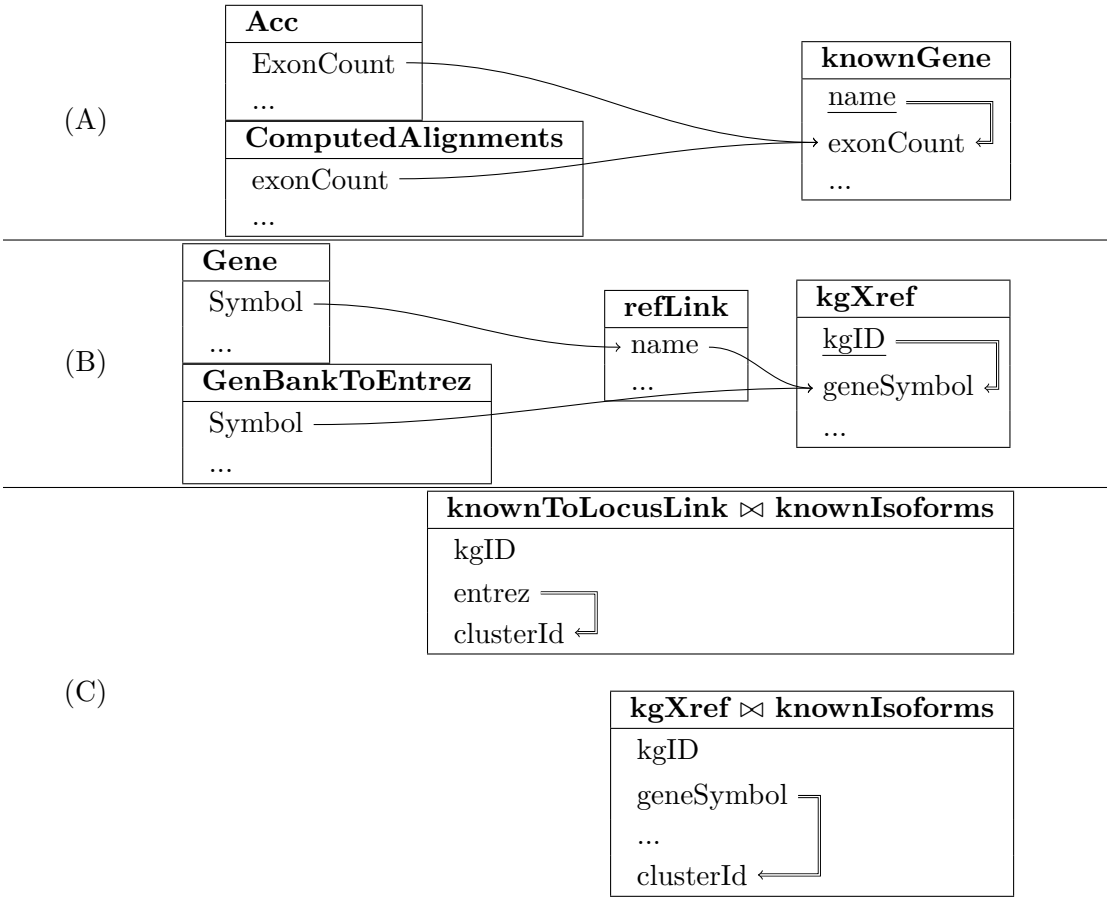

Figure 6.1: Critical parts of the schema mapping. Single arrows represent value propagation via tgds, and double arrows represent functional dependencies (egds). Keys (also egds) are underlined. (A) Competing values for the exon count. (B) Competing values for the gene symbol. (C) Clustering of transcripts according to Entrez Gene ID and gene symbol. The indicated egds give rise to equalities between nulls.

result. Then, each test instance is produced by selecting a randomized subset of the source instances with the desired size and ratio of suspect to total tuples in one particular table (ComputedCrossref), which we use as a rough proxy for the ratio for the entire source. Thus, for the largest size ("full"), the maximum ratio is the ratio found in the raw result (2.9%), and there is only one such instance. For the smaller instances, we are able to select enough suspect tuples to produce larger ratios. We use a randomized selection procedure to materialize a set of instances for each profile; six (at 3% suspect) for small and medium instances, three per ratio for large instances, and, necessarily, just one full instance (at 2.9% suspect). The characteristics of these instances are given in Table 6.2.

Table 6.2: Test instances have sizes small (S), medium (M), large (L), and full (F), and 0, 3, 9, or 20 percent of their transcripts suspect.

instance:	L0	L3	L9	L20
source tuples	321k	322k	316k	301k
total tuples	716k	724k	731k	748k
suspect transcripts	0%	3%	9%	20%
suspect tuples*	0%	2.0%	5.8%	13.4%

instance:	S3	M3	L3	F3
source tuples	3.5k	36k	322k	1,846k
total tuples	7.9k	77k	724k	5,354k
suspect transcripts	3%	3%	3%	2.9%
suspect tuples*	1.8%	1.8%	2.0%	3.4%

*includes source and target

Table 6.3 lists our query suite. Queries labeled "ep$N$" are adapted from the EQUIP query suite [47]: five of the 21 queries given there are applicable to our target schema. Additional queries labeled "xr$N$" are new queries created to exercise the critical parts of the schema mapping, including what is XR-certain knowledge in the knownGene relation, and what pairs of transcripts reside in the same cluster in the knownIsoforms relation. In our experiments, we run the queries sequentially.

Table 6.3: Query Suite, with approximate answer counts for the large-size instances.

Query	Answers
ep1() :- refLink(symbol, _, acc, protacc, _, _, _, _), kgXref(ucscid, _, spid, _, symbol, _, _, _, _, _)	1*
ep2(protacc) :- refLink(symbol, _, acc, protacc, _, _, _, _), kgXref(ucscid, _, spid, _, symbol, _, _, _, _, _)	6,000
ep3(protacc,spid) :- refLink(symbol, _, acc, protacc, _, _, _, _), kgXref(ucscid, _, spid, _, symbol, _, _, _, _, _)	12,000
ep15(symbol) :- kgXref(ucscid, _, _, _, symbol, refseq, _, _, _, _), refLink(_, product, refseq, _, _, _, entrez, _)	1,500
ep16(symbol,entrez) :- kgXref(ucscid, _, _, _, symbol, refseq, _, _, _, _), refLink(_, product, refseq, _, _, _, entrez, _)	1,500
xr1() :- knownGene(kgid, ch, sd, txs, txe, cs, ce, exc, exs, exe, pac, alignid)	1*
xr2(kgid) :- knownGene(kgid, ch, sd, txs, txe, cs, ce, exc, exs, exe, pac, alignid)	10,000
xr3(kgid, ch, sd, txs, txe, cs, ce, exc, exs, exe, pac, ai) :- knownGene(kgid, ch, sd, txs, txe, cs, ce, exc, exs, exe, pac, ai)	10,000**
xr4() :- knownIsoforms(cluster, transcript1), knownIsoforms(cluster, transcript2)	1*
xr5(transcript1) :- knownIsoforms(cluster, transcript1), knownIsoforms(cluster, transcript2)	10,000
xr6(transcript1, transcript2) :- knownIsoforms(cluster, transcript1), knownIsoforms(cluster, transcript2)	35,000

*boolean **projection-free

6.2 Reference Implementation and Results

We have implemented two XR-certain query answering algorithms. The first is a straightforward implementation of the reduction to disjunctive logic programmings given in Theorem 4.9. The disjunctive logic program is passed to a third-party solver. The second implementation (described later) utilizes optimizations presented in Section 6.3. The two implementations share a front-end (which parses inputs and performs schema mapping translations) and a test-harness (which automatically runs the requested implementation for various inputs, and records runtimes and answers for later analysis).

6.2.1 Implementation

Using the reduction given in Theorem 4.9, we have built a reference implementation of XR-certain query answering. Our implementation accepts the following inputs: (1) a weakly acyclic GLAV+(GLAV, EGD) schema mapping; (2) an arbitrary source instance, specified either as text or as a JDBC connection string; (3) a union of conjunctive queries over the target schema. The (text) inputs are formatted in a datalog-like language that supports tgds, egds, and unions of conjunctive queries, including named variables, unnamed variables, and string and numeric constants. Existentially quantified variables are determined implicitly (those that appear in tgd heads without definition in the body).

The implementation begins by checking that the input schema mapping is indeed weakly acyclic, and then by transforming it into a GAV+(GAV, EGD) schema mapping using an optimized version of the reduction in Theorem 5.1. The query is simultaneously transformed into a new union of conjunctive queries as in the same reduction. Finally, a disjunctive logic program is written out to a file and run using clingo 4.4.0, a solver from the Potsdam Answer Set Solving Collection[37]

(Potassco). All computational steps other than disjunctive logic solving are implemented using Java 1.7.0_80.

6.2.2 Results

All experiments were run on an 8-core Intel Core i7-2720QM CPU @2.20GHz with 16GB RAM, running Ubuntu 14.04 LTS (Linux 3.13.0 SMP x86_64). The plots of Figure 6.2 depict the runtime for these programs versus the percentage of suspect tuples and versus the instance size, respectively, with a line for each query. The latter is a log-log plot, since each instance size is an order of magnitude larger than the last.

These results illustrate a significant problem with this monolithic logic program approach: the cost of transforming the data from the source schema into the target schema is embedded in the execution cost of each individual query, which causes large instances to become unworkable even for simple queries. Additionally, the rapid increase in query runtimes as instance size increases, even for queries whose answers should be easy to compute, suggest that this implementation fails to take advantage of some exploitable structure in the instance and schema mapping. This is perhaps a symptom of the fact that the disjunctive logic program's rules are no simpler for areas of the source and target instances that are unaffected by egd violations than for those that are affected. In the following section, we will develop techniques to identify and exploit such structure by grounding the egds.

Additionally, the approach is rather opaque: it is particularly difficult to explore the intermediate states that a DLP solver visits on its way to a solution, especially for large programs. This made it hard to discover flaws in early versions of our schema mapping. However, we would be remiss to not note that debugging schema mappings is itself a challenging problem worthy of study [20], and

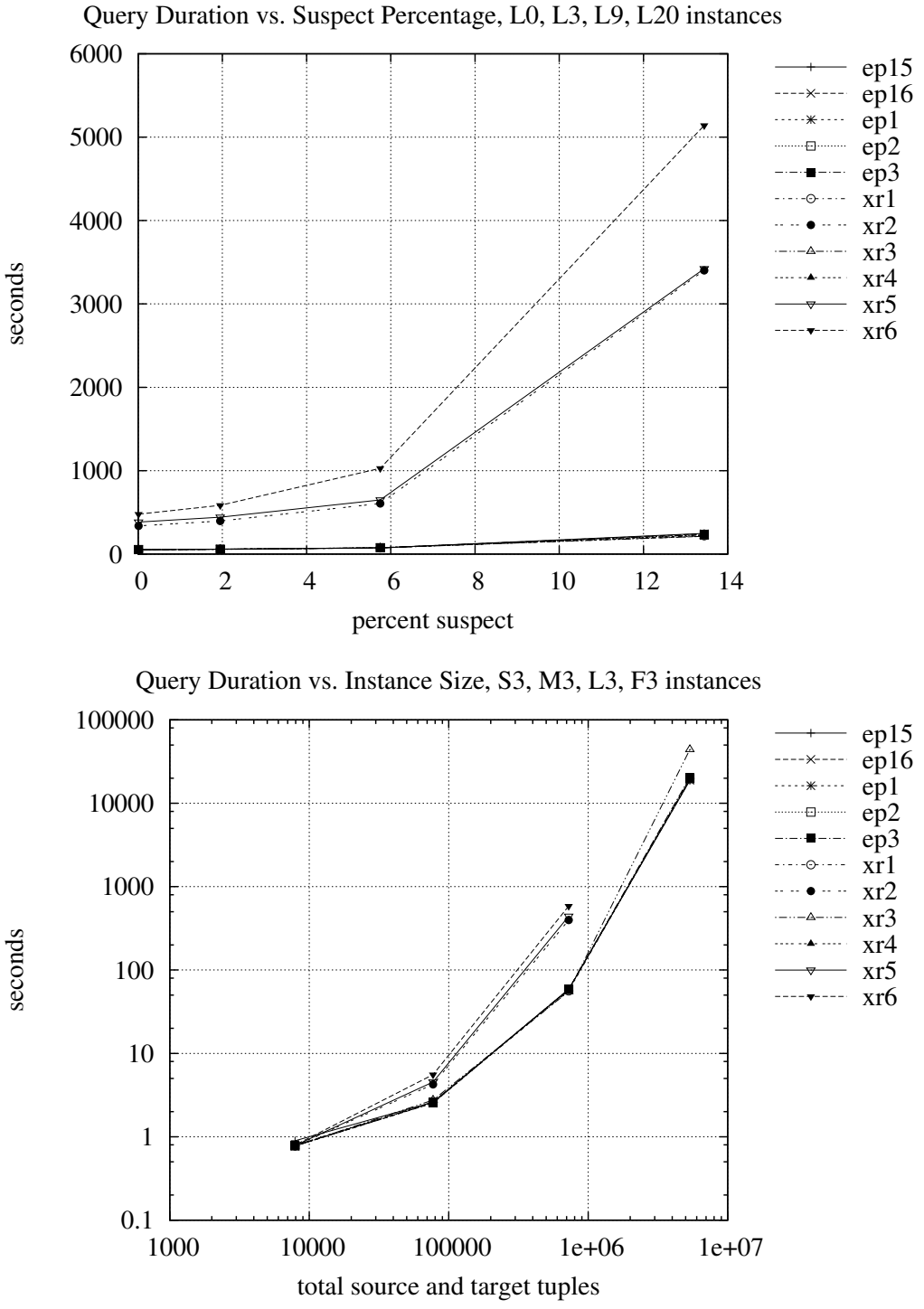

Figure 6.2: Performance of XR-certain query answering using clingo.

furthermore that some tools do exist for debugging ASP programs [35].

6.3 Practical Optimizations for Implementation of XR-certain Semantics

The approaches to XR-certain query answering in the previous sections produce one monolithic instance of a hard problem, to hand-off to a solver. In this section, we present techniques to exploit structure inherent in many data exchange settings in order to enable XR-certain query answering by solving a collection of small instances of hard problems instead. We will see that this can yield a big practical benefit in some cases.

We draw on techniques described in the literature for query answering over inconsistent databases, specifically, the notion of *repair envelopes* introduced by Eiter *et al.* [29]. We split query answering into two phases. The first, the *exchange phase*, is a tractable-time query-independent preprocessing step, which enables the second, the *query phase*, in which XR-certain answers to a particular query are computed by solving a collection of small disjunctive logic programs. Although the problem at hand is coNP-complete, this new approach allows us to answer queries by solving many small hard problems rather than one large one, which can give a significant performance boost in practical settings. We will see in Section 6.3.6 how an implementation of this approach performs.

6.3.1 Candidate Answers

The following definition of *candidate answers* gives an easy-to-compute complete approximation of the set of XR-Certain answers of a query.

Definition 6.1. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a GAV+(GAV, EGD) schema mapping.

1. We denote by $\mathcal{M}^{tgd}$ the schema mapping $(\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}}^{tgd})$ where Σ_{t}^{tgd} consists of the tgds from Σ_{t}. That is, all egds are dropped.

2. The *canonical quasi-solution* of a source instance I w.r.t. $\mathcal{M}$ is the canonical universal solution of I w.r.t. $\mathcal{M}^{tgd}$.

3. For every UCQ q over $\mathbf{T}$, the *candidate answers* to q w.r.t. I and $\mathcal{M}$ are $q(J)$, where J is the canonical quasi-solution of I w.r.t. $\mathcal{M}$.

Proposition 6.2. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a* GAV+(GAV, EGD) *schema mapping. Let I be an $\mathbf{S}$-instance. Let J be the canonical quasi-solution of I w.r.t. $\mathcal{M}$.*

1. *For all canonical* XR-solutions (I', J') *for I w.r.t. $\mathcal{M}$, it holds that $J' \subseteq J$.*

2. *For every UCQ q, we have that* XR-certain$(q, I, \mathcal{M}) \subseteq q(J)$. *That is, every XR-Certain answer is indeed a candidate answer.*

Proof. Let (I', J') be a canonical XR-solution for I w.r.t. $\mathcal{M}$. Since $\mathcal{M}$ is GAV+(GAV, EGD) and J' is the canonical universal solution for I' w.r.t. $\mathcal{M}$, J' is simply the closure of I' w.r.t. the tgds of $\mathcal{M}$. Therefore J' is also the canonical universal solution for I' w.r.t. $\mathcal{M}^{tgd}$. Finally, since the chase procedure is monotone [32] for GAV+GAV schema mappings and $I' \subseteq I$, we have that $J' \subseteq J$.

The second item follows directly from the first, since UCQs are monotone queries. □

6.3.2 Source Repair Envelopes

It is often possible to exclude a large portion of the database from high-complexity computations. We will define a notion similar to a *repair envelope* from [29] but suited to the setting of data exchange.

Definition 6.3. Let $\mathcal{M}$ be a weakly acyclic GLAV+(GLAV, EGD) schema mapping and I a source instance. A subset E of I is a *source repair envelope* if $(I \setminus I') \subseteq E$ for all source repairs I'.

We will now see that we can restrict our attention within a source repair envelope when computing source repairs.

Proposition 6.4. *Let $\mathcal{M}$ be a weakly acyclic* GLAV+(GLAV, EGD) *schema mapping, I a source instance, and $E \subseteq I$ a source repair envelope for I w.r.t. $\mathcal{M}$. Then $\{I' \mid I'$ is a source repair of I w.r.t. $\mathcal{M}\} = \{E' \cup (I \setminus E) \mid E'$ is a source repair of E w.r.t. $\mathcal{M}\}$.*

Proof. Claim: Suppose I' is a source repair of I w.r.t. $\mathcal{M}$. Let $E' = I' \cap E$. Then E' is a source repair of E w.r.t. $\mathcal{M}$.
Clearly $E' \subseteq I'$, so by Lemma 3.4, every solution for I' w.r.t. $\mathcal{M}$ is also a solution for E' w.r.t. $\mathcal{M}$. Suppose towards a contradiction that there exists an E'' such that $E' \subset E'' \subseteq E$ and E'' has a solution w.r.t. $\mathcal{M}$. Since I' is a source repair, it cannot be the case that $I' \cup E''$ has a solution w.r.t. $\mathcal{M}$, but since E'' has a solution w.r.t. $\mathcal{M}$ it must be the case that E'' is contained in some source repair, which must therefore exclude some facts from $I' \setminus E$. This contradicts that E is a source repair envelope. □

Proof. Claim: $I' \cap E$ is a source repair of E.
Suppose for the sake of contradiction that $I' \cap E$ is not a source repair of E. Then either $I' \cap E$ has no solution (which cannot be the case due to monotonicity of the chase) or $I' \cap E$ is strictly contained in a source repair E'' of E. But then E'' must be contained in a source repair I'' of I. However, the definition of a source repair envelope tells us that I'' contains all of $E'' \cup (I \setminus E)$. Hence it contains I', so I' wasn't a source repair of I after all.

Claim: if E' is a source repair of E then $E' \cup (I \setminus E)$ is a source repair of I. Indeed, E' has a solution w.r.t. $\mathcal{M}$ so it must be contained in a source repair of I, and by the definition of a source repair envelope that source repair of I must contain all of $(I \setminus E)$. Therefore it contains $E' \cup (I \setminus E)$. However, it cannot be a strict superset of $E' \cup (I \setminus E)$ because then E' could be extended to a larger source repair of E. □

There are many ways to calculate a source repair envelope (e.g., I is a trivial source repair envelope). Consider the *ideal source repair envelope*, given by $I - \bigcap\{I' \mid I' \text{ is a source repair for } I \text{ w.r.t. } \mathcal{M}\}$. Equivalently, the ideal source repair envelope is the minimal source repair envelope for I w.r.t. $\mathcal{M}$. The next result tells us that computing this envelope is hard.

Theorem 6.5. *Fix a schema mapping $\mathcal{M}$. Let the* intersection of source repairs membership problem *be the following decision problem: given a source instance I, and a fact f of I, is f contained in the intersection of all source repairs (that is, is $f \in \bigcap\{I' \mid I' \text{ is a source repair for } I \text{ w.r.t. } \mathcal{M}\}$)? This problem is in* coNP*, and there is a* GAV+(GAV, EGD) *schema mapping for which this problem is* coNP*-hard.*

Proof. The following is an NP algorithm to determine if f does not belong to the intersection of source repairs: Guess a source repair I'. Check if I' is a source repair of I w.r.t. $\mathcal{M}$ (PTIME) [19]. If so, and f does not belong to I', print "yes".

The hardness result is proven by reduction from the complement of 3-colorability. Let $G = (V, E)$ be a graph, with edge set $E = \{e_1, \ldots, e_n\}$. I_G is the source instance, with active domain $V \cup \{1, \ldots, n\}$, containing, for each edge $e_i = (a, b)$ the fact $E(a, b, i, i + 1)$; for each vertex $a \in V$, the facts $C_r(a), C_g(a), C_b(a)$; and one additional fact, namely $F(n, 1)$.

$\mathcal{M}$ is the schema mapping consisting of the source-to-target tgds

- $E(x, y, u, v) \wedge C_z(x) \rightarrow E'(x, y)$ (for $z \in \{r, g, b\}$)
- $E(x, y, u, v) \wedge C_z(x) \rightarrow F'(u, v)$ (for $z \in \{r, g, b\}$)
- $P_z(x) \rightarrow P'_z(x)$ (for $z \in \{r, g, b\}$)
- $F(u, v) \rightarrow F'(u, v)$

and target constraints

- $E'(x, y) \wedge P'_z(x) \wedge P'_z(y) \wedge F'(u, v) \rightarrow u = v$ (for all $z \in \{r, g, b\}$)
- $F'(u, v) \wedge F'(v, w) \rightarrow F'(u, w)$
- $F'(u, u) \wedge F'(v, w) \rightarrow v = w$

Note that I has no solutions with respect to $\mathcal{M}$, regardless of whether G is 3-colorable. This is because a solution J of I has to contain a directed cycle of F'-edges (of length n), and F' must be transitive, which means that J would have to include facts of the form $F'(i, i)$, leading to an egd violation. Indeed, every source-repair of I must either (i) omit at least one of the E-facts, or (ii) omit all P_z-facts ($z \in \{r, g, b\}$) for some vertex or (iii) omit the $F(n, 1)$ fact. It is then not hard to see that G is 3-colorable if and only if some source repair omits $F(n, 1)$. Note that, if G is 3-colorable, then there is a source repair that retains all E-facts and that retains at least one P_z-fact for each vertex ($z \in \{r, g, b\}$). This source repair must then omit the $F(n, 1)$ fact. If, on the other hand, G is not 3-colorable, then every source repair has to satisfy (i) or (iii) and, consequently, will include $F(n, 1)$. □

The following corollary extends the above result to the target (that is, the intersection of exchange repair solutions).

Corollary 6.6. *Fix a weakly acyclic schema mapping $\mathcal{M}$. Given a source instance I and a target fact t, it is in* coNP *to determine if t belongs to the intersection of exchange repair solutions, and there is a* GAV+(GAV, EGD) *schema mapping for which this problem is* coNP*-hard.*

Proof. The following is an NP algorithm to determine if t does not belong to the intersection of exchange repair solutions: Guess a soruce repair I'. Check if I' is a source repair of I w.r.t. $\mathcal{M}$ (PTIME). If so, check if t belongs to chase(I'). If not, print "yes".

For the hardness result, the proof of the theorem is sufficient, with the observation that the fact $F(n, 1)$ is subject to a source-to-target copy constraint, and so $F'(n, 1)$ is contained in the intersection of exchange repair solutions if and only if G is 3-colorable. □

The hardness established in Theorem 6.5 shows that computing the ideal source repair envelope is not practical, given that the purpose of a source repair envelope is to help reduce the need for high-complexity computations. Our next result pertains to a source repair envelope that *can* be computed in PTIME.

First, we introduce the notion of *support sets* for a target fact. For an egd or GAV tgd σ and an instance I, we denote by $ground(\sigma, J)$ the set of all groundings of σ using values from the active domain of I (that is, quantifier-free formulas that can be obtained from σ by replacing universally quantified variables by values from the active domain of I). Note that, if J is a canonical quasi-solution for a source instance I w.r.t. a GAV+(GAV, EGD) schema mapping, then the active domain of J is already included in the active domain of I.

Definition 6.7. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a GAV+(GAV, EGD) schema mapping. Let I be an $\mathbf{S}$-instance with canonical quasi-solution J, and let $f \in J$ be a fact.

- A *support set* for f is a set of the form $\{f_1, \ldots, f_n\}$ where $(f_1 \wedge \cdots \wedge f_n \rightarrow f) \in \text{ground}(\Sigma_{\text{st}} \cup \Sigma_{\text{t}}, I)$, and $(I, J) \models f_1 \wedge \cdots \wedge f_n$. The set of all support sets of f is denoted by support_sets$(f, I, \mathcal{M})$.

- The *support closure* for a set F of facts, denoted as support$^*(F, I, \mathcal{M})$, is the smallest set containing F such that whenever $g \in \text{support}^*(F, I, \mathcal{M})$, then all facts that belong to a support set of g belong to support$^*(F, I, \mathcal{M})$ as well.

Definition 6.8. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a GAV+(GAV, EGD) schema mapping. Let J be the canonical quasi-solution of I. Let

$$\text{violations}(I, \mathcal{M}) = \left\{ f \;\middle|\; \begin{array}{l} f \text{ occurs in the body of some} \\ \sigma \in \text{ground}(\Sigma_{\text{t}}, I) \text{ with } J \not\models \sigma \end{array} \right\}$$

We say that a source fact $f \in I$ is *suspect* (w.r.t. $\mathcal{M}$) if it belongs to support*(violations$(I, \mathcal{M}), I, \mathcal{M})$, and that it is *safe* otherwise. The set of suspect facts of I is denoted by I_{suspect} and the set of safe facts of I is denoted by I_{safe}.

The violation set is, intuitively, the set of facts that are directly involved in an egd violation, while the violation closure is, intuitively, the set of facts that are, possibly indirectly, involved in an egd violation. The notation I_{safe} and I_{suspect} assumes that it is clear from the context which schema mapping is being referred to.

Proposition 6.9. *Let $\mathcal{M}$ be a* GAV+(GAV, EGD) *schema mapping and I a source instance. Then $I_{suspect}$ is a source repair envelope for I (w.r.t. $\mathcal{M}$). Moreover, $I_{suspect}$ can be computed in polynomial time (data complexity).*

To see that this proposition holds, suppose that some $f \in I$ is omitted by a source repair I' of I. Then $I' \cup \{f\}$ has no solution. It follows that f must be in

the violation closure of I w.r.t. $\mathcal{M}$ (with the steps of the chase of $I' \cup \{f\}$ serving as proof of such). Therefore, f belongs to I_{suspect}.

The following example reminds us that I_{suspect} is not necessarily a *minimal* source repair envelope.

Example 6.10. Let $I = \{P(a,b), P(a,c), Q(b,c)\}$, and let $\mathcal{M} = (\{P,Q\}, \{P',Q'\}, \{P(x,y) \to P'(x,y), Q(x,y) \to Q'(x,y)\}, \{P'(x,y) \wedge P'(x,y') \to y = y', P'(x,y) \wedge P'(x,y') \wedge Q'(y,y') \to y = y'\})$. Then

$$I_{\text{suspect}} = \{P(a,b), P(a,c), Q(b,c)\}$$

However, the key constraint on P' forces every XR-solution to have at most one of $P(a,b), P(a,c)$, so the second egd in Σ_{t} is satisfied without eliminating $Q(b,c)$. Indeed, the ideal source repair envelope for I is $\{P(a,b), P(a,c)\}$.

Nonetheless, I_{suspect} is a potentially useful source repair envelope: we vary the proportion of facts in I_{suspect} versus I_{safe} in our test instances in order to evaluate the importance of this measure.

We will now extend the notion of a source repair envelope to include target instances.

Definition 6.11 (exchange repair envelope)**.** Let $\mathcal{M}$ be a GAV+(GAV, EGD) schema mapping, and I a source instance. Let J be the canonical quasi-solution of I. Two sets $E \subseteq I$ and $F \subseteq J$ of facts, together comprise an *exchange repair envelope* (E,F) if, for all canonical XR-solutions (I', J') of I w.r.t. $\mathcal{M}$, we have that $(I \setminus I') \subseteq E$ and $(J \setminus J') \subseteq F$.

We saw in Corollary 6.6 that computing the ideal exchange repair envelope is hard. We will now see how to extend an existing source repair envelope to the target, in polynomial time.

Definition 6.12. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a GAV+(GAV, EGD) schema mapping, and I be an $\mathbf{S}$-instance. Let J be the canonical quasi-solution of I w.r.t. $\mathcal{M}$. Define the *influence* of a set of facts $E \subseteq I$, denoted influence$(E, I, \mathcal{M})$, as the smallest set containing E such that whenever $g \in$ influence$(E, I, \mathcal{M})$, every fact f that has a support set containing g also belongs to influence$(E, I, \mathcal{M})$.

Proposition 6.13. *Let $\mathcal{M}$ be a* GAV+(GAV, EGD) *schema mapping and I a source instance. Let J be the canonical quasi-solution of I w.r.t. $\mathcal{M}$. Let E be a source repair envelope for I w.r.t. $\mathcal{M}$, and let $F = \mathit{influence}(E, I, \mathcal{M})$. Then (E, F) is an exchange repair envelope for I w.r.t. $\mathcal{M}$.*

In particular, $(I_{\mathit{suspect}}, J_{\mathit{suspect}})$ is an exchange repair envelope for I, where $J_{\mathit{suspect}} = \mathit{influence}(I_{\mathit{suspect}}, I, \mathcal{M})$.

Proof. Let (I', J') be a canonical XR-solution for I w.r.t. $\mathcal{M}$. Let t be a fact in $J \setminus J'$. Since t is not in (I', J'), there must be some fact f in support$^*(\{t\}, I, \mathcal{M})$ in $I \setminus I'$. Therefore, f is contained in the source repair envelope E, so by definition we have that $t \in$ influence$(E, I, \mathcal{M})$. □

Fact 6.1. The following two statements hold, where F is an arbitrary set of target facts:

- The influence of the source restriction of the support closure of F contains the support closure of F; and
- A support closure of F and its influence are equal over their source restrictions.

In light of the above, we can refer to violation influences instead of violation closures whenever we wish to work with exchange repair envelopes rather than source repair envelopes. It is important to notice that a fact may have one support

set which places it in a violation influence, but also another support set whose facts are not contained in any violation influence. Such facts lie in the difference between the violation influence and the ideal exchange repair envelope, but are nonetheless easy to identify.

6.3.3 Violation Clusters

Violation clusters are a concept that will help us further reduce the combinatorial complexity of computing XR-certain answers.

We start with a motivating example.

Example 6.14. Let $I = \{P_1(a,b), P_1(a,c), P_2(a,b), P_2(a,c), \ldots, P_n(a,b), P_n(a,c)\}$, and let

$$\mathcal{M} = (\{P_1, \ldots, P_n\}, \{Q_1, \ldots, Q_n\}, \{P_1(x,y) \rightarrow Q_1(x,y), \ldots P_n(x,y) \rightarrow Q_n(x,y)\},$$
$$\{Q_1(x,y) \wedge Q_1(x,y') \rightarrow y = y', \ldots, Q_n(x,y) \wedge Q_n(x,y') \rightarrow y = y'\})$$

There are 2^n source repairs, which can be built by choosing one source atom from each of the n relations, in every possible combination. In this sense, the set of source repairs for this example is highly structured.

Now consider the query $q(x)$:- $\exists y Q_1(x,y)$. Every source repair of I w.r.t. $\mathcal{M}$ contains either $P_1(a,b)$ or $P_1(a,c)$, so we can conclude that $q(a) \in$ XR-certain$(q, I, \mathcal{M})$ by considering just the two possibilities for the the P_1 relation. In so doing, we ignore the other $n-1$ relations and avoid having to consider 2^n source repairs.

In this section, we will generalize the above observation and demonstrate that it can be used to reduce the size of instances and schema mappings for which we must explore all source repairs. To do so, we introduce a notion of *independence*

that captures when particular egd violations are sufficiently isolated from each other to be processed separately.

To simplify the presentation, it will be convenient to consider schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ in which Σ_{t} may contain *grounded egds* (where universally quantified variables have been replaced by constants). This will allow us to more easily describe how an instance is carved up into segments that can be processed independently. Intuitively, each grounding of an egd describes one potential violation of that egd. The notions of solutions, universal solutions, source repairs, and exchange repair solutions all apply without modification to schema mappings containing grounded egds.

As a slight abuse of notation, when D is a set of target constraints, we will write $\mathcal{M} \cup D$ to denote the schema mapping obtained by adding the constraints in D to the target constraint set of the schema mapping $\mathcal{M}$.

Definition 6.15. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a GAV+(GAV, EGD) schema mapping. Let I be an $\mathbf{S}$-instance. Let J be the canonical quasi-solution for I w.r.t. $\mathcal{M}$. Let $\sigma_1, \sigma_2 \in \text{ground}(\Sigma_{\text{t}}, I)$ be distinct grounded egds with $J \not\models \sigma_1$ and $J \not\models \sigma_2$. Let E_1 be the ideal source repair envelope for I w.r.t $(\mathcal{M}^{tgd} \cup \{\sigma_1\})$, and let E_2 be the ideal source repair envelope for I w.r.t $(\mathcal{M}^{tgd} \cup \{\sigma_2\})$. We say σ_1 and σ_2 are *pairwise-independent* if source_repairs$(I, \mathcal{M}^{tgd} \cup \{\sigma_1, \sigma_2\}) = \{(I \setminus (E_1 \cup E_2)) \cup E_1' \cup E_2' \mid E_1' \in \text{source_repairs}(I \cap E_1, \mathcal{M}^{tgd} \cup \{\sigma_1\})$ and $E_2' \in \text{source\_repairs}(I \cap E_2, \mathcal{M}^{tgd} \cup \{\sigma_2\})\}$. We say σ_1 and σ_2 are *pairwise-dependent* if they are not *pairwise-independent*.

Consider the graph of all egd violations in the quasi-solution and connect each pair of pairwise dependent egd violations by an edge. Each connected component of this graph is called a *violation cluster*. If σ_1 and σ_n reside in distinct violation clusters then we say σ_1 and σ_n are *independent*.

If a pair of violations is independent, by definition, their XR-solutions can be processed separately, then suitably recombined. Note that the above definition of violation clusters does not give us a way to identify them efficiently. The following proposition provides an efficiently computable approximation.

Proposition 6.16. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a* GAV+(GAV, EGD) *schema mapping. Let I be an $\mathbf{S}$-instance. Let J be the canonical quasi-solution I w.r.t. $\mathcal{M}$. Let σ_1, σ_2 be distinct grounded egds in ground(Σ_t, I) such that $J \not\models \sigma_1$ and $J \not\models \sigma_2$. Let E_1, E_2 be source repair envelopes for I w.r.t. $\mathcal{M} \cup \sigma_1$ and $\mathcal{M} \cup \sigma_2$, respectively. If E_1 and E_2 are disjoint, then σ_1 and σ_2 are pairwise-independent.*

Proof. Suppose $E_1 \cap E_2 = \emptyset$, and let (I', J') be a canonical XR-solution for I w.r.t. $\mathcal{M} \cup \{\sigma_1, \sigma_2\}$. Let $E_1' = E_1 \cap I'$ and let $E_2' = E_2 \cap I'$. By definition of a source repair envelope, $(I \setminus E_1) \cup E_1'$ is an XR-solution for I w.r.t. $\mathcal{M} \cup \{\sigma_1\}$, and likewise $(I \setminus E_2) \cup E_2'$ is an XR-solution for I w.r.t. $\mathcal{M} \cup \{\sigma_2\}$, and since E_1 and E_2 are disjoint, it is easy to see that E_1' is an XR-solution for E_1 w.r.t. $\mathcal{M} \cup \{\sigma_1, \sigma_2\}$ and E_2' is an XR-solution for E_2 w.r.t. $\mathcal{M} \cup \{\sigma_1, \sigma_2\}$. Finally, since GAV chase is monotone, we have that $(I \setminus E_1 \setminus E_2) \cup (E_1') \cup (E_2')$ has a solution w.r.t. $\mathcal{M} \cup \{\sigma_1, \sigma_2\}$, and there is no instance $I'' \subset I$ which strictly contains $(I \setminus E_1 \setminus E_2) \cup (E_1') \cup (E_2')$ and also has a solution w.r.t. $\mathcal{M} \cup \{\sigma_1, \sigma_2\}$. □

We can, in polynomial time, compute the support closure for each egd violation, then compute an overapproximation of the violation clusters based on the restriction to the source schema of those closures. The next proposition follows simply from the definition of a support closure, but provides some intuition and gives a shortcut for computing a source repair envelope for a violation cluster.

Proposition 6.17. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a* GAV+(GAV, EGD) *schema mapping. Let I be an $\mathbf{S}$-instance. Let J be the canonical quasi-solution for I w.r.t. $\mathcal{M}$.*

Let $\sigma_1, \ldots, \sigma_n$ be a violation cluster (so each σ_i is a ground egd where $J \not\models \sigma_i$), with support closures $E_1, \ldots, E_n$. Then $E_1 \cup \ldots \cup E_n$ is the support closure of the facts in $\sigma_1, \ldots, \sigma_n$, and its **S***-restriction is a source repair envelope for I w.r.t. $\mathcal{M}^{tgd} \cup \{\sigma_1, \ldots, \sigma_n\}$*

We have now seen how to compute a conservative approximation of the violation clusters for an instance w.r.t. a given schema mapping. Proposition 6.17 makes clear the fact that distinct violation clusters have disjoint source repair envelopes, and are therefore pairwise-independent themselves. Definition 6.15 tells us that we can thus compute the source repairs for an entire instance by computing separately the source repairs for the envelope of each violation cluster. We will now see how this supports query answering.

6.3.4 Answering Queries

We now show how to use the techniques introduced in the previous sections to compute XR-certain answers for unions of conjunctive queries. Recall that, without loss of generality, we can restrict attention to projection-free atomic queries (those that merely query membership of a particular fact in the intersection of XR-solutions for a given schema mapping and source instance). Let J be the canonical quasi-solution for an instance I w.r.t. a schema mapping $\mathcal{M}$. We will use the term *candidate facts* to refer to all of the facts in J (as opposed to the candidate **answers** for a projection-free atomic query $q(\mathbf{x})$:- $\mathrm{P}(\mathbf{x})$, which are just the P-facts in J).

Consider the definition of *support sets* in Section 6.3.2, and suppose $P(\mathbf{a})$ is a candidate fact. By definition, $P(\mathbf{a})$ is in XR-certain$(q(\mathbf{x})$:- $\mathrm{P}(\mathbf{x}), I, \mathcal{M})$ if it is contained in every exchange repair solution. Since exchange repair solutions satisfy the constraints of the schema mapping, it is easy to see that $P(\mathbf{a})$ must

have at least one support set in every XR-solution, or, equivalently, in every canonical XR-solution.

Proposition 6.17, which shows how to easily combine source repair envelopes, naturally extends to exchange repair envelopes as well. Thus we define a *violation cluster influence* as the union of the influences of the violations in a cluster, and note that this is a PTIME-computable exchange repair envelope for the set of violations in the cluster.

Example 6.18. This example illustrates that a candidate fact f may belong to the influences of multiple distinct violation clusters. Let $I = \{\mathrm{P}(a_1, a_2), \mathrm{P}(a_1, a_3), \mathrm{Q}(a_1, a_2), \mathrm{Q}(a_1, a_3)\}$, and let $\mathcal{M} = (\{\mathrm{P}, \mathrm{Q}\}, \{\mathrm{R}, \mathrm{S}, \mathrm{T}\}, \{\mathrm{P}(x, y) \rightarrow \mathrm{R}(x, y), \mathrm{Q}(x, y) \rightarrow \mathrm{S}(x, y)\}, \{\mathrm{R}(x, y) \wedge \mathrm{S}(x, z) \rightarrow \mathrm{T}(x, y, z), \mathrm{R}(x, y) \wedge \mathrm{R}(x, y') \rightarrow y = y', \mathrm{S}(x, y) \wedge \mathrm{S}(x, y') \rightarrow y = y'\})$. Then the violation cluster influence for $\{\mathrm{R}(a_1, a_2), \mathrm{R}(a_1, a_3)\}$ is

$$\left\{\begin{array}{l}\mathrm{P}(a_1, a_2), \mathrm{P}(a_1, a_3), \mathrm{R}(a_1, a_2), \mathrm{R}(a_1, a_3), \\ \mathrm{T}(a_1, a_2, a_3), \mathrm{T}(a_1, a_3, a_2), \mathrm{T}(a_1, a_2, a_2), \mathrm{T}(a_1, a_3, a_3)\end{array}\right\}$$

and the violation cluster influence for $\mathrm{S}(a_1, a_2), \mathrm{S}(a_1, a_3)$ is

$$\left\{\begin{array}{l}\mathrm{Q}(a_1, a_2), \mathrm{Q}(a_1, a_3), \mathrm{S}(a_1, a_2), \mathrm{S}(a_1, a_3), \\ \mathrm{T}(a_1, a_2, a_3), \mathrm{T}(a_1, a_3, a_2), \mathrm{T}(a_1, a_2, a_2), \mathrm{T}(a_1, a_3, a_3)\end{array}\right\}$$

which are disjoint in their restriction to the source schema, yet both contain the target facts $\mathrm{T}(a_1, a_2, a_3)$, $\mathrm{T}(a_1, a_3, a_2)$, $\mathrm{T}(a_1, a_2, a_2)$, and $\mathrm{T}(a_1, a_3, a_3)$.

This example illustrates how distinct target violations with non-overlapping source repair envelopes may jointly affect target tuples; we cannot determine the status of tuples in T without considering violations of both key constraints.

Suppose f is a candidate fact in T. Each support set for f may be con-

tained in only certain combinations of XR-solutions from the violation cluster influences containing f. So, to determine if f has at least one support set in every XR-solution w.r.t. the broader schema mapping, it is necessary to consider **all** combinations of XR-solutions from the violation cluster influences containing f. We call the set of violation clusters whose influences contain f the *signature* of f, denoted signature(f), and for each signature we define the *focus area*, denoted $(I_{\text{focus}}, J_{\text{focus}})$, as the union of violation cluster influences for violation clusters in the signature. Recall that I_{safe} denotes the set of source facts that are *safe*, that is, not *suspect* (Definition 6.8). In the following, let J_{safe} denote $chase(I_{\text{safe}}, \mathcal{M})$.

Theorem 6.19. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a* GAV+(GAV, EGD) *schema mapping. Let I be an $\mathbf{S}$-instance. Let J be the canonical quasi-solution for I w.r.t. $\mathcal{M}$. Let q be the atomic projection-free query $q(\mathbf{x}) := \mathrm{P}(\mathbf{x})$. Let $\mathbf{f}$ be a candidate answer for q (that is, $\mathrm{P}(\mathbf{f})$ is in J). Let I_{focus} and J_{focus} be, respectively, the source and target parts of $\bigcup\{V \mid V$ the influence for a violation cluster in* signature($\mathrm{P}(\mathbf{f})$)$\}$. *Then $f \in$ XR-certain$(q, I, \mathcal{M}) \leftrightarrow \mathrm{P}(\mathbf{f}) \in \bigcap\{chase(J'_{focus} \cup J_{safe}, \Sigma_t) \mid (I'_{focus}, J'_{focus})$ is an* XR-solution *for I_{focus} w.r.t. $\mathcal{M}$)$\}$.*

proof sketch. By definition, the violations (w.r.t. $\mathcal{M}$) in J_{focus} are exactly the violations in the clusters in signature($\mathrm{P}(\mathbf{f})$). Thus $(I_{\text{focus}}, J_{\text{focus}})$ is an exchange repair envelope for I w.r.t. $\mathcal{M}^{\text{tgd}}$ augmented with those violations. Furthermore, all of the violations in J_{focus} are pairwise independent of all of the violations in $J \setminus J_{\text{focus}}$, from which we conclude that every fact in every support set for $\mathrm{P}(\mathbf{f})$ is contained in $(I_{\text{focus}}, J_{\text{focus}})$ or in $(I_{\text{safe}}, J_{\text{safe}})$. □

It is necessary to include the chase on the right-hand side of the bi-implication in the theorem because GAV chase is not closed under union.

This result gives us the following algorithm for computing XR-certain answers for an instance I and schema mapping $\mathcal{M}$, using a (hopefully large) J_{safe} combined

with, for each signature, a (hopefully small) J_{focus}.

For the exchange phase:

> Chase I with $\mathcal{M}^{tgd}$, compute the violation set of I w.r.t. $\mathcal{M}$, and compute the support closure of each violation. Then mark source facts safe if they do not reside in any violation closure, chase I_{safe} with $\mathcal{M}^{tgd}$, and mark every resulting fact safe. Lastly, compute violation clusters, and the influence of each cluster.

For the query phase:

> For a given query q, compute the candidate facts, marking safe those with support sets in J_{safe}. Next, compute the signature of each unmarked candidate fact. Finally, generate and solve a grounded disjunctive logic program to compute the XR-certain answers to q w.r.t. $\mathcal{M}$ for $(I_{\text{focus}}, J_{\text{focus}})$ for each signature. This program is the restriction to $(I_{\text{focus}}, J_{\text{focus}})$ of the grounding of the program from Theorem 4.9. Facts in $(I_{\text{focus}}, J_{\text{focus}}) \cap (I_{\text{safe}}, J_{\text{safe}})$ may be represented by the value TRUE in the program.

6.3.5 Implementation

Using the techniques developed in Section 6.3, we have implemented the optimized approach to XR-certain query answering using Java 1.7.0_80, MySQL 5.5.42, and clingo 4.4.0. This implementation accepts the same inputs as the reference implementation (schema mapping, source instance, query), and additionally accepts a specification of where to store target instances, as a JDBC connection string. It also persists the skeleton rewriting of the schema mapping to facilitate transforming additional queries at a later time.

At this point, the optimized implementation diverges from the reference implementation. It creates tables in the target instance, deducing both data types

and appropriate choices of indexes for each table (based on the joins present in the schema mapping and the data types and indexes in the source instance). It then runs the exchange phase of the query answering algorithm described in Section 6.3.4.

The exchange phase performs the following steps, which are all implemented as Java procedures that issue sequences of SQL INSERT...SELECT... queries.

- Materializes the target instance in MySQL using a chase procedure written in Java.
- Computes violation closures.
- Approximates violation clusters.
- Computes the influences of the approximated violation clusters.
- Computes the "safe" part of the source and target instances.
- Computes fact signatures.
- Precomputes sentences of the disjunctive logic program that will be needed for each violation cluster influence.

At query time, the following steps are performed:

- The query is transformed according to the skeleton rewriting of the schema mapping.
- The query is further transformed into a projection-free atomic query using the reduction described in Lemma 3.13, by adding certain GAV constraints to the schema mapping.
- The new GAV constraints are chased, which produces candidate answers.

- Candidate answers with supports wholly-contained in the safe part of the target are marked “safe”.
- Signatures are computed for the rest of the candidate answers.
- Sentences of the disjunctive logic program are precomputed for each non-“safe” candidate answer.
- For each distinct signature, a disjunctive logic program is constructed by selecting relevant precomputed sentences.
- Each disjunctive logic program is then solved with clingo, and the results are used to mark each non-“safe” candidate answer either “accepted” or “rejected”.

The “safe” candidates and “accepted” candidates together comprise the XR-certain query answers.

6.3.6 Results

Table 6.4: Duration of the exchange phase, in seconds.

instance	L0	L3	L9	L20
duration	150.7	196.7	235.7	297.3
instance	S3	M3	L3	F3
duration	36.5	50.8	196.7	2229.7

Table 6.4 gives the runtime of the exchange phase for each instance. Notice that for large instances, the exchange phase compares very favorably against the per-query runtime of the reference implementation. The plots in Figure 6.3 give the performance of each query as we scale the rate of violations and the instance

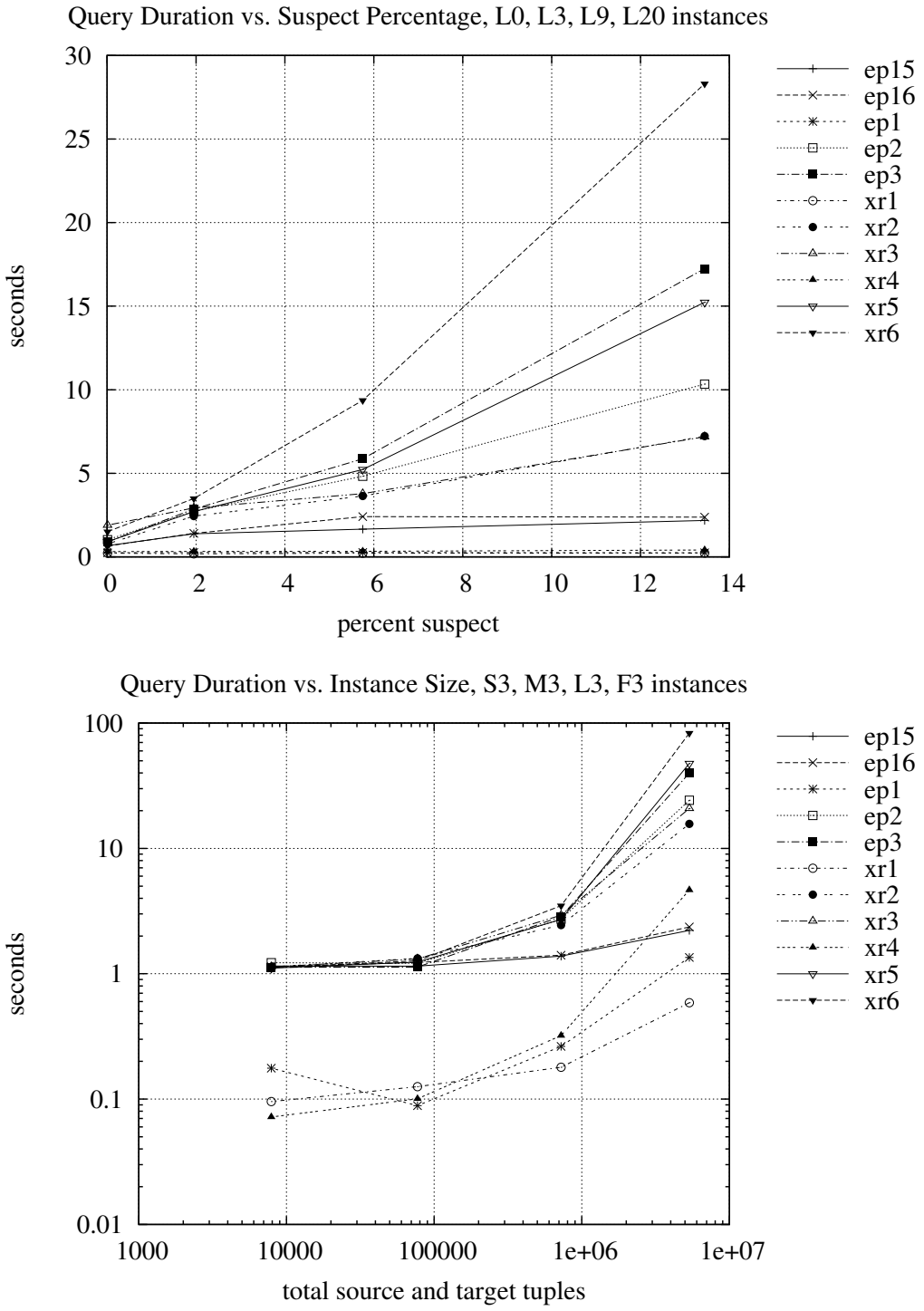

Figure 6.3: Performance of XR-certain query answering using MySQL along with clingo.

size, respectively, with the latter on a log-log scale. These results improve considerably on the reference implementation: the query phase runtimes are between ten times and one thousand times faster for large and full instances.

6.4 Discussion

Clearly, the optimized results beat the reference implementation by a considerable factor in our experiments. Although this outcome is encouraging, it is merely a single datapoint in the vast space of possible schema mappings. Nonetheless, we believe that many schema mappings and queries enjoy the significant speedup observed in this case because the defining characteristic of schema mappings that are easily exploited by the optimizations described in Section 6.3 is *data parallelism*, which is a common characteristic of schema mappings designed for integration of, and especially migration of, scientific data.

Chapter 7

Approximations of XR-certain

In this chapter, we will explore several approximations of XR-certain query answering. Prior studies in the domain of ontology-based query answering have focused on inconsistency-tolerant semantics, and moreover, on approximations. Because of the close connections between OBDA and data exchange (discussed in Section 3.1.3), the approximations developed there deserve study here. Recall that a knowledge base over a schema $\mathbf{T}$ is a pair $\langle \Sigma, D \rangle$, where D is a $\mathbf{T}$-instance and Σ is a set of constraints expressed in some logical formalism over $\mathbf{T}$. The instance D is called the ABox, and the constraint set Σ is called the TBox. Also recall that an *ABox Repair* of $\langle \Sigma, D \rangle$ is a $\mathbf{T}$-instance D' with the following properties: (i) $D' \subseteq D$; (ii) $\text{mod}(\langle \Sigma, D' \rangle) \neq \emptyset$; (iii) D' is an inclusion maximal sub-instance of D having the second property, i.e., there is no $\mathbf{T}$-instance D'' such that $D' \subset D'' \subseteq D$ and $\text{mod}(\langle \Sigma, D'' \rangle) \neq \emptyset$.

We established in Section 3.1.3 that a simple embedding of data exchange scenarios into the OBDA framework exists such that the XR-certain semantics on the original are equal to the AR-certain semantics on the embeddeding. As noted in that section, there are several key differences between data exchange and OBDA, including research goals, and the types of relations and constraints

typically studied. In what follows, we adapt approximations of the AR-certain semantics to the setting of data exchange.

We first present two types of intersection-based under-approximations of XR-certain query answering, inspired by the *Intersection of ABox Repairs (IAR) semantics* developed by Lembo *et al.* [53]. We then turn our attention to a pair of parameterized approximations, *k-support* and *k-defeater* semantics, introduced by Bienvenue and Rosati [14]. Although these approaches exhibit favorable computational behavior for various classes of constraints studied in the context of OBDA, we will see that for the classes of constraints typically used in data exchange, their computational complexity is no better than XR-certain query answering.

7.1 Intersection-based Approximation of XR-certain

In [53], Lembo *et al.* showed that an intersection-based under-approximation of AR-certain query answering is PTIME-tractable for the *DL-Lite* family of ontology languages.

Definition 7.1 (Intersection ABox Repair (IAR)-certain answers [53])**.** Given a knowledgebase $\langle \Sigma, D \rangle$ over schema $\mathbf{T}$, an *Intersection ABox Repair (IAR)* of $\langle \Sigma, D \rangle$ is a $\mathbf{T}$-instance D' such that $D' = \bigcap \{ D'' \mid D''$ is an ABox Repair of $\langle \Sigma, D \rangle \}$. Let q be a Boolean query over the schema $\mathbf{T}$. We say that q is *entailed by* $\langle \Sigma, D \rangle$ *under IAR-semantics* if for the IAR-repair D' of $\langle \Sigma, D \rangle$, for every $\mathbf{T}$-instance $J \in \text{mod}(\langle \Sigma, D' \rangle)$, we have that $J \models q$.

Now we will see how intersection-based approximation applies to our setting. First, we introduce the *intersection of source repairs (ISR)-certain answers*, and show their equivalence to the IAR semantics. Then, we explore another

intersection-based approximation of XR-certain query answering, called the *intersection of exchange repairs (IXR)-certain answers.*

Definition 7.2. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a schema mapping, let I be an $\mathbf{S}$-instance, and let q be a union of conjunctive queries. Define the *intersection of source repairs-certain answers* of q on I w.r.t. $\mathcal{M}$, denoted ISR-certain$(q, I, \mathcal{M})$, as:

$$\text{certain}(q, \bigcap \{I' \mid I' \text{ a source repair of } I \text{ w.r.t. } \mathcal{M}\}, \mathcal{M})$$

Proposition 7.3. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a schema mapping, let I be an $\mathbf{S}$-instance, and let q be a union of conjunctive queries. Then*

$$\text{ISR-certain}(q, I, \mathcal{M}) = \text{IAR-certain}(q, I, \Sigma_{st} \cup \Sigma_t)$$

Proof. The equality follows from the definitions and the embedding of exchange repair into an OBDA setting given in Section 3.1.3. □

We will now consider an alternative approximation, the intersection of exchange repair solutions.

Definition 7.4. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a schema mapping, let I be an $\mathbf{S}$-instance, and let q be a union of conjunctive queries. Define the *intersection of* XR-solution*s-certain answers* of q on I w.r.t. $\mathcal{M}$, denoted IXR-certain$(q, I, \mathcal{M})$, as:

$$q(\bigcap \{J' \mid (I', J') \text{ an XR-solution of } I \text{ w.r.t. } \mathcal{M}\})$$

In Section 6.3.2 we established that it is coNP-hard in data complexity to determine if a tuple t belongs to the intersection of source repairs, and likewise for the intersection of exchange repair solutions. We now give the corresponding upper bounds:

Proposition 7.5. *Fix a weakly acyclic schema mapping $\mathcal{M}$.*

- *The following decision problem is in* coNP*: Given a source instance I and a source fact f, does f belong to the intersection of source repairs?*
- *The following decision problem is in* coNP*: Given a source instance I and a target fact t, does t belong to the intersection of exchange repairs?*

Proof. The following is an NP algorithm to determine if f does **not** belong to the intersection of source repairs: Guess a source repair I'. Check if I' is a source repair of I w.r.t. $\mathcal{M}$ (PTIME). If so, and f does not belong to I', print "yes".

The following is an NP algorithm to determine if t does **not** belong to the intersection of exchange repair solutions: Guess a source repair I'. Check if I' is a source repair of I w.r.t. $\mathcal{M}$ (PTIME). If so, check if t belongs to chase$(I', \mathcal{M})$. If not, print "yes". □

We now relate this back to ISR-certain semantics and IXR-certain semantics.

Proposition 7.6. *Fix a weakly acyclic schema mapping $\mathcal{M}$ and a boolean query q. The following decision problems are in* coNP *(data complexity):*

- *Given a source instance I, does* ISR-certain$(q, I, \mathcal{M}) =$ TRUE*.*
- *Given a source instance I, does* IXR-certain$(q, I, \mathcal{M}) =$ TRUE*.*

Moreover, there exist a schema mapping $\mathcal{M}$ and an boolean conjunctive query q such that both problems are coNP*-hard.*

Proof. For the upper bound for ISR-certain semantics, let $I_1 \subseteq I$ such that certain$(q, I_1, \mathcal{M}) =$ TRUE. By Lemma 3.4, the intersection of source repairs has a solution w.r.t. $\mathcal{M}$, and therefore ISR-certain$(q, I, \mathcal{M}) =$ TRUE if and only if q is TRUE on a solution for the intersection of source repairs w.r.t. $\mathcal{M}$. This

motivates the following NP algorithm for the complement of ISR-certain query answering, i.e., to determine if ISR-certain$(q, I, \mathcal{M})$ = FALSE:
We will attempt to guess an instance that contains the intersection of source repairs, and whose chase w.r.t. $\mathcal{M}$ does not satisfy q. Guess an instance I', and for every source fact f in $I \setminus I'$, guess a source repair I'_f that does not contain f. This constitutes a polynomial number of polynomial-sized instances as a single guess. Verify for each I'_f that it is a source repair, and thus that f does not belong to the intersection of source repairs. Verify that $q(\text{chase}(I', \mathcal{M}))$ = FALSE– if so, print "yes".

For the upper bound for IXR-certain semantics, the following NP algorithm determines if the IXR-certain$(q, I, \mathcal{M})$ = FALSE:
We will attempt to guess an instance that contains the intersection of exchange-repair solutions, and that does not satisfy q. Let J be the canonical quasi-solution for I w.r.t. $\mathcal{M}$. Guess an instance (I', J'), and for every target fact t in $J \setminus J'$, guess an XR-solution (I'_t, J'_t) that does not contain t. This constitutes a polynomial number of polynomial-sized instances as a single guess. Verify for each (I'_t, J'_t) that it is an XR-solution, and thus that t does not belong to the intersection of exchange-repair solutions. Verify that $q(J')$ = FALSE – if so, print "yes".

For the corresponding lower bounds, consider a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ in which it is coNP-hard to determine if some fact $\text{P}(\mathbf{a})$ belongs to the intersection of source repairs (such as the schema mapping in the proof of Theorem 6.5 in Chapter 6). Let $\mathcal{M}' = (\mathbf{S}, \mathbf{T}', \Sigma'_{\text{st}}, \Sigma_{\text{t}})$ be the schema mapping where $\mathbf{T}'$ is $\mathbf{T}$ extended with P' and Σ'_{st} is Σ_{st} extended with the s-t tgd $\text{P}(\mathbf{x}) \rightarrow \text{P}'(\mathbf{x})$. Let q :- $(\exists \mathbf{x})\text{P}'(\mathbf{x})$. Then it is easy to see that for a given instance I, we have ISR-certain$(q, I, \mathcal{M})$ = TRUE if and only if $\text{P}(\mathbf{a})$ is in the intersection of source repairs. Therefore it can be coNP-hard in data complexity to determine

if ISR-certain$(q, I, \mathcal{M})$ = TRUE. The same argument also gives us that it can be coNP-hard in data complexity to determine if IXR-certain$(q, I, \mathcal{M})$ = TRUE (since P$'$ is merely a copy of P). □

The next proposition characterizes the relationships between our two intersection-based approximations and XR-certain semantics.

Proposition 7.7. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a weakly acyclic schema mapping, let I be an $\mathbf{S}$-instance, and let q be a union of conjunctive queries. Then*

$$\text{ISR-certain}(q, I, \mathcal{M}) \subseteq \text{IXR-certain}(q, I, \mathcal{M}) \subseteq \text{XR-certain}(q, I, \mathcal{M})$$

Furthermore, both containments may be proper.

Proof. To see the first containment, let $\mathbf{f}$ be a tuple in ISR-certain (q,I,$\mathcal{M}$). Then there is some image δ of $q(\mathbf{f})$ such that $\delta \subseteq$ chase$(\bigcap \{I' \mid I' \text{ a source repair of } I \text{ w.r.t. } \mathcal{M}\}, \mathcal{M})$. Since $\bigcap \{I' \mid I' \text{ a source repair of } I \text{ w.r.t. } \mathcal{M}\}$ is contained in every source repair of I (and every source repair, by definition, has a solution w.r.t. $\mathcal{M}$), and since the chase procedure is monotone for weakly acyclic schema mappings, it holds that $\delta \subseteq J'$ for every XR-solution (I', J'). Thus $\mathbf{f} \in$ IXR-certain$(q, I, \mathcal{M})$.

To see the second containment, let $\mathbf{f}$ be a tuple in IXR-certain (q,I,$\mathcal{M}$). Then there is some image δ of $q(\mathbf{f})$ such that $\delta \subseteq \bigcap \{J' \mid (I', J')$ an XR-solution of I w.r.t. $\mathcal{M}\}$, so we can trivially see that $\mathbf{f} \in$ XR-certain$(q, I, \mathcal{M})$.

To see that first containment may be proper, consider the schema mapping $\mathcal{M} = (\mathbf{S} = \{\text{S}\}, \mathbf{T} = \{\text{T}\}, \Sigma_{\text{st}} = \{\text{S}(x, y) \rightarrow \text{T}(x, y) \wedge \text{U}(x)\}, \Sigma_{\text{t}} = \{T(x, y) \wedge T(x, y') \rightarrow y = y'\})$, the $\mathbf{S}$-instance $I = \{\text{S}(a, b), \text{S}(a, c)\}$, and the query $q(x) := U(x)$. There are two source repairs of I w.r.t. $\mathcal{M}$, $\{\text{S}(a, b)\}$ and

$\{\mathrm{S}(a,c)\}$, and two corresponding canonical XR-solutions, $\{\mathrm{S}(a,b), \mathrm{T}(a,b), \mathrm{U}(a)\}$ and $\{\mathrm{S}(a,c), \mathrm{T}(a,c), \mathrm{U}(a)\}$. It is easy to see that $(a) \in$ IXR-certain$(q, I, \mathcal{M})$ because $U(a)$ is in both canonical XR-solutions. However, notice that the intersection of the source repairs is $\emptyset$, and chase$(\emptyset, \mathcal{M})$ is itself $\emptyset$, so ISR-certain$(q, I, \mathcal{M}) = \emptyset$.

To see that the second containment may be proper, consider the same schema mapping and source instance, and the query $q'(x) := \exists y T(x, y)$. Clearly, IXR-certain$(q', I, \mathcal{M}) = \emptyset$ (since there is no T-fact in common over both canonical XR-solutions). However, XR-certain$(q', I, \mathcal{M}) = \{(a)\}$ (since both canonical XR-solutions contain a fact of the form $T(a, \_)$). □

Notice that to establish that the second containment may be proper, we used a conjunctive query containing a projection. It turns out that either projection or disjunction is of the essence. Consider, for example, the schema mapping $\mathcal{M}' = (\mathbf{S} = \{\mathrm{S}\}, \mathbf{T} = \{\mathrm{T}\}, \Sigma_{\mathrm{st}} = \{\mathrm{S}(x,y) \rightarrow \mathrm{T}(x,y)\}, \Sigma_{\mathrm{t}} = \{T(x,y) \wedge T(y,x) \rightarrow y = x\})$ and the UCQ $q''(x,y) := T(x,y) \vee T(y,x)$ applied to the source instance $I' = \{\mathrm{S}(a,b), \mathrm{S}(b,a)\}$. There are two XR-solutions: $\{S(a,b), T(a,b)\}$ and $\{S(b,a), T(b,a)\{$ both of which satisfy $q''(a,b)$, even though their intersection is empty. On the other hand, if we consider a projection-free conjunctive query $q''(\mathbf{x})$, each query answer $(\mathbf{a})$ has exactly one image δ of $q(\mathbf{a})$. Therefore, if $(\mathbf{a}) \in$ XR-certain$(q'', I, \mathcal{M})$, then every XR-solution for I w.r.t. $\mathcal{M}$ contains δ, so $(\mathbf{a})$ is also contained in IXR-certain$(q'', I, \mathcal{M})$. This yields the following result:

Proposition 7.8. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a schema mapping in which Σ_{st} is a set of s-t tgds and Σ_t is the union of a set of egds with a weakly-acyclic set of tgds. Let I be an $\mathbf{S}$-instance, and let q be a projection-free conjunctive query. Then*

$$\text{IXR-certain}(q, I, \mathcal{M}) = \text{XR-certain}(q, I, \mathcal{M})$$

In light of the above, it is natural to ask the more general question of when the approximation semantics may be equal to the XR-certain semantics. We call these the ISR-certain *sufficiency problem* and the IXR-certain *sufficiency problem*, respectively.

Definition 7.9. Fix a weakly acyclic schema mapping $\mathcal{M}$ and a query q. Then the ISR-certain *sufficiency problem* is the following decision problem: Given a source instance I, is it true that ISR-certain$(q, I, \mathcal{M})$ = XR-certain$(q, I, \mathcal{M})$? The IXR-certain *sufficiency problem* is the following decision problem: Given a source instance I, is it true that IXR-certain$(q, I, \mathcal{M})$ = XR-certain$(q, I, \mathcal{M})$?

Proposition 7.10. *Let* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ *be a weakly acyclic* GLAV+(GLAV, EGD) *schema mapping. Let* q *be a boolean union of conjunctive queries. Then the* ISR-certain SUFFICIENCY PROBLEM *and the* IXR-certain SUFFICIENCY PROBLEM *are in* coDP *in data complexity. Furthermore, there exist a schema mapping* $\mathcal{M}$ *and query* q *such that the* ISR-certain SUFFICIENCY PROBLEM *and the* IXR-certain SUFFICIENCY PROBLEM *are* coDP*-complete.*

Proof. We now give a DP algorithm for the complement of the ISR-certain SUFFICIENCY PROBLEM:

Use an NP oracle to check if XR-certain$(q, I, \mathcal{M})$ = TRUE. If so, use an NP oracle to check if ISR-certain$(q, I, \mathcal{M})$ = FALSE. If so, print "yes".

We now give a DP algorithm for the complement of the IXR-certain SUFFICIENCY PROBLEM:

Use an NP oracle to check if XR-certain$(q, I, \mathcal{M})$ = TRUE. If so, use an NP oracle to check if IXR-certain$(q, I, \mathcal{M})$ = FALSE. If so, print "yes".

It is easy to see that these algorithms are correct. According to Proposition 7.7 we have that ISR-certain$(q, I, \mathcal{M}) \subseteq$ IXR-certain$(q, I, \mathcal{M}) \subseteq$ XR-certain$(q, I, \mathcal{M})$. Therefore, the answer to the complement of the ISR-certain

SUFFICIENCY PROBLEM is "yes" only if XR-certain$(q, I, \mathcal{M})$ = TRUE and ISR-certain$(q, I, \mathcal{M})$ = FALSE, and the answer to the complement of the IXR-certain SUFFICIENCY PROBLEM is "yes" only if XR-certain$(q, I, \mathcal{M})$ = TRUE and IXR-certain$(q, I, \mathcal{M})$ = FALSE. The XR-certain query answering decision problem is known to be in coNP (Theorem 3.12), and the complements of the ISR-certain and IXR-certain query answering problems are known to be in NP (Proposition 7.6).

For the corresponding lower bounds, consider the 3-SAT/UNSAT PROBLEM, which is DP-complete []: given two 3-CNF formulas Φ and Ψ, is Φ satisfiable and Ψ unsatisfiable? We will now give a reduction from 3-SAT/UNSAT to the complement of the 1-SUPPORT SUFFICIENCY PROBLEM. We will consider separate side-by-side encodings of Φ and Ψ. We simplify the presentation by using the symbol "_" to represent a variable that occurs only once in a constraint.

Let $\Phi = \phi_1 \wedge \cdots \wedge \phi_m$ be an arbitrary 3-DNF formula, where each clause ϕ_i has form $(L_1 \vee L_2 \vee L_3)$ and each L_j is a literal of the form X or $\neg X$ for some boolean variable X. Let $\Psi = \psi_1 \wedge \cdots \wedge \psi_n$ be an arbitrary 3-DNF formula, where each clause $\psi_i = (L_1 \vee L_2 \vee L_3)$ and each L_j is a literal of the form X or $\neg X$ for some boolean variable X. Assume for simplicity that Φ and Ψ use disjoint sets of variables. The complement of Ψ is $\bar{\Psi} = \bar{\psi_1} \vee \cdots \vee \bar{\psi_n}$, where each term $\bar{\psi_i} = (\bar{L_1} \wedge \bar{L_2} \wedge \bar{L_3})$ and each $\bar{L_j}$ is the complement of the literal L_j.

The letters "C", "L", "F", and "T" in the relation names of the schema mapping indicate Clauses, Literals, Formulae, and Terms, respectively.

For a given Φ and Ψ, the source instance I consists of the following facts:

- All facts of the form $\mathrm{L}_\Phi(var, polarity)$ where var is a variable in Φ (e.g., "X") and $polarity$ is a value in $\{\text{TRUE}, \text{FALSE}\}$ (corresponding to X and $\neg X$ respectively).

- All facts of the form $\mathrm{C}_{\Phi}(i, i+1, var_1, pol_1, var_2, pol_2, var_3, pol_3)$ for a term $\phi_i = (L_1 \vee L_2 \vee L_3)$ $(1 \leq i \leq m)$, where var_1, var_2, var_3 are the variables appearing in the literals L_1, L_2, L_3, and pol_1, pol_2, pol_3 are the associated polarities (TRUE for a positive literal, FALSE for a negated one).

- The fact $\mathrm{F}_{\Phi,\mathrm{UNSAT}}(1, m+1)$, where m is the number of clauses of Φ.

- All facts of the form $\mathrm{L}_{\Psi}(var, polarity)$ where var is a variable in Ψ (e.g., "Y") and $polarity$ is a value in $\{\mathrm{TRUE}, \mathrm{FALSE}\}$ (corresponding to Y and $\neg Y$ respectively).

- All facts of the form $\mathrm{T}_{\bar{\Psi}}(i, var_1, pol_1, var_2, pol_2, var_3, pol_3)$ for a term $\bar{\psi}_i = (\bar{L_1} \wedge \bar{L_2} \wedge \bar{L_3})$ $(1 \leq i \leq n)$ in the complement $\bar{\Psi}$ of Ψ, where var_1, var_2, var_3 are the variables in the literals L_1, L_2, L_3, and pol_1, pol_2, pol_3 are the complements of the associated polarities.

For example, if $\Phi = (A \vee B \vee C) \wedge (\neg A \vee \neg B \vee \neg C)$ and $\Psi = (D \vee E \vee F) \wedge (\neg D \vee E \vee F)$ (so the complement of Ψ is $\bar{\Psi} = (\neg D \wedge \neg E \wedge \neg F) \vee (D \wedge \neg E \wedge \neg F)$), then the

instance I is:

$$\left\{\begin{array}{l}
\mathrm{L}_{\Phi}(\text{“}A\text{”}, \mathrm{TRUE}), \mathrm{L}_{\Phi}(\text{“}B\text{”}, \mathrm{TRUE}), \mathrm{L}_{\Phi}(\text{“}C\text{”}, \mathrm{TRUE}), \\
\mathrm{L}_{\Phi}(\text{“}A\text{”}, \mathrm{FALSE}), \mathrm{L}_{\Phi}(\text{“}B\text{”}, \mathrm{FALSE}), \mathrm{L}_{\Phi}(\text{“}C\text{”}, \mathrm{FALSE}), \\
\mathrm{C}_{\Phi}(1, 2, \text{“}A\text{”}, \mathrm{TRUE}, \text{“}B\text{”}, \mathrm{TRUE}, \text{“}C\text{”}, \mathrm{TRUE}), \\
\mathrm{C}_{\Phi}(2, 3, \text{“}A\text{”}, \mathrm{FALSE}, \text{“}B\text{”}, \mathrm{FALSE}, \text{“}C\text{”}, \mathrm{FALSE}), \\
\mathrm{F}_{\Phi,\mathrm{UNSAT}}(1, 3), \\
\\
\mathrm{L}_{\Psi}(\text{“}D\text{”}, \mathrm{TRUE}), \mathrm{L}_{\Psi}(\text{“}E\text{”}, \mathrm{TRUE}), \mathrm{L}_{\Psi}(\text{“}F\text{”}, \mathrm{TRUE}), \\
\mathrm{L}_{\Psi}(\text{“}D\text{”}, \mathrm{FALSE}), \mathrm{L}_{\Psi}(\text{“}E\text{”}, \mathrm{FALSE}), \mathrm{L}_{\Psi}(\text{“}F\text{”}, \mathrm{FALSE}), \\
\mathrm{T}_{\bar{\Psi}}(1, \text{“}D\text{”}, \mathrm{FALSE}, \text{“}E\text{”}, \mathrm{FALSE}, \text{“}F\text{”}, \mathrm{FALSE}), \\
\mathrm{T}_{\bar{\Psi}}(2, \text{“}D\text{”}, \mathrm{TRUE}, \text{“}E\text{”}, \mathrm{FALSE}, \text{“}F\text{”}, \mathrm{FALSE})
\end{array}\right\}$$

We use the following schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$, with two parts devoted to two formulas Φ and Ψ. Note that “Φ” and “Ψ” in the schema mapping are merely annotations to indicate the purpose of each relation (likewise with “SAT” and “UNSAT”), and that the schema mapping does not depend upon the 3-SAT/UNSAT instance.

Let $\mathbf{S} = \{\mathrm{L}_{\Phi}, \mathrm{C}_{\Phi}, \mathrm{F}_{\Phi,\mathrm{UNSAT}}, \mathrm{L}_{\Psi}, \mathrm{T}_{\bar{\Psi}}\}$ with arities $2, 8, 2, 2, 7$ respectively

Let $\mathbf{T} = \{\mathrm{L}'_{\Phi}, \mathrm{C}_{\Phi,\mathrm{SAT}}, \mathrm{F}'_{\Phi,\mathrm{UNSAT}}, \mathrm{L}'_{\Psi}, \mathrm{T}_{\bar{\Psi},\mathrm{SAT}}\}$ with arities $2, 2, 2, 2, 1$ respectively

$$\text{Let } \Sigma_{\text{st}} = \left\{ \begin{array}{ll} \mathrm{L}_\Phi(var, polarity) \rightarrow \mathrm{L}'_\Phi(var, polarity) & [1] \\ \mathrm{C}_\Phi(i, i_{\text{next}}, var_1, pol_1, \_, \_, \_, \_) \wedge \mathrm{L}_\Phi(var_1, pol_1) \rightarrow \mathrm{C}_{\Phi,\text{SAT}}(i, i_{\text{next}}) & [2] \\ \mathrm{C}_\Phi(i, i_{\text{next}}, \_, \_, var_2, pol_2, \_, \_) \wedge \mathrm{L}_\Phi(var_2, pol_2) \rightarrow \mathrm{C}_{\Phi,\text{SAT}}(i, i_{\text{next}}) & [3] \\ \mathrm{C}_\Phi(i, i_{\text{next}}, \_, \_, \_, \_, var_3, pol_3) \wedge \mathrm{L}_\Phi(var_3, pol_3) \rightarrow \mathrm{C}_{\Phi,\text{SAT}}(i, i_{\text{next}}) & [4] \\ \mathrm{F}_{\Phi,\text{UNSAT}}(x, y) \rightarrow \mathrm{F}'_{\Phi,\text{UNSAT}}(x, y) & [5] \\ \\ \mathrm{L}_\Psi(var, polarity) \rightarrow \mathrm{L}'_\Psi(var, polarity) & [6] \\ \mathrm{T}_{\bar{\Psi}}(i, var_1, pol_1, var_2, pol_2, var_3, pol_3) \wedge & \\ \qquad \mathrm{L}_\Psi(var_1, pol_1) \wedge \mathrm{L}_\Psi(var_2, pol_2) \wedge \mathrm{L}_\Psi(var_3, pol_3) \rightarrow \mathrm{T}_{\bar{\Psi},\text{SAT}}(i) & [7] \end{array} \right\}$$

$$\text{Let } \Sigma_{\text{t}} = \left\{ \begin{array}{ll} \mathrm{L}'_\Phi(var, polarity) \wedge \mathrm{L}'_\Phi(var, polarity') \rightarrow polarity = polarity' & [8] \\ \mathrm{C}_{\Phi,\text{SAT}}(i_1, i_2) \wedge \mathrm{C}_{\Phi,\text{SAT}}(i_2, i_3) \rightarrow \mathrm{C}_{\Phi,\text{SAT}}(i_1, i_3) & [9] \\ \mathrm{F}'_{\Phi,\text{UNSAT}}(x, y) \wedge \mathrm{C}_{\Phi,\text{SAT}}(x, y) \rightarrow x = y & [10] \\ \\ \mathrm{L}'_\Psi(var, polarity) \wedge \mathrm{L}'_\Psi(var, polarity') \rightarrow polarity = polarity' & [11] \end{array} \right\}$$

$$\text{Let } q_{\Phi,\text{UNSAT}} \text{ :- } (\exists x, y)\mathrm{F}'_{\Phi,\text{UNSAT}}(x, y)$$

$$\text{Let } q_{\bar{\Psi},\text{VALID}} \text{ :- } (\exists x)\mathrm{T}_{\bar{\Psi},\text{SAT}}(x)$$

$$\text{Let } q \text{ :- } q_{\Phi,\text{UNSAT}} \cup q_{\bar{\Psi},\text{VALID}}$$

Claim: Φ is satisfiable and Ψ is unsatisfiable if and only if ISR-certain$(q, I, \mathcal{M}) \neq$ XR-certain$(q, I, \mathcal{M})$.

The schema mapping is constructed such that source repairs of an instance correspond to assignments to the variables of the formulae Φ and Ψ. We will intially consider the two parts of the schema mapping separately.

Source-to-target tgd [1] encodes that L_Φ is copied to the target. Source-to-target tgds [2],[3], and [4] encode that the fact $\mathrm{C}_{\Phi,\text{SAT}}(i, i_{\text{next}})$ should be generated in the target if any one of the L-facts representing the literals of clause ϕ_i is present

in the source. Thus the fact $\mathrm{C}_{\Phi,\mathrm{SAT}}(i, i_{\mathrm{next}})$ indicates satisfaction of clause ϕ_i by the literals that hold in the given source. Source-to-target tgd [5] encodes that $\mathrm{F}_{\Phi,\mathrm{UNSAT}}$ is copied to the target.

Target constraint [8] enforces that a literal in Φ and its complement cannot both hold. Target constraint [9] encodes transitivity of the $\mathrm{C}_{\Phi,\mathrm{SAT}}$ relation, so that the two attributes of the relation hold a (left-inclusive, right-exclusive) range of clauses that are all satisfied. Target constraint [10] encodes that a fact $\mathrm{C}_{\Phi,\mathrm{SAT}}(x, y)$ (which represents satisfaction of clauses $[x, y)$ of Φ) cannot coexist with the fact $\mathrm{F}'_{\Phi,\mathrm{UNSAT}}(x, y)$ (which represents unsatisfiability of the formula given by clauses $[x, y)$ of Φ).

If Φ is not satisfiable, then for any source instance I' that has a solution w.r.t. $\mathcal{M}$, it holds that $I' \cup \{\mathrm{F}_{\Phi,\mathrm{UNSAT}}(1, m+1)\}$ also has a solution w.r.t. $\mathcal{M}$. On the other hand, if Φ is satisfiable, then there is some source repair (corresponding to a satisfying assignment of Φ) that excludes the fact $\mathrm{F}_{\Phi,\mathrm{UNSAT}}(1, m+1)$. It should now be plain to see that we have ISR-certain$(q_{\Phi,\mathrm{UNSAT}}, I, \mathcal{M}) =$ FALSE if and only if Φ is satisfiable.

For the second part of the schema mapping, s-t tgd [6] encodes that the fact $\mathrm{C}_{\bar{\Psi},\mathrm{SAT}}(i)$ should be generated in the target if the term $\bar{\psi}_i$ is satisfied. Source-to-target tgd [7] encodes that L_{Ψ} is copied to the target. Target constraint [11] encodes that a literal from Ψ and its complement cannot both hold.

Each source repair will correspond to an assignment to the variables of Ψ (it will contain either $\mathrm{L}_{\Psi}(\text{“}X''$, TRUE) or $\mathrm{L}_{\Psi}(\text{“}X''$, FALSE) for each variable X in Ψ). Thus we have XR-certain$(q_{\bar{\Psi},\mathrm{VALID}}, I, \mathcal{M}) =$ TRUE if and only if $\bar{\Psi}$ is valid.

Notice that ISR-certain$(q_{\bar{\Psi},\mathrm{VALID}}, I, \mathcal{M}) =$ FALSE regardless of whether $\bar{\Psi}$ is valid, because the intersection of all source repairs contains no positive nor negative literals for Ψ, and thus has solutions containing no $\mathrm{T}_{\bar{\Psi},\mathrm{SAT}}$ facts.

Therefore ISR-certain$(q, I, \mathcal{M})$ = ISR-certain$(q_{\Phi,\text{UNSAT}}, I, \mathcal{M})$, which we previously established will be FALSE if Φ is satisfiable and TRUE otherwise. If ISR-certain$(q_{\Phi,\text{UNSAT}}, I, \mathcal{M})$ = FALSE, then since Φ is satisfiable we will have also that XR-certain$(q_{\Phi,\text{UNSAT}}, I, \mathcal{M})$ = FALSE. Therefore, XR-certain$(q, I, \mathcal{M})$ = XR-certain$(q_{\bar{\Psi},\text{VALID}}, I, \mathcal{M})$, which is TRUE if Ψ is unsatisfiable and false otherwise. Finally, (using Proposition 7.7 again) we have that Φ is satisfiable and Ψ is unsatisfiable if and only if ISR-certain$(q, I, \mathcal{M}) \neq$ XR-certain$(q, I, \mathcal{M})$.

Claim: Φ is satisfiable and Ψ is unsatisfiable if and only if IXR-certain$(q, I, \mathcal{M}) \neq$ XR-certain$(q, I, \mathcal{M})$. The argument above holds for IXR-certain semantics as well, given the following observation: for every XR-solution containing a fact t in the $\text{T}_{\bar{\Psi},\text{SAT}}$ relation, there is another XR-solution corresponding to an assignment where the variables in the clause represented by t are assigned the complements of their polarities in the clause. In other words, there is an XR-solution that makes that clause false, and thus excludes the fact t. Therefore the intersection of XR-solutions contains no $\text{T}_{\bar{\Psi},\text{SAT}}$ facts. The rest of the argument applies without modification. □

Finally, one might ask whether there are simple syntactic conditions on a schema mapping and query that guarantee the answer to the ISR-certain SUFFICIENCY PROBLEM will be trivially "yes" (as we saw in Proposition 7.8 for IXR-certain semantics). Given a weakly acyclic schema mapping $\mathcal{M}$ and a query q, it should be easy to see that if there is only a single unique minimal source instance I_q for which $q(\text{chase}(I_q, \mathcal{M})$ = TRUE, then ISR-certain$(q, I, \mathcal{M})$ = XR-certain$(q, I, \mathcal{M})$ for all I (because if I contains I_q then q is TRUE in both semantics, and otherwise q is TRUE in neither). However, the syntactic properties that guarantee such a single unique minimal source instance exists are quite restrictive. What follows is an example of such syntactic conditions. Define the

term *projection-free tgd* to mean a tgd where every variable appearing on the left-hand side also appears in every atom on the right-hand side.

Proposition 7.11. *Let* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ *be a* LAV+(LAV, EGD) *schema mapping in which the tgds are projection-free, and no relation symbol appears on the right-hand side of more than one tgd, and let* q *be a boolean projection-free conjunctive query. Then* ISR-certain$(q, I, \mathcal{M})$ = XR-certain$(q, I, \mathcal{M})$ *for all source instances* I*.*

Proof. First, notice that the restriction on relation symbol appearances means that the $\mathcal{M}$ is acyclic (and of course, also weakly acyclic).

We will give a proof by induction. For the base case, take the image of q – call it I_0. Clearly $q(\text{chase}(I_0, \mathcal{M})$ = TRUE. For the inductive step, we will compute I_{k+1} from I_k. Assume $q(\text{chase}(I_k, \mathcal{M})$ = TRUE. For each fact f in I_k, if f is a source fact, then f is copied into I_{k+1} as well. Otherwise, f can be generated by at most one tgd τ in $\mathcal{M}$, and since the tgds are projection-free, there is exactly one image of the left-hand side of τ that could generate f. Place the facts of that image into I_{k+1}. It's easy to see that $q(\text{chase}(I_{k+1}, \mathcal{M})$ = TRUE. Stop if all of the facts in I_{k+1} are source facts. The induction will terminate in a finite number of steps because $\mathcal{M}$ is acyclic, and the final image I_n is the unique minimal source instance such that $q(\text{chase}(I_n, \mathcal{M})$ = TRUE. □

7.2 Parameterized Approximations of XR-certain

In this section, we introduce a generalization of the ISR-certain semantics, called *k-support* XR-certain *answers*. We will see that these semantics coincide with the *k-support semantics* introduced by Bienvenue and Rosati in [14] as an

approximation of AR-certain semantics. In that work, Bienvenue and Rosati find that k-support semantics are tractable (in fact, first-order rewritable) for most ontology languages in the *DL-Lite* family (specifically, those that are known to be first-order rewritable for classical OBDA query answering). In practice, this means that it is possible to add approximate inconsistency tolerance to systems utilizing these ontology languages without sacrificing first-order rewritability and the practical advantages included therewith. In principal, one could pursue a similar generalization of IXR-certain semantics as well, but we choose here to focus on k-support semantics because its utility has been demonstrated in the literature. In the subsections below, we will explore the computational properties of this approximation more broadly with respect to the constraint languages typically studied in data exchange, and later in the section we will explore the complementary notion of *k-defeater semantics*, also from [14]. First, we give the definition of k-support XR-certain answers and relate it to the definition of k-support semantics.

Definition 7.12 (k-support XR-certain answers)**.** Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a schema mapping specified by tgds and egds. Let I be an $\mathbf{S}$-instance. Let q be a union of conjunctive queries. A tuple $\mathbf{t}$ is a *k-support XR-Certain answer* of q over I w.r.t. $\mathcal{M}$, denoted $\mathbf{t} \in k\text{-supp-XR-certain}(q, I, \mathcal{M})$, if there exist subsets $s_1, \ldots, s_k$ of I such that every source repair I' of I w.r.t. $\mathcal{M}$ contains at least one of $s_1, \ldots, s_k$, and where for each s_i we have $\mathbf{t} \in \text{certain}(q, s_i, \mathcal{M})$.

In order to give the definition of k-support semantics as stated in [14], we need to first introduce some terminology and basic definitions. Let $\langle \Sigma, D \rangle$ be a knowledgebase. A set S of ground facts is called Σ-*consistent* if $\langle \Sigma, S \rangle \not\models \bot$. A set of facts $S \subseteq D$ is said to be a Σ-support for query q in D if S is Σ-consistent and $\langle \Sigma, S \rangle \models q$.

Definition 7.13 (k-support semantics [14])**.** A query q is entailed by $\langle \Sigma, D \rangle$ under the *k-support semantics*, written $\langle \Sigma, D \rangle \models_{\text{k-supp}} q$, if there exist (not necessarily distinct) subsets $S_1, \ldots, S_k$ of D satisfying the following conditions:

- each S_i is a Σ-support for q in D
- for every ABox repair D', there is some S_i with $S_i \subseteq D'$.

We are now ready to show that k-support XR-certain answers coincide with the k-support semantics when we embed a data exchange scenario into OBDA as described in Proposition 3.11. The proof is little more than an unraveling of the definitions.

Proposition 7.14. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a* GAV+(GAV, EGD) *schema mapping. Let I be an $\mathbf{S}$-instance. Consider the knowledgebase $\langle \Sigma_{st} \cup \Sigma_t, I \rangle$ with I as the ABox and the union $\Sigma_{st} \cup \Sigma_t$ over the schema $\mathbf{S} \cup \mathbf{T}$ as the TBox. Let k be a constant. Then for every UCQ q it holds that $\mathbf{a} \in k$-supp-XR-certain$(q, I, \mathcal{M})$ if and only if $\langle \Sigma_{st} \cup \Sigma_t, I \rangle \models_{k-supp} q(\mathbf{a})$.*

Proof of Proposition 7.14. Suppose $\mathbf{a} \in k$-supp-XR-certain$(q, I, \mathcal{M})$. Then there exist subsets $s_1, \ldots, s_k$ of I where for each s_i we have $s_i, \Sigma_{\text{st}} \cup \Sigma_{\text{t}} \models q$, and for every source repair I' of I w.r.t. $\mathcal{M}$, we have that I' contains at least one of $s_1, \ldots, s_k$. By Proposition 3.11, we have that I' is a source repair of I w.r.t. $\mathcal{M}$ if and only if I' is an ABox repair of $\langle \Sigma_{\text{st}} \cup \Sigma_{\text{t}}, I \rangle$. Thus $\langle \Sigma_{\text{st}} \cup \Sigma_{\text{t}}, I \rangle \models_{k-\text{supp}} q$. It is easy to see that the reverse direction also holds. □

7.2.1 k-Support Exchange-Repair Certain Answers

We will now explore the relationship of k-supp-XR-certain semantics to XR-certain semantics, as well its computational properties. To start, we give the relationship between k-supp-XR-certain semantics and ISR-certain semantics.

Proposition 7.15. *Let* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ *be a weakly acyclic* GLAV+(GLAV, EGD) *schema mapping. Let* I *be an* $\mathbf{S}$*-instance. Let* q *be a UCQ. Then* ISR-certain$(q, I, \mathcal{M})$ = 1-supp-XR-certain$(q, I, \mathcal{M})$.

Proof. W.L.O.G. assume q is boolean. Suppose ISR-certain$(q, I, \mathcal{M})$ = TRUE. Then the intersection of source repairs of I w.r.t. $\mathcal{M}$ contains some subset S such that certain$(q, S, \mathcal{M})$ = TRUE. Therefore 1-supp-XR-certain$(1, I, \mathcal{M})$ = TRUE, because every source repair of I w.r.t. $\mathcal{M}$ contains S. □

This result mirrors Theorem 1 of [14].

Next, we compare k-supp-XR-certain semantics back to XR-certain semantics.

Proposition 7.16. *Let* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ *be a weakly acyclic* GLAV+(GLAV, EGD) *schema mapping. Let* I *be an* $\mathbf{S}$*-instance. Let* q *be a UCQ. Then for all* k *we have* k-supp-XR-certain$(q, I, \mathcal{M}) \subseteq$ XR-certain$(q, I, \mathcal{M})$.

Proof. Consider an arbitrary tuple $\mathbf{a} \in k$-supp-XR-certain$(q, I, \mathcal{M})$. Then by definition, every source repair I' of I contains one of the subsets $s_1, \ldots, s_k \subset I$, where for each s_i we have $\mathbf{a} \in$ certain$(q, s_i, \mathcal{M})$. Thus $\mathbf{a} \in$ certain$(q, I', \mathcal{M})$ for each source repair I', so $\mathbf{a} \in$ XR-certain$(q, I, \mathcal{M})$. □

The next proposition introduces the K-SUPPORT CHECKING problem, which is a generalization of the problem of determining if a tuple belongs to the intersection of source repairs. The problem will be helpful for evaluating the computational behavior of k-supp-XR-certain semantics.

Proposition 7.17. *Let the* K-SUPPORT CHECKING *problem be the following decision problem: Fix a schema mapping* $\mathcal{M}$*. Given a source instance* I *and a set of subsets* $I_1, \ldots, I_k \subseteq I$*, is it true that every source repair of* I *w.r.t.* $\mathcal{M}$ *contains at*

least one of $I_1, \ldots, I_k$*? The* K-SUPPORT CHECKING *problem is in* coNP *(data complexity). Furthermore, there is a schema mapping* $\mathcal{M}$ *such that the* K-SUPPORT CHECKING *problem is* coNP*-hard.*

Proof. The following is an NP algorithm for the complement of the K-SUPPORT CHECKING problem: Guess a subset $I' \subseteq I$ and check whether it is a source repair (in PTIME per Theorem 3.12). If so, check whether any of $I_1, \ldots, I_k$ is contained in I'. If none are, print "yes".

For the lower bound, recall that the problem of determining whether a source fact f is contained in the intersection of all source repairs is coNP-hard in data complexity (Theorem 6.5). That problem is a special case of the K-SUPPORT CHECKING problem where $k = 1$ and $I_1 = \{f\}$. Therefore there exists a schema mapping $\mathcal{M}$ for which the K-SUPPORT CHECKING problem is coNP-hard in data complexity. □

We are now ready to establish the computational complexity of k-supp-XR-certain semantics. Fix a schema mapping $\mathcal{M}$, query q, and a constant k. Let the SUPPORT XR-CERTAIN$(q, \mathcal{M}, k)$ problem be the following decision problem: given an instance I and a tuple $\mathbf{a}$, is $\mathbf{a}$ in k-supp-XR-certain$(q, I, \mathcal{M})$?

Proposition 7.18. *Let* $\mathcal{M}$ *be a weakly acyclic schema mapping, let* q *be a UCQ, and let* k *be a natural number. Then the* SUPPORT XR-CERTAIN$(q, \mathcal{M}, k)$ *problem for* $\mathcal{M}$*,* q*, and* k *is in* Σ_2^{P}*.*

Proof. The following is a Σ_2^{P} algorithm (NP with an NP oracle) to determine if $\mathbf{a} \in k$-supp-XR-certain$(q, I, \mathcal{M})$: Guess a set of k subsets $I_1, \ldots, I_k \subseteq I$. For each I_i, check whether I_i has a solution w.r.t. $\mathcal{M}$, and if so, check whether $\mathbf{a}$ is an answer to q over a universal solution for I_i w.r.t. $\mathcal{M}$ (in PTIME, per [32]). If these conditions hold for all I_i, use an NP oracle to check the K-SUPPORT

CHECKING problem for $I_1, \ldots, I_k$, and if successful, print "yes". The correctness of this algorithm is easy to see. □

The lower-bound complexity of k-supp-XR-certain when k is fixed remains an open problem for $k > 1$. However, we will see next that for the uniform version of the problem, the Σ_2^{P} upper bound still holds, and moreover, the corresponding lower bound is known. Fix a schema mapping $\mathcal{M}$, and a query q. Let the UNIFORM XR-CERTAIN$(q, \mathcal{M})$ problem be the following decision problem: given an instance I, a tuple $\mathbf{a}$, and a natural number k in unary, is $\mathbf{a}$ in k-supp-XR-certain$(q, I, \mathcal{M})$?

Proposition 7.19. *Let $\mathcal{M}$ be a weakly acyclic schema mapping, and let q be a UCQ. Then the* UNIFORM SUPPORT XR-CERTAIN$(q, \mathcal{M}, k)$ *problem for $\mathcal{M}$ and q is in Σ_2^{P}. Furthermore, there is a weakly acyclic schema mapping $\mathcal{M}$ and a boolean conjunctive query q such that the* UNIFORM SUPPORT XR-CERTAIN$(q, \mathcal{M}, k)$ *problem for $\mathcal{M}$ and q is Σ_2^{P}-hard.*

Proof. The following is a Σ_2^{P} algorithm (NP with an NP oracle) for the UNIFORM SUPPORT XR-CERTAIN$(q, \mathcal{M}, k)$ problem: Guess a set of k subsets $I_1, \ldots, I_k \subseteq I$. For each I_i, check whether I_i has a solution w.r.t. $\mathcal{M}$, and if so, check whether $\mathbf{a}$ is an answer to q over a universal solution for I_i w.r.t. $\mathcal{M}$ (in PTIME, per [32]). If these conditions hold for all I_i, use an NP oracle to check the K-SUPPORT CHECKING problem for $I_1, \ldots, I_k$, and if successful, print "yes". The correctness of this algorithm is easy to see.

For the lower bound, consider the following decision problem, which we call k-TERM DNF TAUTOLOGY: Given a 3-DNF tautology Φ and an integer k, is there a subset of at most k terms from Φ whose disjunction is also a tautology? This is the standard parameterization of the MIN DNF TAUTOLOGY optimization problem, which is known to be Σ_2^{P}-hard [74]. Notice that the problem becomes trivial for

$k \geq |\Phi|$, where $|\Phi|$ is the number of terms in Φ, and this gives us Σ_2^{P}-hardness for k-TERM DNF TAUTOLOGY (where k is part of the input) as well.

We will now give a reduction from k-TERM DNF TAUTOLOGY to UNIFORM SUPPORT XR-CERTAIN$(q, \mathcal{M}, k)$. Our reduction uses the following schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$, and query q:

$$\text{Let } \mathbf{S} = \{\mathrm{L}, \mathrm{T}\} \text{ with arities 2 and 7 respectively}$$

$$\text{Let } \mathbf{T} = \{\mathrm{L}', \mathrm{T}'\} \text{ with arities 2 and 1 respectively}$$

$$\text{Let } \Sigma_{\mathrm{st}} = \left\{ \begin{array}{l} \mathrm{L}(\mathit{var}, \mathit{polarity}) \rightarrow \mathrm{L}'(\mathit{var}, \mathit{polarity}) \\ \mathrm{T}(i, \mathit{var}_1, \mathit{pol}_1, \mathit{var}_2, \mathit{pol}_2, \mathit{var}_3, \mathit{pol}_3) \rightarrow \mathrm{L}'(\mathit{var}_1, \mathit{pol}_1) \wedge \\ \qquad\qquad \mathrm{L}'(\mathit{var}_2, \mathit{pol}_2) \wedge \mathrm{L}'(\mathit{var}_3, \mathit{pol}_3) \wedge \mathrm{T}_{\mathrm{SAT}}(i) \end{array} \right\}$$

$$\text{Let } \Sigma_{\mathrm{t}} = \left\{ \; \mathrm{L}'(\mathit{vid}, \mathit{polarity}) \wedge \mathrm{L}'(\mathit{vid}, \mathit{polarity}') \rightarrow \mathit{polarity} = \mathit{polarity}' \; \right\}$$

Let q :- $(\exists x)\mathrm{T}_{\mathrm{SAT}}(x)$.

Let $\Phi = \phi_1 \vee \cdots \vee \phi_n$ be an arbitrary 3-DNF formula, where each term $\phi_i = (L_1 \wedge L_2 \wedge L_3)$ and each L_j is a literal of the form X or $\neg X$ for some boolean variable X. For a given Φ, the source instance I consists of the following facts:

- All facts of the form $\mathrm{L}(\mathit{var}, \mathit{polarity})$ where var is a variable in Φ (e.g., "X") and $\mathit{polarity}$ is a value in $\{\mathrm{TRUE}, \mathrm{FALSE}\}$ (corresponding to X and $\neg X$ respectively).

- All facts of the form $\mathrm{T}(i, \mathit{var}_1, \mathit{pol}_1, \mathit{var}_2, \mathit{pol}_2, \mathit{var}_3, \mathit{pol}_3)$ for a term $\phi_i = (L_1 \wedge L_2 \wedge L_3)$ $(1 \leq i \leq n)$, where $\mathit{var}_1, \mathit{var}_2, \mathit{var}_3$ are the variables appearing in the literals L_1, L_2, L_3, and $\mathit{pol}_1, \mathit{pol}_2, \mathit{pol}_3$ are the associated polarities (TRUE for a positive literal, FALSE for a negated one).

Claim: k-supp-XR-certain$(q, I, \mathcal{M})$ = TRUE if and only if there is a subset of at most k terms from Φ whose disjunction is a tautology.

We will see that such a source instance has one source repair for every assignment. The truth of terms and literals in Φ for a given assignment is encoded by existence of corresponding facts in the repair. The first s-t tgd of the schema mapping copies encodings of literals to the target, and the target key constraint enforces that a literal and its complement cannot both hold. It's easy to see that this produces one source repair for every assignment: for a source instance I_1 in which a literal and its complement both hold, the target egd ensures that I_1 has no solution, whereas for a source instance I_2 which has a solution, if I_2 contains neither a literal L("X'', "TRUE$''$) nor its complement L("X'', "FALSE$''$), clearly $I_2 \cup \{\text{L}(\text{“}X'', \text{“TRUE}'')\}$ also has a solution (and likewise with the literal's complement). The second s-t tgd produces in the target, for each term, the three literals required to satisfy it. Thus, for source repaires representing assignments where those literals do not hold, the source representation of the term also cannot hold, and for source repairs representing assignments where those literals do hold, the source representation of the term must hold (to achieve maximality amongst source instances that have solutions). Therefore, if there is a subset of k terms from Φ whose disjunction is a tautology, then at least one of those k terms is contained in every source repair. Then we have k-supp-XR-certain$(q, I, \mathcal{M})$ = TRUE if and only if there is a subset of at most k terms from Φ whose disjunction is a tautology. □

In addition to studying the problem of evaluation for k-supp-XR-certain semantics, it is natural to ask when this approximation coincides with XR-certain. The following decision problem captures this question. We will utilize the complexity of the evaluation problem to help obtain bounds for the sufficiency prob-

lem, so we consider both the fixed-k and uniform versions here.

Definition 7.20. Fix a schema mapping $\mathcal{M}$, a query q, and a natural number k. Let the SUPPORT XR-CERTAIN$(q, \mathcal{M}, k)$ problem be the following decision problem: Given a source instance I, is it true that k-supp-XR-certain$(q, I, \mathcal{M}) =$ XR-certain$(q, I, \mathcal{M})$?

Fix a schema mapping $\mathcal{M}$ and a query q. Let the UNIFORM SUPPORT XR-CERTAIN$(q, \mathcal{M}, k)$ problem be the following decision problem: Given a source instance I and a natural number k in unary, is it true that k-supp-XR-certain$(q, I, \mathcal{M}) =$ XR-certain$(q, I, \mathcal{M})$?

Proposition 7.21. *Let $\mathcal{M}$ be a weakly acyclic schema mapping, let q be a UCQ, and let k be a natural number. Then the* SUPPORT XR-CERTAIN$(q, \mathcal{M}, k)$ *problem is in Σ_2^{P} (data complexity).*

Proof. Without loss of generality, assume q is a boolean query. We now give a Σ_2^{P} algorithm (NP with an NP oracle) for the UNIFORM SUPPORT XR-CERTAIN$(q, \mathcal{M})$ SUFFICIENCY problem:
Guess k subsets $I_1, \ldots, I_k \subseteq I$ and for each I_i check whether certain$(q, I_i, \mathcal{M}) =$ TRUE (in PTIME, per [32]). If so (for all I_i), use an NP oracle to check the K-SUPPORT CHECKING problem for $I_1, \ldots, I_k$. If so, print "yes". □

The lower-bound complexity of checking whether k-supp-XR-certain$(q, I, \mathcal{M}) =$ XR-certain$(q, I, \mathcal{M})$ when k is fixed remains an open problem for $k > 1$.

Proposition 7.22. *Let $\mathcal{M}$ be a weakly acyclic schema mapping, and let q be a UCQ. Then the* UNIFORM SUPPORT XR-CERTAIN$(q, \mathcal{M})$ SUFFICIENCY *problem is in Σ_2^{P} (data complexity). Furthermore, there exist a schema mapping $\mathcal{M}$ and a*

boolean conjunctive query q such that the UNIFORM SUPPORT XR-CERTAIN$(q, \mathcal{M})$ SUFFICIENCY *problem is Σ_2^{P}-hard.*

Proof. Without loss of generality, assume q is a boolean query. We now give a Σ_2^{P} algorithm (NP with an NP oracle) for the UNIFORM SUPPORT XR-CERTAIN$(q, \mathcal{M})$ SUFFICIENCY problem:
Guess k subsets $I_1, \ldots, I_k \subseteq I$ and for each I_i check whether certain$(q, I_i, \mathcal{M}) =$ TRUE (in PTIME, per [32]). If so (for all I_i), use an NP oracle to check the K-SUPPORT CHECKING problem for $I_1, \ldots, I_k$. If so, print "yes".

For the lower bound, let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$ be a weakly acyclic GLAV+(GLAV, EGD) schema mapping, let q_1 be a boolean conjunctive query, and let k be a constant such that it is Σ_2^{P}-hard to determine if k-supp-XR-certain$(q_1, I_1, \mathcal{M}_\infty) =$ TRUE (such as in the proof of Proposition 7.19). Observe that we can easily construct a query $q_2 \text{ :- } F_1'(a) \wedge \ldots, F_{k+1}'(a)$ (where a is an arbitrary constant), a source instance $I_2 = \{F_1(a), \ldots, F_{k+1}(a)\}$, and an s-t tgd $F(x) \rightarrow F'(x)$ such that k-supp-XR-certain$(q_2, I_2, (\{F\}, \{F'\}, \{F(x) \rightarrow F'(x)\}, \{\})) =$ FALSE but XR-certain$(q_2, I_2, (\{F\}, \{F'\}, \{F(x) \rightarrow F'(x)\}, \{\})) =$ TRUE. Assume w.l.o.g. that F and F' do not appear in q_1 or $\mathcal{M}_\infty$. Then we can simply combine our two data exchange scenarios into one: $I = I_1 \cup I_2$, and $\mathcal{M}$ is composed of the unioned schemas and constraint sets of $\mathcal{M}_\infty$ and $\mathcal{M}_\in$), and $q = q_1 \vee q_2$. Clearly, (-supp-XR-certain$q, I, \mathcal{M}) =$ XR-certain$(q, I, \mathcal{M})$ if and only if (-supp-XR-certain$q_1, I_1, \mathcal{M}_\infty) =$ TRUE. □

7.2.2 k-Defeater Exchange-Repair Certain Answers

Bienvenue and Rosati introduce the k-defeater semantics to complement the k-support semantics. The k-defeater semantics is an over-approximation of

AR-certain semantics (that is, every answer given by AR-certain is also an answer under k-defeater semantics). Furthermore, k-defeater semantics converges to AR-certain for increasing k. As above, we will introduce an appropriate definition in the context of data exchange, then relate it back to Bienvenue and Rosati's definition using the embedding described in Section 3.1.3. Then, we will explore its computational properties in data exchange.

Definition 7.23 (k-defeater XR-certain answers)**.** Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a schema mapping specified by tgds and egds. Let I be an $\mathbf{S}$-instance. Let q be a union of conjunctive queries. A subset $d \subset I$ is called a *defeater* for $q(\mathbf{t})$ over I w.r.t. $\mathcal{M}$ if d has a solution w.r.t. $\mathcal{M}$ and, for every minimal subset s of I where $\mathbf{t} \in \text{certain}(q, s, \mathcal{M})$, we have that $d \cup s$ has no solution w.r.t. $\mathcal{M}$. A tuple $\mathbf{t}$ is a *k-defeater XR-Certain answer* of q over I w.r.t. $\mathcal{M}$, denoted $\mathbf{t} \in k\text{-def-XR-certain}(q, I, \mathcal{M})$, if there is no defeater d for $q(\mathbf{t})$ with size less than or equal to k.

Definition 7.24 (k-defeater semantics [14])**.** A query q is entailed by $\langle \Sigma, D \rangle$ under the *k-defeater semantics*, written $\langle \Sigma, D \rangle \models_{\text{k-def}} q$, if there does not exist a Σ-consistent subset S of D satisfying the following conditions:

- $|S| \leq k$
- $\langle \Sigma, S \cup C \rangle \models \bot$ for every minimal Σ-support $C \subseteq A$ of q.

Proposition 7.25. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a* GAV+(GAV, EGD) *schema mapping. Let I be an $\mathbf{S}$-instance. Consider the knowledgebase $\langle \Sigma_{st} \cup \Sigma_t, I \rangle$ with I as the ABox and the union $\Sigma_{st} \cup \Sigma_t$ over the schema $\mathbf{S} \cup \mathbf{T}$ as the TBox. Let $k > 0$ be a constant. Then for every UCQ q it holds that $\mathbf{a} \in k\text{-def-XR-certain}(q, I, \mathcal{M})$ if and only if $\langle \Sigma_{st} \cup \Sigma_t, I \rangle \models_{k-def} q(\mathbf{a})$.*

Proof. We will prove the contrapositive. Suppose $\langle \Sigma_{\text{st}} \cup \Sigma_{\text{t}}, I \rangle \not\models_{k-\text{def}} \mathbf{a}$. Then there exists a $\Sigma_{\text{st}} \cup \Sigma_{\text{t}}$-consistent subset d of D such that $|d| \leq k$ and $\langle \Sigma_{\text{st}} \cup \Sigma_{\text{t}}, d \cup C \rangle \models \bot$ for every minimal $\Sigma_{\text{st}} \cup \Sigma_{\text{t}}$-support $C \subseteq I$ of $q(\mathbf{a})$. We will show that d is a defeater for $q(\mathbf{a})$ over I w.r.t. $\mathcal{M}$. Let s be a subset of I. We have that $\mathbf{a} \in \text{certain}(q, s, \mathcal{M})$ if and only if s is a $\Sigma_{\text{st}} \cup \Sigma_{\text{t}}$-support for $q(\mathbf{a})$. Thus, every minimal $\Sigma_{\text{st}} \cup \Sigma_{\text{t}}$-support $s \subseteq I$ of $q(\mathbf{a})$ is also a minimal subset of I where $\mathbf{a} \in \text{certain}(q, s, \mathcal{M})$, and for each such s we have that $d \cup s$ has no solution w.r.t. $\mathcal{M}$. Therefore, d is a k-defeater for $q(\mathbf{a})$ over I w.r.t. $\mathcal{M}$. For the reverse direction, suppose $\mathbf{a} \notin k\text{-def-XR-certain}(q, I, \mathcal{M})$. Then there is a defeater $d \subset I$ for $q(\mathbf{a})$ such that $|d| \leq k$. By definition, for every minimal subset s where $\mathbf{a} \in \text{certain}(q, s, \mathcal{M})$ we have that $d \cup s$ has no solution w.r.t. $\mathcal{M}$, and thus for every minimal $\Sigma_{\text{st}} \cup \Sigma_{\text{t}}$-support $C \subseteq I$ of $q(\mathbf{a}$, we have $\langle \Sigma_{\text{st}} \cup \Sigma_{\text{t}}, d \cup C \rangle \models \bot$. Therefore, $\langle \Sigma_{\text{st}} \cup \Sigma_{\text{t}}, I \rangle \not\models_{\text{k-def}} \mathbf{a}$. □

Next, we consider the complexity of k-defeater semantics. Fix a schema mapping $\mathcal{M}$, query q, and a constant k. Then let the DEFEATER XR-CERTAIN$(q, \mathcal{M}, k)$ problem be the following decision problem: given an instance I and a tuple $\mathbf{a}$, is $\mathbf{a}$ in k-def-XR-certain$(q, I, \mathcal{M})$?

Proposition 7.26. *Let $\mathcal{M}$ be a weakly acyclic schema mapping, let q be a UCQ, and let k be a natural number. Then the* DEFEATER XR-CERTAIN$(q, \mathcal{M}, k)$ *problem is in* NP*. Furthermore, there is a weakly acyclic schema mapping $\mathcal{M}$, a boolean conjunctive query q, and a natural number k such that the* DEFEATER XR-CERTAIN$(q, \mathcal{M}, k)$ *problem is* NP*-hard.*

Proof. The following is an NP algorithm for the DEFEATER XR-CERTAIN$(q, \mathcal{M}, k)$ problem: For each set D consisting of at most k facts from I, guess a subset $S_D \subseteq I$. The S_D collectively constitute a single guess, polynomial in the size of I. For each D, check if $D \cup S_D$ has a solution w.r.t. $\mathcal{M}$, then check if certain$(q, D \cup$

$S_D, \mathcal{M}) = \text{TRUE}$. If both conditions hold for all S_D, print "yes". This algorithm prints "yes" if its guess is a counterexample to the existence of the k-defeater, in that every k-sized subinstance D of I is shown to be compatible with at least one instance S_D, this is, the union of D with S_D has a solution w.r.t. $\mathcal{M}$ and $\mathbf{a} \in \text{certain}(q, D \cup S_D, \mathcal{M})$.

For the corresponding lower bound, consider the 3-SAT PROBLEM, which is NP-complete[45]: given a 3-CNF formula Φ, is Φ satisfiable? We will now give a reduction from 3-SAT to 1-def-XR-certain. We simplify the presentation by using the symbol "_" to represent a variable that occurs only once in a constraint.

Let $\Phi = \phi_1 \wedge \cdots \wedge \phi_m$ be an arbitrary 3-DNF formula, where each clause ϕ_i has form $(L_1 \vee L_2 \vee L_3)$ and each L_j is a literal of the form X or $\neg X$ for some boolean variable X.

We will now give an encoding of an instance, schema mapping, and query such that Φ is satisfiable if and only if 1-def-XR-certain$(q, I, \mathcal{M}) = \text{TRUE}$. The instance captures both the formula Φ and a copy of Φ in which all literals have been complemented – this second encoding serves to make the construction symmetric, which ensures that none of the facts representing literals in Φ can act as a 1-defeater. The schema mapping, which does not depend upon the formula Φ, specifies that a particular target fact should exist if Φ is satisfied, and the query simply checks the existence of such a fact. The details of this specification are explained below. In the schema, the letters "C", "L", and "F" as relation names indicate Clauses, Literals, and Formulae, respectively, while the letter "E" indicates Either of two clauses.

For a given Φ, the source instance I consists of the following facts:

- All facts of the form L($var, polarity$) where var is a variable in Φ (e.g., "X") and $polarity$ is a value in $\{\text{TRUE}, \text{FALSE}\}$ (corresponding to X and $\neg X$

respectively).

- All facts of the form $\mathrm{C}(i, i+1, var_1, pol_1, var_2, pol_2, var_3, pol_3)$ for a term $\phi_i = (L_1 \vee L_2 \vee L_3)$ $(1 \leq i \leq m)$, where var_1, var_2, var_3 are the variables appearing in the literals L_1, L_2, L_3, and pol_1, pol_2, pol_3 are the associated polarities (TRUE for a positive literal, FALSE for a negated one).

- All facts of the form $\mathrm{C}'(i, i+1, var_1, pol'_1, var_2, pol'_2, var_3, pol'_3)$ for a term $\phi_i = (L_1 \vee L_2 \vee L_3)$ $(1 \leq i \leq m)$, where var_1, var_2, var_3 are the variables appearing in the literals L_1, L_2, L_3, and pol'_1, pol'_2, pol'_3 are the complements of the associated polarities (FALSE for a positive literal, TRUE for a negated one). This constitutes a second encoding of Φ in which each literal is complemented.

- The facts $\mathrm{F}_{\text{UNSATISFIABLE}}(1, m+1)$ and $\mathrm{F}_{\text{FULL}}(1, m+1)$, where m is the number of clauses of Φ.

For example, if $\Phi = (A \vee B \vee \neg C) \wedge (\neg A \vee \neg B \vee C)$, then the instance I is:

$$\left\{\begin{array}{l}
\mathrm{L}(\text{“}A\text{”}, \mathrm{TRUE}), \mathrm{L}(\text{“}B\text{”}, \mathrm{TRUE}), \mathrm{L}(\text{“}C\text{”}, \mathrm{TRUE}), \\
\mathrm{L}(\text{“}A\text{”}, \mathrm{FALSE}), \mathrm{L}(\text{“}B\text{”}, \mathrm{FALSE}), \mathrm{L}(\text{“}C\text{”}, \mathrm{FALSE}), \\
\mathrm{C}(1, 2, \text{“}A\text{”}, \mathrm{TRUE}, \text{“}B\text{”}, \mathrm{TRUE}, \text{“}C\text{”}, \mathrm{FALSE}), \\
\mathrm{C}(2, 3, \text{“}A\text{”}, \mathrm{FALSE}, \text{“}B\text{”}, \mathrm{FALSE}, \text{“}C\text{”}, \mathrm{TRUE}), \\
\mathrm{C}'(1, 2, \text{“}A\text{”}, \mathrm{FALSE}, \text{“}B\text{”}, \mathrm{FALSE}, \text{“}C\text{”}, \mathrm{TRUE}), \\
\mathrm{C}'(2, 3, \text{“}A\text{”}, \mathrm{TRUE}, \text{“}B\text{”}, \mathrm{TRUE}, \text{“}C\text{”}, \mathrm{FALSE}), \\
\mathrm{F}_{\text{UNSATISFIABLE}}(1, 3) \\
\mathrm{F}_{\text{FULL}}(1, 3)
\end{array}\right\}$$

We use the following schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$, which does not depend upon the 3-SAT instance.

Let $\mathbf{S} = \{\mathrm{L}, \mathrm{C}, \mathrm{F}_{\text{UNSATISFIABLE}}\}$ with arities $2, 8, 2$ respectively

Let $\mathbf{T} = \{\mathrm{L}', \mathrm{C}_{\text{SATISFIED}}, \mathrm{F}'_{\text{UNSATISFIABLE}}, \mathrm{F}'_{\text{FULL}}\}$ with arities $2, 2, 2, 2$ respectively

$$\text{Let } \Sigma_{\text{st}} = \left\{ \begin{array}{ll} \mathrm{L}(var, polarity) \rightarrow \mathrm{L}'(var, polarity) & [1] \\ \mathrm{C}(i, i_{\text{next}}, var_1, pol_1, \_, \_, \_, \_) \wedge \mathrm{L}(var_1, pol_1) \rightarrow \mathrm{C}_{\text{SATISFIED}}(i, i_{\text{next}}) & [2] \\ \mathrm{C}(i, i_{\text{next}}, \_, \_, var_2, pol_2, \_, \_) \wedge \mathrm{L}(var_2, pol_2) \rightarrow \mathrm{C}_{\text{SATISFIED}}(i, i_{\text{next}}) & [3] \\ \mathrm{C}(i, i_{\text{next}}, \_, \_, \_, \_, var_3, pol_3) \wedge \mathrm{L}(var_3, pol_3) \rightarrow \mathrm{C}_{\text{SATISFIED}}(i, i_{\text{next}}) & [4] \\ \mathrm{C}'(i, i_{\text{next}}, var_1, pol_1, \_, \_, \_, \_) \wedge \mathrm{L}(var_1, pol_1) \rightarrow \mathrm{C}'_{\text{SATISFIED}}(i, i_{\text{next}}) & [5] \\ \mathrm{C}'(i, i_{\text{next}}, \_, \_, var_2, pol_2, \_, \_) \wedge \mathrm{L}(var_2, pol_2) \rightarrow \mathrm{C}'_{\text{SATISFIED}}(i, i_{\text{next}}) & [6] \\ \mathrm{C}'(i, i_{\text{next}}, \_, \_, \_, \_, var_3, pol_3) \wedge \mathrm{L}(var_3, pol_3) \rightarrow \mathrm{C}'_{\text{SATISFIED}}(i, i_{\text{next}}) & [7] \\ \mathrm{F}_{\text{FULL}}(x, y) \rightarrow \mathrm{F}'_{\text{FULL}}(x, y) & [8] \end{array} \right\}$$

$$\text{Let } \Sigma_{\text{t}} = \left\{ \begin{array}{ll} \mathrm{L}'(var, polarity) \wedge \mathrm{L}'(var, polarity') \rightarrow polarity = polarity' & [9] \\ \mathrm{C}_{\text{SATISFIED}}(i_1, i_2) \wedge \mathrm{C}_{\text{SATISFIED}}(i_2, i_3) \rightarrow \mathrm{C}_{\text{SATISFIED}}(i_1, i_3) & [10] \\ \mathrm{C}'_{\text{SATISFIED}}(i_1, i_2) \wedge \mathrm{C}'_{\text{SATISFIED}}(i_2, i_3) \rightarrow \mathrm{C}'_{\text{SATISFIED}}(i_1, i_3) & [11] \\ \mathrm{C}_{\text{SATISFIED}}(i_1, i_2) \wedge \mathrm{F}'_{\text{FULL}}(x, y) \rightarrow \mathrm{E}_{\text{SATISFIED}}(i_1, i_2) & [12] \\ \mathrm{C}'_{\text{SATISFIED}}(i_1, i_2) \wedge \mathrm{F}'_{\text{FULL}}(x, y) \rightarrow \mathrm{E}_{\text{SATISFIED}}(i_1, i_2) & [13] \end{array} \right\}$$

$$\text{Let } q \text{ :- } (\exists x, y)\mathrm{E}_{\text{SATISFIED}}(x, y)$$

Claim: Φ is satisfiable if and only if 1-def-XR-certain$(q, I, \mathcal{M})$ = TRUE.

The schema mapping is constructed such that source repairs of an instance correspond to assignments to the variables of Φ.

Source-to-target tgd [1] encodes that L is copied to the target. Source-to-target tgds [2], [3], and [4] encode that the fact $\mathrm{C}_{\text{SATISFIED}}(i, i_{\text{next}})$ should be generated in the target if any one of the L-facts representing the literals of clause ϕ_i is present in the source. Thus the fact $\mathrm{C}_{\text{SATISFIED}}(i, i_{\text{next}})$ indicates satisfaction of clause ϕ_i by the assignment represented by the literals in the given source. Source-to-target

tgds [5], [6], and [7] encode the same for the literal-complemented copy of the clauses of Φ. Source-to-target tgd [8] encodes that $\mathrm{F}_{\mathrm{FULL}}$ is copied to the target.

Target constraint [9] enforces that a literal in Φ and its complement cannot both hold. Target constraints [10] and [11] encode transitivity of the $\mathrm{C}_{\mathrm{SATISFIED}}$ relation and the $\mathrm{C}'_{\mathrm{SATISFIED}}$ relation, so that the two attributes of each relation hold a (left-inclusive, right-exclusive) range of clauses that are all satisfied. Target constraints [12] and [13] populate $\mathrm{E}_{\mathrm{SATISFIED}}$ if $\mathrm{C}_{\mathrm{SATISFIED}}$ or $\mathrm{C}'_{\mathrm{SATISFIED}}$ contains a fact representing the satisfaction of the entire formula or its literal-complement, so a fact $\mathrm{E}_{\mathrm{SATISFIED}}(x, y)$ indicates that either the range $[x, y)$ of clauses of Φ is satisfied, or else the range $[x, y)$ of literal-complemented clauses of Φ is satisfied.

If Φ is unsatisfiable, then there is no subset s of I where $\mathrm{certain}(q, s, \mathcal{M}) = \mathrm{TRUE}$, so $\text{1-def-XR-certain}(q, I, \mathcal{M}) = \mathrm{FALSE}$. If Φ is satisfiable, then there exists an exchange-repair solution (I', J') where I' corresponds to a satisfying assignment a for Φ, and thus J' contains the fact $\mathrm{C}_{\mathrm{SATISFIED}}(1, m+1)$ where m is the number of clauses in Φ, and J' must also contain the fact $\mathrm{E}_{\mathrm{SATISFIED}}(1, m+1)$. Moreover, there is another exchange-repair solution (I'', J'') where I' corresponds to the complement of a. It is easy to see that the complement of a is a satisfying assignment for the literal-complemented copy of Φ, and therefore that J'' contains $\mathrm{C}'_{\mathrm{SATISFIED}}(1, m+1)$ and $\mathrm{E}_{\mathrm{SATISFIED}}(1, m+1)$. Since I' corresponds to a and I'' corresponds to the complement of a, we have the $I' \cup I'' = I$. Therefore, there cannot be a 1-defeater for q: every fact of I that is not in I' is in I'', and both I' and I'' support q. □

7.3 Discussion

We have explored several notions of approximation for inconsistency-tolerant semantics in data exchange. For ontology-based data access scenarios, the approx-

imations examined here are known to be tractable for certain well-studied classes of constraints (e.g., *DL-Lite* ontology languages). However, our results show that the same approximations are intractable, and in some cases even computationally harder than XR-certain query answering, for the classes of constraints typically studied in data exchange. Furthermore, we saw in Section 7.1 that there is a critical difference between ISR-certain semantics and IXR-certain semantics, in that the former is equivalent to XR-certain semantics only under higly restrictive syntactic conditions, while the latter is equivalent to XR-certain semantics under only modest syntactic restrictions. This result is complementary to the finding in Section 3.2 that every GLAV+(GLAV, EGD) schema mapping $\mathcal{M}$ and union of conjunctive queries q can be expressed as a GLAV+(GLAV, EGD) schema mapping $\mathcal{M}'$ and a projection-free atomic query q' (that is, both settings have the same XR-certain answers for all source instances I). Nonetheless, the results in Chapter 6 suggest that XR-certain may sometimes have acceptable computational behavior in practice, despite the apparent difficulty of approximating it.

Chapter 8

Concluding Remarks

In this work, we have explored semantics for query answering in data exchange despite the presence of target inconsistencies. Intuitively, inconsistencies arise when the target constraints expose existing inconsistent or conflicting information in the data sources. This observation motivated our approach of repairing the source instances, rather than the target. Through examples, we have shown that this approach, called XR-certain query answering, gives non-trivial, meaningful answers to target queries, and that these answers compare favorably against both naıve approaches and other proposals in the literature.

In Chapter 3, we proved that XR-certain query answering is coNP-complete in data complexity. Chapters 4 through 6 explored several aspects of computing XR-certain in light of its computational complexity. For a broad class of schema mappings (those that are weakly acyclic), we have given a precise translation to disjunctive logic programming, and studied it both in theory (including via a reference implementation) and in practice (via an optimized implementation) for a non-trivial scientific data exchange scenario. Our results for the optimized implementation are encouraging. Nonetheless, we pursued approximation semantics in Chapter 7. Those results, however, are mostly negative. We find that approxima-

tions known to be tractable for the classes of constraints typically considered in ontology-based data access turn out to be intractable (that is, to be at least NP- or coNP-hard) for the classes of constraints typically considered in data exchange. This means they are no better, complexity-wise, than XR-certain semantics.

Some areas lend themselves to further exploration. There may be reductions to other formalisms for which high-quality solvers exist. There may also be better optimizations, or better approximation semantics. Moreover, we have pursued neither the materialization of target instances nor the interaction of aggregate queries with our proposed semantics; both areas are deserving of study. Some concrete problems also remain open. One of these is the data complexity of k-supp-XR-certain semantics when k is fixed (our lower bound result applies only to the uniform case). Another open problem is whether there is a concise reduction from XR-certain query answering to (the complement of) satifiability - since the former is coNP-complete, a polynomial time reduction must exist, but such a reduction may or may not be efficient enough for practical purposes.

Ultimately, we are hopeful that this thesis will lay the groundwork for such future research, and that our proposed semantics might become the de facto standard for inconsistency-tolerant data exchange.

Bibliography

[1] *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*. IEEE Computer Society, 1999.

[2] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*, volume 29. 1995.

[3] Foto N. Afrati and Phokion G. Kolaitis. Repair checking in inconsistent databases: algorithms and complexity. In Fagin [31], pages 31–41.

[4] Troels Andreasen, Amihai Motro, Henning Christiansen, and Henrik Legind Larsen, editors. *Flexible Query Answering Systems, 5th International Conference, FQAS 2002, Copenhagen, Denmark, October 27-29, 2002, Proceedings*, volume 2522 of *Lecture Notes in Computer Science*. Springer, 2002.

[5] Marcelo Arenas, Pablo Barceló, Leonid Libkin, and Filip Murlak. *Relational and XML Data Exchange*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.

[6] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In Vianu and Papadimitriou [77], pages 68–79.

[7] Marcelo Arenas, Ronald Fagin, and Alan Nash. Composition with target constraints. *Logical Methods in Computer Science*, 7(3), 2011.

[8] Marcelo Arenas, Jorge Pérez, Juan L. Reutter, and Cristian Riveros. The language of plain so-tgds: Composition, inversion and structural properties. *J. Comput. Syst. Sci.*, 79(6):763–784, 2013.

[9] Leopoldo E. Bertossi. *Database Repairing and Consistent Query Answering*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[10] Leopoldo E. Bertossi, Jan Chomicki, Alvaro Cortés-Calabuig, and Claudio Gutiérrez. Consistent answers from integrated data sources. In Andreasen et al. [4], pages 71–85.

[11] Meghyn Bienvenu. On the complexity of consistent query answering in the presence of simple ontologies. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.

[12] Meghyn Bienvenu, Camille Bourgaux, and François Goasdoué. Querying inconsistent description logic knowledge bases under preferred repair semantics. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 996–1002, 2014.

[13] Meghyn Bienvenu and Riccardo Rosati. Tractable approximations of consistent query answering for robust ontology-based data access. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, 2013.

[14] Meghyn Bienvenu and Riccardo Rosati. Tractable approximations of consistent query answering for robust ontology-based data access. In Rossi [70].

[15] Loreto Bravo and Leopoldo E. Bertossi. Logic programs for consistently querying data integration systems. In Gottlob and Walsh [41], pages 10–15.

[16] Andrea Calì, Marco Console, and Riccardo Frosini. Deep separability of ontological constraints. *CoRR*, abs/1312.5914, 2013.

[17] Andrea Calì, Domenico Lembo, and Riccardo Rosati. Query rewriting and answering under constraints in data integration systems. In Gottlob and Walsh [41], pages 16–21.

[18] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati. Ontology-based database access. In *Proceedings of the Fifteenth Italian Symposium on Advanced Database Systems, SEBD 2007, 17-20 June 2007, Torre Canne, Fasano, BR, Italy*, pages 324–331, 2007.

[19] Balder ten Cate, Richard L. Halpert, and Phokion G. Kolaitis. Exchange-repairs: Managing inconsistency in data exchange. In *RR*, volume 8741 of *Lecture Notes in Computer Science*, pages 140–156. Springer, 2014.

[20] Laura Chiticariu and Wang Chiew Tan. Debugging schema mappings with routes. In Dayal et al. [24], pages 79–90.

[21] Jan Chomicki and Jerzy Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.*, 197(1-2):90–121, 2005.

[22] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

[23] Isabel F. Cruz, Elena Ferrari, Yufei Tao, Elisa Bertino, and Goce Trajcevski, editors. *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*. IEEE Computer Society, 2014.

[24] Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim, editors. *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*. ACM, 2006.

[25] James P. Delgrande and Wolfgang Faber, editors. *Logic Programming and Nonmonotonic Reasoning - 11th International Conference, LPNMR 2011, Vancouver, Canada, May 16-19, 2011. Proceedings*, volume 6645 of *Lecture Notes in Computer Science*. Springer, 2011.

[26] Alin Deutsch, editor. *15th International Conference on Database Theory, ICDT '12, Berlin, Germany, March 26-29, 2012*. ACM, 2012.

[27] Oliver M. Duschka and Michael R. Genesereth. Answering recursive queries using views. In Mendelzon and Özsoyoglu [62], pages 109–116.

[28] Balder ten Cate, Gaëlle Fontaine, and Phokion G. Kolaitis. On the data complexity of consistent query answering. In Deutsch [26], pages 22–33.

[29] Thomas Eiter, Michael Fink, Gianluigi Greco, and Domenico Lembo. Repair localization for query answering from inconsistent databases. *ACM Trans. Database Syst.*, 33(2), 2008.

[30] Thomas Eiter, Georg Gottlob, and Heikki Mannila. Adding disjunction to datalog. In Vianu [76], pages 267–278.

[31] Ronald Fagin, editor. *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, volume 361 of *ACM International Conference Proceeding Series*. ACM, 2009.

[32] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[33] Ronald Fagin, Phokion G. Kolaitis, Alan Nash, and Lucian Popa. Towards a theory of schema-mapping optimization. In Lenzerini and Lembo [56], pages 33–42.

[34] Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang Chiew Tan. Composing schema mappings: Second-order dependencies to the rescue. *ACM Trans. Database Syst.*, 30(4):994–1055, 2005.

[35] Onofrio Febbraro, Kristian Reale, and Francesco Ricca. ASPIDE: integrated development environment for answer set programming. In Delgrande and Faber [25], pages 317–330.

[36] Ariel Fuxman and Renée J. Miller. First-order query rewriting for inconsistent databases. *J. Comput. Syst. Sci.*, 73(4):610–635, 2007.

[37] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and M. Schneider. Potassco: The Potsdam answer set solving collection. *AI Communications*, 24(2):107–124, 2011.

[38] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. Mapping and cleaning. In Cruz et al. [23], pages 232–243.

[39] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Kowalski and Bowen [49], pages 1070–1080.

[40] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9(3/4):365–386, 1991.

[41] Georg Gottlob and Toby Walsh, editors. *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*. Morgan Kaufmann, 2003.

[42] Gösta Grahne and Adrian Onet. Data correspondence, exchange and repair. In Segoufin [71], pages 219–230.

[43] Pascal Hitzler and Thomas Lukasiewicz, editors. *Web Reasoning and Rule Systems - Fourth International Conference, RR 2010, Bressanone/Brixen, Italy, September 22-24, 2010. Proceedings*, volume 6333 of *Lecture Notes in Computer Science*. Springer, 2010.

[44] Fan Hsu, W. James Kent, Hiram Clawson, Robert M. Kuhn, Mark Diekhans, and David Haussler. The UCSC known genes. *Bioinformatics*, 22(9):1036–1046, 2006.

[45] Richard M. Karp. Reducibility among combinatorial problems. In Miller and Thatcher [63], pages 85–103.

[46] Phokion G. Kolaitis, Jonathan Panttaja, and Wang Chiew Tan. The complexity of data exchange. In Vansummeren [75], pages 30–39.

[47] Phokion G. Kolaitis, Enela Pema, and Wang-Chiew Tan. Efficient querying of inconsistent databases with binary integer programming. *PVLDB*, 6(6):397–408, 2013.

[48] Paraschos Koutris and Jef Wijsen. The data complexity of consistent query answering for self-join-free conjunctive queries under primary key constraints. In Milo and Calvanese [64], pages 17–29.

[49] Robert A. Kowalski and Kenneth A. Bowen, editors. *Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, August 15-19, 1988 (2 Volumes)*. MIT Press, 1988.

[50] Markus Krötzsch and Sebastian Rudolph. Extending decidable existential rules by joining acyclicity and guardedness. In Walsh [78], pages 963–968.

[51] Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Source inconsistency and incompleteness in data integration. In *KRDB*, 2002.

[52] Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. Inconsistency-tolerant semantics for description logics. In *Web Reasoning and Rule Systems - Fourth International Conference, RR 2010, Bressanone/Brixen, Italy, September 22-24, 2010. Proceedings*, pages 103–117, 2010.

[53] Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. Inconsistency-tolerant semantics for description logics. In Hitzler and Lukasiewicz [43], pages 103–117.

[54] Domenico Lembo and Marco Ruzzi. Consistent query answering over description logic ontologies. In Marchiori et al. [59], pages 194–208.

[55] Maurizio Lenzerini. Data integration: A theoretical perspective. In Popa et al. [68], pages 233–246.

[56] Maurizio Lenzerini and Domenico Lembo, editors. *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, 2008.

[57] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Log.*, 7(3):499–562, 2006.

[58] Thomas Lukasiewicz, Maria Vanina Martinez, Andreas Pieris, and Gerardo I. Simari. From classical to consistent query answering under existential rules. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 1546–1552, 2015.

[59] Massimo Marchiori, Jeff Z. Pan, and Christian de Sainte Marie, editors. *Web Reasoning and Rule Systems, First International Conference, RR 2007, Innsbruck , Austria, June 7-8, 2007, Proceedings*, volume 4524 of *Lecture Notes in Computer Science*. Springer, 2007.

[60] Bruno Marnette. Generalized schema-mappings: from termination to tractability. In Paredaens and Su [67], pages 13–22.

[61] Bruno Marnette. Resolution and datalog rewriting under value invention and equality constraints. *CoRR*, abs/1212.0254, 2012.

[62] Alberto O. Mendelzon and Z. Meral Özsoyoglu, editors. *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 12-14, 1997, Tucson, Arizona, USA*. ACM Press, 1997.

[63] Raymond E. Miller and James W. Thatcher, editors. *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York*, The IBM Research Symposia Series. Plenum Press, New York, 1972.

[64] Tova Milo and Diego Calvanese, editors. *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015*. ACM, 2015.

[65] National Center for Biotechnology Information. NCBI ToolBox, 2001. `http://www.ncbi.nlm.nih.gov/IEB/ToolBox/`.

[66] National Center for Biotechnology Information. Entrez Programming Utilities, 2013. `http://www.ncbi.nlm.nih.gov/books/NBK25501/`.

[67] Jan Paredaens and Jianwen Su, editors. *Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June 19 - July 1, 2009, Providence, Rhode Island, USA*. ACM, 2009.

[68] Lucian Popa, Serge Abiteboul, and Phokion G. Kolaitis, editors. *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*. ACM, 2002.

[69] Riccardo Rosati. On the complexity of dealing with inconsistency in description logic ontologies. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 1057–1062, 2011.

[70] Francesca Rossi, editor. *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. IJCAI/AAAI, 2013.

[71] Luc Segoufin, editor. *Database Theory - ICDT 2010, 13th International Conference, Lausanne, Switzerland, March 23-25, 2010, Proceedings*, ACM International Conference Proceeding Series. ACM, 2010.

[72] Balder ten Cate, Gaëlle Fontaine, and Phokion G. Kolaitis. On the data complexity of consistent query answering. *Theory Comput. Syst.*, 57(4):843–891, 2015.

[73] Balder ten Cate, Richard L. Halpert, and Phokion G. Kolaitis. Exchange-repairs - managing inconsistency in data exchange. *J. Data Semantics*, 5(2):77–97, 2016.

[74] Christopher Umans. Hardness of approximating sigma$_2^{\mathrm{p}}$ minimization problems. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA* [1], pages 465–474.

[75] Stijn Vansummeren, editor. *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*. ACM, 2006.

[76] Victor Vianu, editor. *Proceedings of the Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 24-26, 1994, Minneapolis, Minnesota, USA*. ACM Press, 1994.

[77] Victor Vianu and Christos H. Papadimitriou, editors. *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania, USA*. ACM Press, 1999.

[78] Toby Walsh, editor. *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*. IJCAI/AAAI, 2011.

Index