FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Foundational Aspects of Schema Mapping Optimization and Normalization

DISSERTATION

zur Erlangung des akademischen Grades

Doktor/in der technischen Wissenschaften

eingereicht von

Vadim Savenkov
Matrikelnummer 0627305

an der
Fakultät für Informatik der Technischen Universität Wien
Betreuung: Prof. Dr. Reinhard Pichler

Diese Dissertation haben begutachtet:

(Prof. Dr. Reinhard Pichler) (Prof. Dr. Nicole Schweikardt)

Wien, 05.07.2012

(Vadim Savenkov)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Foundational Aspects of Schema Mapping Optimization and Normalization

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor/in der technischen Wissenschaften

by

Vadim Savenkov

Registration Number 0627305

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Prof. Dr. Reinhard Pichler

The dissertation has been reviewed by:

(Prof. Dr. Reinhard Pichler) (Prof. Dr. Nicole Schweikardt)

Wien, 05.07.2012

(Vadim Savenkov)

Erklärung zur Verfassung der Arbeit

Vadim Savenkov
Kölblgasse 15/2/13, 1030 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschliesumlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Acknowledgements

I would like to express my deep gratitude to my advisor, Prof. Reinhard Pichler, for his enduring support and encouragement throughout the work on this thesis. Most of my research skills have been acquired through joint work with him. Just as importantly, I had to completely rethink the concept of *having patience* after observing him in various situations over the past years.

I would like to thank the co-authors of the publications in which the results presented in this thesis have first appeared. I thank Prof. Georg Gottlob for insightful discussions regarding the material in the first part of this thesis. It was great fun to work with Emanuel Sallinger on the relaxed notions of equivalence, and on many other things. Ingo Feinerer has driven the work on the equivalence of Second-Order tgds, which was an exciting experience too.

I am also thankful to Sebastian Skritek, who has been my friendly roommate for quite some time, has corrected quite a few (well, a lot of) mistakes in my German, and has proofread quite a few of my writings, including this thesis. Many further colleagues and friends in Vienna, in Moscow, and meanwhile in many other places all over the world have helped me, and enriched my life in one way or another.

Last but by no means least, I would like to thank my family. There is not a single day when I don't think of my parents, Lidia and Anatoly, even if I don't manage to call them as often. My dear wife Aleksandra has never ceased to give me her love and support, no matter what I was doing. I am grateful to her for that, and for doing so many things in her unique way and no single thing halfheartedly, and of course for our two boys, who now also give their love to me and keep inspiring me all the time.

This thesis has been supported by the Erasmus Mundus External Cooperation Window Programme of the European Union and by the Vienna Science and Technology Fund (WWTF) through project ICT08-032.

Abstract

Schema mappings represent a basic concept of data integration and data exchange, expressing the relationship between database schemas. They can be seen as declarative programs, transforming the data from one schema to another, or rewriting a query against one schema as a query against another schema. A typical situation is when syntactically different mappings have the same effect with respect to a given task in information integration. However, syntactical differences may have dramatic effects on the consumption of computational resources, which are required for solving the task.

Considering mappings leading to the same end result for the task at hand as equivalent, one can formulate the *semantic optimization* problem as follows: among multiple equivalent mappings find those that yield the least resource consumption. The first step towards the goal of schema mapping optimization has been made by Fagin et al. in 2008 by introducing three basic notions of schema mapping equivalence and studying their properties. This dissertation extends the theory of schema mapping optimization in several directions.

The first part of this dissertation deals with the question of optimization and normalization of schema mappings with respect to the standard notion of logical equivalence. We formulate the concrete optimality criteria for schema mappings given by embedded dependencies, and define a system of rewrite rules, transforming a mapping given by a set of source-to-target dependencies into an optimal one. We also prove that the result of applying our rewrite rule system is unique up to renaming of variables. Moreover, we extend this result by defining a unique normal form in the presence of target equality generating dependencies and show the trade-off between uniqueness and optimality in this setting.

In the second part of the dissertation, we move on to the relaxed notions of equivalence, proposed by Fagin et al.: namely, to data exchange equivalence and to conjunctive query equivalence. If no integrity constraints are defined over the target schema, these notions are known to coincide with logical equivalence. We show that this result holds even if the target schema includes integrity constraints, under the assumption that the schema and the constraints are fixed. If the target constraints are taken as part of the mapping and are allowed to vary, the relaxed notions of equivalence are known to become undecidable. For conjunctive query equivalence, this holds even if the target constraints are restricted to primary keys. We separate the two relaxed notions of equivalence by identifying a practically relevant class of target integrity constraints

(containing functional and inclusion dependencies), for which data exchange equivalence is decidable and offers more optimization potential than logical equivalence.

Finally, we consider testing conjunctive query equivalence for mappings based on Second-Order tuple generating dependencies and show undecidability of this task, under a realistic assumption that primary keys are defined over the source schema.

Kurzfassung

Schemaabbildungen (“schema mappings”) sind ein grundlegendes Konzept in Datenintegration (“data integration”) und Datenaustausch (“data exchange”), welches die Beziehung zwischen zwei Datenbankschemas beschreibt. Schemaabbildungen können als deklarative Programme verstanden werden, die entweder Daten von einem Schema in ein anderes transferieren, oder eine Abfrage über einem Schema in eine Abfrage über einem anderem Schema übersetzen. Dabei tritt häufig die Situation auf, dass syntaktisch verschiedene Abbildungen denselben Effekt bezüglich einer bestimmten Aufgabe im Bereich der Informationsintegration (“information integration”) haben. Solche Abbildungen werden als äquivalent (bezüglich dieser Aufgabe) bezeichnet. Ein wichtiger Aspekt dabei ist, dass syntaktische Unterschiede zwischen äquivalenten Abbildungen eine große Auswirkung auf die Menge der benötigten Ressourcen (Zeit und Speicher) haben können.

Auf Grund dessen ist das Problem der “semantischen Optimierung” wie folgt definiert: Gegeben eine Schemaabbildung, finde jene äquivalente Abbildung, welche es erlaubt, die gestellte Aufgabe mit dem geringsten Ressourcenverbrauch zu lösen.

Der erste Schritt in Richtung Abbildungsoptimierung wurde von Fagin et al. 2008 gesetzt, indem sie drei Äquivalenzbegriffe definierten und deren grundlegende theoretische Eigenschaften untersuchten. Die vorliegende Arbeit erweitert die Theorie der Optimierung von Schemaabbildungen in mehrere Richtungen.

Der erste Teil dieser Arbeit beschäftigt sich mit der Frage der Optimierung und Normalisierung von Schemaabbildungen bezüglich logischer Äquivalenz, dem Standardbegriff von Äquivalenz. Für Schemaabbildungen, definiert durch sogenannte “embedded dependencies”, formulieren wir konkrete Optimalitätskriterien und definieren ein System von Transformationsregeln, das eine Abbildung (gegeben als sogenannte Quelle-zu-Ziel (“source-to-target”) Beziehungen) in eine optimale Darstellung überführt. Weiters beweisen wir, dass das Ergebnis der Anwendung dieser Transformationsregeln eindeutig ist.

Darüber hinaus erweitern wir dieses Resultat durch die Definition einer Normalform von Schemaabbildungen, welche zusätzlich Gleichheit erzeugende Abhängigkeiten (“equality generating dependencies”) auf dem Ziel-Schema erlauben, und zeigen die Unvereinbarkeit von Eindeutigkeit und Optimalität in dieser Situation.

Der zweite Teil der Arbeit betrachtet weniger strenge (gelockerte) Äquivalenzbegriffe, wie sie von Fagin et al. definiert wurden, nämlich Datenaustauschäquivalenz (“data exchange equiva-

lence"; DE-Äquivalenz) und Äquivalenz bezüglich konjunktiver Abfragen ("conjunctive query equivalence"; CQ-Äquivalenz).

Es ist bekannt, dass diese Begriffe mit logischer Äquivalenz zusammenfallen wenn keine Integritätsbedingungen auf dem Ziel-Schema definiert sind. Wir zeigen, dass dies auch der Fall ist wenn auf dem Ziel-Schema Integritätsbedingungen definiert sind, vorausgesetzt dass das Ziel-Schema und die Bedingungen fixiert sind. Weiters ist bekannt, dass die gelockerten Äquivalenzbegriffe im Allgemeinen unentscheidbar sind, wenn diese Ziel-Bedingungen ("target constraints") nicht fixiert sind. Für CQ-Äquivalenz gilt dies sogar für den Fall dass diese Bedingungen nur Schlüssel sind.

Wir identifizieren eine praktisch relevante Klasse von Ziel-Bedingungen, die sowohl die Funktionalen- als auch Inklusionsabhängigkeiten beinhaltet. Für diese ist DE-Äquivalenz entscheidbar, und darüber hinaus bietet sie zusätzliches Optimierungspotential gegenüber logischer Äquivalenz. Abschließend betrachten wir das Problem SO-Abbildungen ("SO-tgds") auf CQ-Äquivalenz zu überprüfen. Wir zeigen, dass dieses Problem unter der realistischen Annahme, dass das Quell-Schema Schlüssel besitzt, unentscheidbar ist.

Contents

CHAPTER 1

Introduction

Schema mappings are high-level specifications that describe the relationship between two database schemas. They play a key role in data integration [32,38] and data exchange [22]. A schema mapping is usually given in the form $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$, indicating the two database schemas $\mathbf{S}$ and $\mathbf{T}$ plus a set Σ of dependencies. These dependencies express constraints that instances of $\mathbf{S}$ and $\mathbf{T}$ must fulfil. In data exchange, $\mathbf{S}$ and $\mathbf{T}$ are referred to as source and target schema. The dependencies in Σ specify, given a source instance (i.e., an instance of $\mathbf{S}$), what a legal target instance (i.e., an instance of $\mathbf{T}$) may look like. Similarly, in data integration, a schema mapping $\mathcal{M}$ describes the relationship between a local data source and a global mediated schema.

Over the past decade, schema mappings have been extensively studied (see [11, 37] for numerous pointers to the literature). However, the question of *schema mapping optimization* has been posed only in 2008 by Fagin et al. [23]. In that work, the authors laid the foundations for schema mapping optimization by defining several forms of equivalence of schema mappings and by proving important properties of the resulting notions. The goal of this thesis is to extend the theory of schema mapping optimization in several directions.

On the one hand, we define a number of optimization criteria and show that under the standard notion of logical equivalence mappings based on source-to-target tgds admit a *unique normal form* which satisfies all these optimization criteria. We also show that even slight increase in complexity of schema mappings leads to a trade-off between uniqueness of the normal form and optimality. These results, published as [31], constitute the first part of this thesis.

On the other hand, we focus on the relaxed notions of equivalence. The main motivation for introducing alternative notions of equivalence in [23] was the fact that logical equivalence is often too restrictive and does not properly reflects the applications of schema mappings. This point is illustrated by examples later on in this chapter (see Section 1.1). Relaxed notions of equivalence address this issue, but are subject to the following paradoxical situation: Already in [23] it has been shown that for relatively simple mappings (e.g., those without target constraints) relaxed notions of equivalence *coincide with the logical equivalence*, that is, bring no additional optimization power. A slight increase of expressiveness of schema mappings, however, immediately leads to *undecidability of the equivalence testing problem*. The second part

of this thesis is dedicated to the exploration of the decidability boundary for alternative notions of equivalence and specifically focuses on two results: a positive one, establishing the notion of *Data Exchange equivalence* as a practical tool for optimization of a broad class of schema mappings, and a negative one, concerned with undecidability of testing the *Conjunctive Query equivalence* for mappings based on Second-Order tgds. These and a number of other results (surveyed in Chapter 6) have been published in [27, 49].

We continue our introduction with a series of motivational examples for schema mapping optimization under various notions of equivalence.

1.1 Motivational examples

Fundamental notions

A few concepts of Data Exchange are needed to motivate the problems studied in this thesis. The Data Exchange Problem associated with a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is the following one: Given an instance I of $\mathbf{S}$, find an instance J of $\mathbf{T}$ such that the combined instance $\langle I, J\rangle$ satisfies Σ (By *combined instance* $\langle I, J\rangle$ we mean simply the union $I \cup J$). Suppose that Σ is a set of *embedded dependencies* [19], that is, the sentences of the form

$$(\forall \overline{x})\,(\varphi(\overline{x}) \rightarrow (\exists \overline{y})\;\psi(\overline{x}, \overline{y}))\,,$$

where $\varphi(\overline{x})$ is a conjunction of atoms and $\psi(\overline{x}, \overline{y})$ is either a conjunction of atoms or an equality $x_i = x_j$ for some variables x_i, x_j from $\overline{x}$. In the former case, the sentence is called *tuple generating dependency*, or *tgd* for short, and in the latter case *equality generating dependency* abbreviated as *egd*. The universal quantification is usually not denoted explicitly. Instead, it is assumed implicitly for all variables in $\varphi(\overline{x}\,)$. The set of dependencies Σ in a schema mapping can be typically subdivided into the disjoint sets $\Sigma_s \cup \Sigma_{st} \cup \Sigma_t$, defined as follows:

- *Source-to-target dependencies* Σ_{st} have the antecedent of the implication φ over the source schema $\mathbf{S}$ and the conclusion of the implication over the target schema $\mathbf{T}$. They specify the relations between two schemas and consist of tgds, called *s-t tgds* for short.
- *Target dependencies* Σ_t express integrity constraints over the target schema.
- *Source dependencies* Σ_s can be used to restrict admissible source instances. We will only consider *source egds* in this thesis, specifically, in Chapters 4 and 8.

For mappings given by sets of embedded dependencies, the Data Exchange Problem can be solved by a natural procedure called *chase* [7, 22]: given a source instance I_S and a set of embedded dependencies Σ, we initialize an instance I over $\langle \mathbf{S}, \mathbf{T}\rangle$ with I_S. Then, for each dependency $(\forall \overline{x})\varphi(\overline{x}) \rightarrow (\exists \overline{y})\;\psi(\overline{x}, \overline{y})$ in Σ and for each assignment $\overline{a}$ for $\overline{x}$ such that the atoms in $\varphi(\overline{a})$ are contained in I (in which case we informally say that the dependency "fires" for an assignment $\overline{a}$), I is updated as follows:

> **Tgd:** If $\psi(\overline{x}, \overline{y})$ is a conjunction of atoms, then I is extended with the atoms of $\psi(\overline{a}, \overline{Z})$ where $\overline{Z}$ assigns to each variable in $\overline{y}$ a distinct labelled null not yet present in I.

Egd: If $\psi(\overline{x}, \overline{y})$ is an equality $x_i = x_j$, the respective values a_i, a_j are unified everywhere in I, unless a_i and a_j are distinct constants: then the chase aborts with failure.

The updating process repeats until all unifying assignments $\overline{a}$ have been tried, or a failure occurs. In the former case, the part of the instance over the schema $\mathbf{T}$ is a solution for the Data Exchange problem [22].

Example 1. *Consider a schema mapping $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$ with $\mathbf{S} = \{L(\cdot,\cdot,\cdot), P(\cdot,\cdot)\}$ and $\mathbf{T} = \{C(\cdot,\cdot), N(\cdot,\cdot)\}$, where L, P, C and N are abbreviations for the relational schemas* Lecture(title, year, prof), Prof(name, area), *and* Course (title, course-area), NetworkLab (room, reserved-for), *respectively. Suppose that Σ contains source-to-target dependencies expressing the following constraints: If any lecture is specified in the source instance, then the title of all lectures for 3rd year students as well as the area of the professor giving this lecture should be present in the* Course*-relation of the target instance. Moreover, Σ contains a specific rule which takes care of the lectures given by professors from the database area. We get the following tgds relating the schemas $\mathbf{S}$ and $\mathbf{T}$:*

$\tau_1 = L(x_1, x_2, x_3) \wedge L(x_4, 3, x_5) \wedge P(x_5, x_6) \rightarrow C(x_4, x_6)$

$\tau_2 = L(x_1, 3, x_2) \wedge P(x_2, \text{'db'}) \rightarrow C(x_1, \text{'db'})$

Moreover, let the following integrity constraints be defined over $\mathbf{T}$: A course in the database area requires a lab with a server access. However, there can be no more than one network lab assigned to a single course:

$\tau_t = C(x, \text{'db'}) \rightarrow (\exists y)\ N(y, x)$

$\epsilon_t = N(x_1, x) \wedge N(x_2, x) \rightarrow x_1 = x_2$

We will start with an input instance $I_S = \{L(\text{'distr-data'}, 3, \text{'bob'}), P(\text{'bob', 'db'})\}$ and apply the chase procedure to it:

1. *Initialize I:= I_S.*
2. *Apply τ_1 to I: extend I with a new fact $C(\text{'distr-data', 'db'})$.*
3. *Apply τ_2 to I: the fact $C(\text{'distr-data', 'db'})$ already exists in I, so proceed to the next step.*
4. *Apply τ_t to I: extend I with a new fact $N(R, \text{'distr-data'})$ with a fresh null R.*
5. *Apply ϵ_t to I: there is only one N-fact in I, so ϵ_t is trivially satisfied.*

   The execution terminates successfully.

The target instance constructed by the chase thus has the form $J = \{C(\text{'distr-data', 'db'}), N(R, \text{'distr-data'})\}$ where R is a labelled null. J is a solution for the Data Exchange Problem associated with Σ, for a given source instance I.

Several remarks are in order:

- Termination of the chase depends on the set of dependencies in Σ. Generally, for mappings given by sets of embedded dependencies, it is undecidable whether the chase terminates on a given instance or on every instance [15]. However, there are very large classes of

schema mappings for which the chase procedure always terminates [43, 46, 56]. In this thesis we only consider mappings given by dependencies which do not cause the chase to diverge.

- The target instances are allowed to contain *labelled nulls* (or simply *nulls*): these can be seen as placeholders for unknown values. (We are thus dealing with incomplete databases, or *v-tables* [35], also known as *naïve tables*). In contrast, we assume *source instances to be ground*, that is, to contain no nulls at all.
- There can be an infinite number of solutions.

Under this provision, the notion of *universal solution* is crucial: Consider a function h assigning values to labelled nulls (a value can be either a labelled null or a constant). Let us call such a function a null valuation. By a certain overloading of notation, given a null valuation h we write $h(J)$ to denote an instance resulting from replacing each null x in J by $h(x)$. Now, a solution J is universal, if for any other solution J', there exists a null valuation h such that $h(J) \subseteq J'$ holds. In particular, the solution J in Example 1 is universal while the solution $J' = \{C(\text{'distr-data'}, \text{'db'}), N(\text{'room1'}, \text{'distr-data'})\}$ is not: no null valuation can turn J' into J.

In their fundamental paper on Data Exchange [22], Fagin et al. showed that if terminated without failure, the chase procedure computes a universal solution for a given Data Exchange Problem. Such a solution is called *canonical universal solution*. Universal solutions have been shown extremely useful for query answering in data exchange: Intuitively, they can be described as "the most general solutions possible". Notably, there can be infinitely many universal solutions, containing an arbitrary number of syntactically distinct tuples containing labelled nulls. Therefore, the notion of minimal universal solution is important: minimal solution is unique up to renaming of nulls and is called *the core universal solution*. More details on cores and universal solutions, as well as formal definitions can be found in Chapter 2.

Logical equivalence

Our goal in this section is to illustrate the basic ideas of schema mapping optimization by simple examples, where it is clear "at a glance" what the optimal form of the schema mappings should look like. In fact, one would expect that a human user designs these mappings in their optimal form right from the beginning. However, as more and more progress is made in the area of automatic generation and processing of schema mappings [10, 11] we shall have to deal with schema mappings of ever increasing complexity. The optimality of these automatically derived schema mappings is by no means guaranteed and schema mapping optimization will become a real necessity.

Example 2. *Consider a schema mapping* $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma_{st} \rangle$ *with the schemas* $\mathbf{S} = \{L(\cdot,\cdot,\cdot), P(\cdot,\cdot)\}$, $\mathbf{T} = \{C(\cdot,\cdot)\}$ *and the two source-to-target dependencies from Example 1:*

$L(x_1, x_2, x_3) \wedge L(x_4, 3, x_5) \wedge P(x_5, x_6) \rightarrow C(x_4, x_6)$

$L(x_1, 3, x_2) \wedge P(x_2, \text{'db'}) \rightarrow C(x_1, \text{'db'})$

The above mapping has a specific form called GAV (global-as-view) [38], i.e., we only have s-t tgds $\varphi(\overline{x}\,) \rightarrow A(\overline{x}\,)$, where the conclusion is a single atom $A(\overline{x}\,)$ without existentially quantified variables. In this special case, we see a close relationship of schema mappings with unions of conjunctive queries (UCQs). Indeed, given a source instance I over $\mathbf{S}$, the tuples which have to be present in any legal target instance J according to the above schema mapping $\mathcal{M}$ are precisely the tuples in the result of the following UCQ:

$ans(x_4, x_6)$:- $L(x_1, x_2, x_3) \wedge L(x_4, 3, x_5) \wedge P(x_5, x_6)$
$ans(x_1, \text{'db'})$:- $L(x_1, 3, x_2) \wedge P(x_2, \text{'db'})$.

The goal of UCQ-optimization is usually twofold [14, 53], namely to minimize the number of CQs and to minimize the number of atoms in each CQ. In the above UCQ, we would thus delete the second CQ and, moreover, eliminate the first atom from the body of the first CQ. In total, the above UCQ can be replaced by a single CQ $ans(x_4, x_6)$:- $L(x_4, 3, x_5) \wedge P(x_5, x_6)$.

We would thus naturally reduce the set Σ of two s-t tgds in Example 2 to the singleton
$\Sigma' = \{L(x_4, 3, x_5) \wedge P(x_5, x_6) \rightarrow C(x_4, x_6)\}$.

So far, the standard query optimization techniques were sufficient for mapping optimization. In the case of GAV mappings, this is generally true, as we point out in Section 1.2. However, as mentioned above, GAV mappings are only a special case of schema mappings given by s-t tgds which, in the general case, may have existentially quantified variables and conjunctions of atoms in the conclusion. As follows from the definition of chase, these existentially quantified variables cause unknown values (labelled nulls) to be generated in the target instance. Hence, as an additional optimization goal, we would like to minimize the number of existentially quantified variables in each s-t tgd. Moreover, we would now also like to minimize the number of atoms in the CQ of the conclusion.

Example 3. *We revisit Example 2 and consider a new mapping $\mathcal{M}$ in the reverse direction so to speak: Let $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$ with $\mathbf{S} = \{C(\cdot, \cdot)\}$ and $\mathbf{T} = \{L(\cdot, \cdot, \cdot),\ P(\cdot, \cdot)\}$ where L, P, and C are as before. Moreover, let Σ be defined as follows:*

$\Sigma = \{\ C(x_1, x_2)\ \rightarrow (\exists y_1, y_2, y_3, y_4)\ L(y_1, y_2, y_3) \wedge L(x_1, 3, y_4) \wedge P(y_4, x_2),$
$\quad C(x_1, \text{'db'}) \rightarrow (\exists y_1)\ L(x_1, 3, y_1) \wedge P(y_1, \text{'db'})\}$

Clearly, Σ is equivalent to the singleton

$\Sigma' = \{C(x_1, x_2) \rightarrow (\exists y_4) L(x_1, 3, y_4) \wedge P(y_4, x_2)\}$.

The above schema mapping corresponds to the special case of LAV (local-as-view) [38] with s-t tgds of the form $A(\overline{x}\,) \rightarrow \exists \overline{y}\ \psi(\overline{x}, \overline{y}\,)$, where the antecedent is a single atom $A(\overline{x}\,)$ and all variables in $A(\overline{x}\,)$ actually do occur in the conclusion. In the most general case (referred to as GLAV mappings), no restrictions are imposed on the CQs in the antecedent and conclusion nor on the variable occurrences. In order to formulate an optimality criterion for schema mappings with s-t tgds of this general form, the analogy with UCQs does not suffice. Indeed, the following example illustrates that we may get a highly unsatisfactory result if we just aim at the minimization of the number of s-t tgds and of the number of atoms inside each s-t tgd.

Example 4. *Let* $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$ *with* $\mathbf{S} = \{L(\cdot,\cdot,\cdot)\}$ *and* $\mathbf{T} = \{C(\cdot,\cdot), E(\cdot,\cdot)\}$ *where* L*, and* C *are as before and* E *denotes the schema* Equal-Year(course1, course2)*, i.e.,* E *contains pairs of courses designed for students in the same year. Moreover, let* Σ *be defined as follows:*

$$\Sigma = \{L(x_1, x_2, x_3) \rightarrow (\exists y) C(x_1, y),$$
$$L(x_1, x_2, x_3) \wedge L(x_4, x_2, x_5) \rightarrow E(x_1, x_4)\}$$

Then Σ *is equivalent to the singleton* Σ' *with the tgd*

$$L(x_1, x_2, x_3) \wedge L(x_4, x_2, x_5) \rightarrow (\exists y) C(x_1, y) \wedge E(x_1, x_4)$$

Now suppose that the title*-attribute is a key in* Lecture*. Let* l_i *denote the title of some lecture in a source instance* I *and suppose that* I *contains* m *lectures for students in the same year as* l_i*. Then the computation of the* canonical universal solution *by the* chase *procedure yields two results of significantly different quality depending on whether we take* Σ *or* Σ'*: In case of* Σ*, we get one tuple* $C(l_i, y)$ *with this course title* l_i*. In contrast, for* Σ'*, we get* m *tuples* $C(l_i, y_1), \ldots, C(l_i, y_m)$ *with the same course title* l_i*. The reason for this is that the s-t tgd "fires" for every possible combination of key values* x_1 *and* x_4*, although for the conjunct* $C(x_1, y)$ *in the conclusion, only the value of* x_1 *is relevant.*

We shall refer to the two s-t tgds in Σ of the above example as the split form of the s-t tgd in Σ'. We shall formally define *splitting* of s-t tgds in Chapter 3. Intuitively, splitting aims at breaking up the conclusion of an s-t tgd in smaller parts such that the variables in the antecedent are indeed related to the atoms in the conclusion. Without this measure, any target instance would be artificially inflated with labelled nulls as we have seen with Σ' in the above example. Splitting helps to avoid such anomalies. Indeed, it can be seen as an analogous operation to the decomposition of relational schemas into normal form where we also want to exclude that some attributes are fully determined by parts of a key. Carrying over this idea to s-t tgds, we want to exclude that some atoms in the conclusion are fully determined by parts of the atoms in the antecedent. Our first *optimization goal* for schema mappings will therefore be to minimize the number of s-t tgds only to the extent that splitting should be applied whenever possible. Minimizing the size of each s-t tgd and the number of existentially quantified variables in the conclusion will, of course, be pursued as another optimization goal. We thus have the following optimality criteria for sets Σ of s-t tgds:

- *cardinality-minimality*, i.e., the number of s-t tgds in Σ shall be minimal;
- *antecedent-minimality*, i.e., the total size of the antecedents of the s-t tgds in Σ shall be minimal;
- *conclusion-minimality*, i.e., the total size of the conclusions of the s-t tgds in Σ shall be minimal;
- *variable-minimality*, i.e., the total number of existentially quantified variables in the conclusions shall be minimal.

We say that a set of s-t tgds is *optimal*, if it is minimal w.r.t. each of these four criteria. Following the above discussion, we only take s-t tgds into consideration for which no further splitting is possible. (We shall give a formal definition of this property and of the four optimality criteria in Chapter 3). Cardinality-minimality together with antecedent-minimality means that the *cost of the join-operations* is minimized when computing a canonical universal solution for some given source instance. Conclusion-minimality and variable-minimality mean that no unnecessary incomplete facts are introduced in the canonical universal solution. For the transformation of arbitrary sets of s-t tgds into optimal ones, we shall present a *novel system of rewrite rules*. Moreover, we shall show that the optimal form of a set of s-t tgds is *unique up to variable renaming*.

In other words, our optimization of schema mappings is also a *normalization of schema mappings*. As an immediate benefit of a normalization, we get a purely syntactical criterion for testing the equivalence of two schema mappings. Another, even more important application of such a normalization is in the area of defining the *semantics of query answering in data exchange*. Several definitions in this area depend on the concrete syntactic representation of the s-t tgds. This is, in particular, the case for queries with negated atoms (see e.g., [3, 39]) and for aggregate queries (see [1]). This semantic dependence on the syntax of a mapping clearly is undesirable. Since the minimal set of s-t tgds produced by our rewrite rules is unique up to variable renaming, we can use it as the desired normal form which eliminates the effect of the concrete representation of the s-t tgds from the semantics of query answering.

Example 5. *Consider a schema mapping $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$ with the source and target schemas $\mathbf{S} = \{S(\cdot,\cdot,\cdot)\}$ and $\mathbf{T} = \{L(\cdot,\cdot,\cdot), P(\cdot,\cdot)\}$, where L and P are as in Example 3, and S denotes the relational schema* Student(name, year, area). *Let Σ express the following constraints: If there exists a student in any year, then there should exist at least one lecture for this year. Moreover, if a student specializes in a particular area, then there should be a professor in this area teaching at least one lecture for this year. We thus have the following set Σ with a single s-t tgd:*

$$S(x_1, x_2, x_3) \rightarrow (\exists y_1, y_2, y_3, y_4, y_5)\ L(y_1, x_2, y_3) \wedge L(y_4, x_2, y_5) \wedge P(y_5, x_3)$$

Clearly, the first atom in the conclusion may be deleted.

Now consider the source instance $I = \{S(\text{'bob'}, 3, \text{'db'})\}$ and suppose that we want to evaluate the query

$$ans(x_2) \text{ :- } L(x_1, x_2, x_3), \neg P(x_3, x_4)$$

over the target instance, i.e., we want to check if, in some year, there exists a lecture which has not been assigned to a professor. In [3,39], query answering via the canonical universal solution is proposed. Depending on whether the s-t tgd in Σ has been simplified or not, we either get $J = \{L(u_1, 3, u_2),\ L(u_3, 3, u_4), P(u_4, \text{'db'})\}$ or the core thereof, $J' = \{L(u_1, 3, u_2), P(u_2, \text{'db'})\}$ as the canonical universal solution. In the first case, the query yields the result $\{\langle 3 \rangle\}$ whereas, in the second case, we get $\emptyset$.

Similarly, a unique normal form of the s-t tgds is crucial for the semantics of aggregate queries in data exchange, whose investigation has been initiated recently by Afrati and Kolaitis [1]. Aggregate queries are of the form SELECT f FROM R, where f is an aggregate operator $\mathsf{min}(R.A)$, $\mathsf{max}(R.A)$, $\mathsf{count}(R.A)$, $\mathsf{count}(*)$, $\mathsf{sum}(R.A)$, or $\mathsf{avg}(R.A)$, and where R is a target relation symbol or, more generally, a conjunctive query over the target schema and A is an attribute of R. On the one hand, [1] defines an interesting and non-trivial semantics of aggregate queries in data exchange. On the other hand, it is shown that the most important aggregate queries can be evaluated in polynomial time (data complexity). In Chapter 5 of this thesis, we shall show how aggregate queries can benefit from our normalization of schema mappings.

So far, we have only mentioned mappings $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$, where Σ is a set of s-t tgds. In addition, the target schema $\mathbf{T}$ may contain integrity constraints. For the sake of uniformity, we adopt the convention of taking the integrity constraints as a part of Σ. One of the most important forms of target constraints are egds, introduced in Section 1.1, which can be considered as a generalization of functional dependencies.

Example 6. *We modify the setting from Example 2 and 3. Let* $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$ *with* $\mathbf{S} = \{C(\cdot, \cdot, \cdot)\}$ *and* $\mathbf{T} = \{P(\cdot, \cdot, \cdot)\}$ *where* C *and* P *denote the relational schemas* Course (title, course-area, prof-area) *and* Prof(name, prof-area, course-area). *The* P*-relation thus contains information on the main research area of the professor as well as on the area(s) of the courses taught by him/her. The set* Σ *of s-t tgds expresses the following constraints: For every course, there exists a professor who teaches courses in his/her main area of expertise and who teaches courses with this combination of course- and prof-area. Moreover, there exists a professor whose expertise matches the area of the course and vice versa. We thus define* Σ *as a mapping with the following two s-t tgds:*

$C(x_1, x_2, x_3) \rightarrow (\exists y_1, y_2)\ P(y_1, y_2, y_2) \wedge P(y_1, x_2, x_3)$

$C(x_1, x_2, x_3) \rightarrow (\exists y_1) P(y_1, x_3, x_2)$

This set of dependencies is minimal. However, suppose that we add the egd

$P(x_1, x_2, x_3) \rightarrow x_2 = x_3,$

expressing that a professor only teaches courses in his/her own area of expertise. Then the atom $P(y_1, y_2, y_2)$ *can be eliminated from the conclusion of the first s-t tgd. Moreover, the first and the second s-t tgd imply each other. Hence,* Σ *can be replaced by either* Σ' *or* Σ'' *with*

$\Sigma' = \{C(x_1, x_2, x_3) \rightarrow (\exists y_1)\ P(y_1, x_2, x_3)\}$ *and*

$\Sigma'' = \{C(x_1, x_2, x_3) \rightarrow (\exists y_1)\ P(y_1, x_3, x_2)\}.$

Example 6 illustrates that, in the presence of target egds, our rewrite rules for the s-t tgds-only case are not powerful enough. To deal with target egds, we will introduce further rewrite rules. In particular, one of these new rewrite rules will result in the introduction of source egds to prevent situations where two sets of s-t tgds only differ on source instances which admit no target instance anyway. Indeed, in Example 6, Σ' and Σ'' only differ if $x_2 \neq x_3$ holds. But this cannot happen due to the egd over the target schema. Hence, Σ should be replaced by Σ^* with

$$\Sigma^* = \{C(x_1, x_2, x_3) \to x_2 = x_3,$$
$$C(x_1, x_2, x_2) \to (\exists y_1)\ P(y_1, x_2, x_2)\}.$$

In summary, we shall be able to prove that our extended set of rewrite rules again leads to a *normal form* which is *unique up to variable renaming*. The main ingredients of our normalization and optimization are the splitting and simplification of tgds. In the presence of target egds, several pitfalls will have to be avoided when defining appropriate splitting and simplification rules so as not to destroy the uniqueness of the normal form.

Alternative notions of equivalence

So far, we have been only dealing with *logical equivalence*, requiring that to be equivalent, the mappings $\mathcal{M}_1$ and $\mathcal{M}_2$ must be satisfied by precisely the same pairs $\langle I, J\rangle$ of source and target instances. This definition is often too strict for practical applications in information integration, as illustrated by the following example:

Example 7. *Consider the schema mapping $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma\rangle$ between the source schema $\mathbf{S} = \{C(\cdot,\cdot,\cdot)\}$ and the target schema $\mathbf{T} = \{P(\cdot,\cdot,\cdot), S(\cdot,\cdot,\cdot)\}$, which extend the schemas from Example 6 with the relation* Student *from Example 5. Let Σ contain a single dependency*

$\tau = C(x_1, x_2, x_2) \to (\exists y)P(y, x_2, x_2)$

which simply copies the information about the courses taught by professors whose area of expertise exactly matches the topic area of the course. Now, let the mapping $\mathcal{M}_1 = (\mathbf{S}, \mathbf{T}, \Sigma_1)$ extend $\mathcal{M}$ with a single target dependency

$\tau_1 = S(x_1, x_2, x_3) \to (\exists y_1, y_2)P(y_1, x_3, y_2)$,

stipulating that if a student specializes in some area, then there must be a professor active in this area as a researcher.

The mappings $\mathcal{M}$ and $\mathcal{M}_1$ are not logically equivalent. To see this, simply consider an instance $\langle I, J\rangle$, where I is empty and J consists of a single fact $S($'bob', 3, 'db'$)$. It is easy to check that both $\langle I, J\rangle \models \mathcal{M}$ and $\langle I, J\rangle \not\models \mathcal{M}_1$ holds.

We see that $\mathcal{M}_1$ and $\mathcal{M}_2$ are not logically equivalent, in spite of the fact that the target dependency in M_1 is defined over a relation which is not required to have any facts, no matter which source instance we consider. In particular, recall the chase procedure which computes an instance over $\mathbf{T}$ satisfying all dependencies in a given schema mapping. The chase would leave the Student relation empty, and thus the target dependency in $\mathcal{M}_1$ would be trivially satisfied. Thus, with respect to this specific application in data exchange, $\mathcal{M}$ and $\mathcal{M}_1$ are indistinguishable, which is however not captured by logical equivalence.

To address such situations, in their 2008 paper [23], Fagin et al. proposed two relaxed definitions of equivalence between the mappings $\mathcal{M}_1$ and $\mathcal{M}_2$:

- *Data exchange equivalence* requires that for every source instance I, the universal solutions under the mappings $\mathcal{M}_1$ and $\mathcal{M}_2$ coincide.

- *$\mathcal{L}$-equivalence* for a class of queries $\mathcal{L}$ requires, that for every source instance I, any query q from $\mathcal{L}$ posed against the target schema yields the same *certain answers* for the mappings $\mathcal{M}_1$ and $\mathcal{M}_2$. The notion of certain answer is central for information integration and refers to answers that hold in *each* solution to a given instance of Data Exchange Problem (see Section 2.6 and Chapter 5).

These two notions capture the immediate applications of schema mappings: In data exchange, universal solutions are recognized to be viable options for materialization [22, 39], so the notion of data exchange equivalence is specially tailored for applications which require universal solutions to be instantiated (An already mentioned example of such application is answering aggregate queries in data exchange, considered in Chapter 5). Since the chase is a tool for constructing universal solutions, the shortcoming of Example 7 can be eliminated by comparing the mappings relative to data exchange equivalence.

The motivation behind $\mathcal{L}$-equivalence is obvious, keeping in mind that answering queries (over transformed schemas) is the ultimate goal of information integration, and that the certain answers semantics is the one most generally agreed upon in the literature. Under the assumption that the class $\mathcal{L}$ of queries is known a priori, $\mathcal{L}$-equivalence allows to abstract away the properties of schema mappings which are not essential for query answering. The class of positive conjunctive queries (CQ) is of immense practical importance. Therefore, the notion of *conjunctive query equivalence* is one of the most remarkable members of the $\mathcal{L}$-equivalence family.

In total, in this thesis we will be dealing with three notions of equivalence, which are defined as follows: Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two schema mappings. We say that $\mathcal{M}_1$ and $\mathcal{M}_2$ are

- *logically equivalent* (denoted $\mathcal{M}_1 \equiv \mathcal{M}_2$) if $\mathcal{M}_1$ and $\mathcal{M}_2$ are satisfied by precisely the same pairs $\langle I, J\rangle$ of source and target instances.
- *DE-equivalent* (denoted $\mathcal{M}_1 \equiv_{DE} \mathcal{M}_2$) if, for every source instance I, the universal solutions under the mappings $\mathcal{M}_1$ and $\mathcal{M}_2$ coincide.
- *CQ-equivalent* (denoted $\mathcal{M}_1 \equiv_{CQ} \mathcal{M}_2$) if, for every source instance I, any conjunctive query posed against the target schema yields the same certain answers for the mappings $\mathcal{M}_1$ and $\mathcal{M}_2$.

In [23], the implications $(\mathcal{M}_1 \equiv \mathcal{M}_2) \Rightarrow (\mathcal{M}_1 \equiv_{DE} \mathcal{M}_2) \Rightarrow (\mathcal{M}_1 \equiv_{CQ} \mathcal{M}_2)$ were proved. In general, the converse of neither implication is true. The following example illustrates that for schema mappings consisting of embedded dependencies, the three notions of equivalence are indeed different and provide different power for detecting the redundancy of dependencies.

Example 8. *Consider the schema mappings $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma\rangle$, $\mathcal{M}_1 = \langle \mathbf{S}, \mathbf{T}, \Sigma_1\rangle$ and $\mathcal{M}_2 = \langle \mathbf{S}, \mathbf{T}, \Sigma_2\rangle$, where the schemas and the first two mappings are the same as in Example 7. Namely, we have three sets of dependencies $\Sigma = \{\tau\}$, $\Sigma_1 = \{\tau, \tau_1\}$, $\Sigma_2 = \{\tau, \tau_2\}$, where τ, τ_1 and τ_2 are defined as follows:*

$\tau = C(x_1, x_2, x_2) \rightarrow P(x_1, x_2, x_2)$

$\tau_1 = S(x_1, x_2, x_3) \rightarrow (\exists y_1, y_2) P(y_1, x_3, y_2)$
$\tau_2 = P(x_1, x_2, x_3) \rightarrow (\exists y) P(y, x_3, x_3)$

The former two dependencies have been introduced in Example 7. The latter one, τ_2 stipulates that if for some reason a professor teaches a course not in his immediate research area, there must also be an expert in the area in the faculty.

For $\mathcal{M}_1$, the equivalence $\mathcal{M}_1 \equiv_{DE} \mathcal{M}$ holds (and hence also $\mathcal{M}_1 \equiv_{CQ} \mathcal{M}$). Intuitively, this is due to the fact that τ_1 has no effect on the universal solutions. On the other hand, as explained in Example 7, we have $\mathcal{M}_1 \not\equiv \mathcal{M}$.

For $\mathcal{M}_2$, we have $\mathcal{M}_2 \equiv_{CQ} \mathcal{M}$ but $\mathcal{M}_2 \not\equiv_{DE} \mathcal{M}$ (and hence $\mathcal{M}_2 \not\equiv \mathcal{M}$).

The CQ-equivalence can be shown using its convenient characterization from [23], namely: The mappings $\mathcal{N}$ and $\mathcal{N}'$ are CQ-equivalent if, for every source instance I, either both $\mathcal{N}$ and $\mathcal{N}'$ have no solution or they both have the same core universal solutions. Recall that the core universal solution is up to renaming of nulls equal to any other minimal universal solution (See Chapter 2 for details).

We argue that for any source instance I, the smallest possible universal solution has a form $J_I^* = \{P(n_a, a, a) \mid (\exists c, b) C(c, a, b) \in C\}$, where n_a is a labelled null, distinct for each a.

Indeed, it is easy to see that such J_I^* is a solution for I under $\mathcal{M}$, since it satisfies τ. It is minimal, as removing any fact from J_I^* would lead to a violation of τ. Moreover, it is universal, since any solution for I under $\mathcal{M}$ resp. $\mathcal{M}_2$ must satisfy τ, and thus must contain a fact $P_a = P(x, a, a)$ for each value a occurring in a second position in C in I, with some arbitrary x. The corresponding atom $P(n_a, a, a) \in J_I^*$ can be turned into P_a by replacing n_a with x.

Now, observe that J_I^* satisfies τ_2 for a given source instance I. Thus, J_I^* is also a solution under $\mathcal{M}_2$. Moreover, we have shown that J_I^* is both minimal and universal and is thus the core universal solution under $\mathcal{M}_2$. Hence, $\mathcal{M}_2 \equiv_{CQ} \mathcal{M}$ holds.

To see the failure of DE-equivalence between $\mathcal{M}$ and $\mathcal{M}_2$, consider the instances $I = \{C(a, b, b)\}$ and $J = \{P(a, b, b), P(y_1, y_2, y_3)\}$ where y_1, y_2, y_3 are distinct nulls. J is a universal solution for I under mapping $\mathcal{M}$ but not a solution under $\mathcal{M}_2$, as $\langle I, J \rangle \not\models \tau_2$.

Second-Order tgds

Finally, we will consider source-to-target dependencies more expressive than sets of embedded dependencies. The motivation for introducing such language comes from the area of *schema mappings management*, which is concerned with transformations of schema mappings.

Several algebraic operators [9, 47] on schema mappings have been intensively studied in recent time like computing inverses [5,6,20,26] and composing schema mappings [10,25,41,48]. The composition operator is arguably a very basic one, defined purely set theoretically: Namely, the composition $\mathcal{M}_1 \circ \mathcal{M}_2$ of schema mappings $\mathcal{M}_1$ and $\mathcal{M}_2$ is the following set of instance pairs $\langle I, J \rangle$, where $\langle I, J \rangle = \{\langle I_1, I_3 \rangle \mid \exists I_2 : \langle I_1, I_2 \rangle \in \mathcal{M}_1 \text{ and } \langle I_2, I_3 \rangle \in \mathcal{M}_2\}$. Many other operators including, notably, the operators expressing inverse of mappings, are defined via the composition operator.

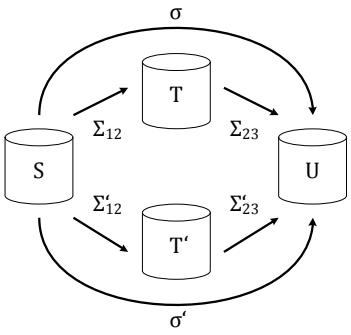

Figure 1.1: Mapping compositions.

Fagin et al. proved that, in general, s-t tgds are not powerful enough to express the composition of two mappings defined by s-t tgds [25]. To remedy this defect, so-called Second-Order tuple generating dependencies (SO tgds) were introduced in [25]. SO tgds extend s-t tgds by existentially quantified function-variables and equalities of (possibly functional) terms in the antecedents of implications. Details and formal definitions are given in Chapter 2. It was shown in [25] that SO tgds capture the closure under composition of mappings defined by s-t tgds.

Example 9 ([25])**.** *Consider the following three schemas. Let* $\mathbf{S}_1$ *consist of the unary relation symbol* $\mathsf{Emp}(\cdot)$ *of employees. Schema* $\mathbf{S}_2$ *consists of a single binary relation symbol* $\mathsf{Mgr}'(\cdot,\cdot)$ *that associates each employee with a manager. Schema* $\mathbf{S}_3$ *consists of a similar binary relation symbol* $\mathsf{Mgr}(\cdot,\cdot)$ *that is intended to provide a copy of* Mgr' *and an additional unary relation symbol* $\mathsf{SelfMgr}(\cdot)$ *to store employees who are their own manager.*

Consider the mappings $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ *and* $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$ *with*

$\Sigma_{12} = \{\forall e(\mathsf{Emp}(e) \rightarrow \exists m\, \mathsf{Mgr}'(e,m))\}$ and

$\Sigma_{23} = \{\forall e, m(\mathsf{Mgr}'(e,m) \rightarrow \mathsf{Mgr}(e,m)),\ \forall e(\mathsf{Mgr}'(e,e) \rightarrow \mathsf{SelfMgr}(e))\}$.

We are looking for the composition of $\mathcal{M}_{12}$ *and* $\mathcal{M}_{23}$*. It can be verified that this composition can be expressed by the SO tgd*

$$\sigma = \exists f\,(\forall e\,(\mathsf{Emp}(e) \rightarrow \mathsf{Mgr}(e, f(e))) \wedge \forall e\,(\mathsf{Emp}(e) \wedge (e = f(e)) \rightarrow \mathsf{SelfMgr}(e))).$$

The question of equivalence of mappings based on SO tgds naturally arises in several scenarios. Figure 1.1 illustrates a *model evolution* scenario, where data structured under some schema $\mathbf{S}$ is first migrated to a database with schema $\mathbf{T}$ and then further transformed to meet schema $\mathbf{U}$. Now suppose that there exists an alternative migration path from schema $\mathbf{S}$ via $\mathbf{T}'$ to schema $\mathbf{U}$. The question if the two migration paths yield the same result comes down to checking if the dependencies σ and σ' (which represent the respective mapping compositions) are equivalent. Actually, Figure 1.1 can also be thought of as illustrating a *peer data management* scenario, where some peer with data structured according to $\mathbf{S}$ provides part of its data to some other peer

with schema $\mathbf{T}$ (resp. $\mathbf{T}'$). The latter peer in turn passes this data on to yet another peer with schema $\mathbf{U}$. Now suppose that a user may access the data only at the peer with schema $\mathbf{U}$. What happens if some link in this peer data network is broken, say the one corresponding to mapping Σ_{23}? Will the path of mappings from $\mathbf{S}$ via $\mathbf{T}'$ to $\mathbf{U}$ still give the user full access to the data provided by the peer with schema $\mathbf{S}$? Testing the equivalence of σ and σ' is thus crucial for answering questions of redundancy and reliability in a peer data network.

1.2 Related work

In this thesis we are dealing with mappings between relational schemas. As already pointed out in the previous section, such mappings are typically represented as sets of embedded dependencies. Moreover, the source schema and the target schema may contain integrity constraints, which can, of course, also be expressed as embedded dependencies. Our goal in this section is thus to draw a distinction between the questions of optimization for the language of embedded dependencies and similar problems studied for similar logical languages.

Logical equivalence

In Section 1.1, we have pointed out the similarities between GAV mappings and UCQs. The well-studied methods of query optimization dealing with CQs [14] and UCQs [53] are not sufficient for LAV and GLAV mappings, let alone mappings with target constraints: see Examples 3 and 6. The same applies to the methods of Datalog optimization considered by Sagiv in [52] and to optimization of logic programs, studied relative to various semantics, including the stable model semantics [17, 18, 42]. Interestingly, Sagiv actually considers embedded dependencies in his work, but not as a subject for optimization: rather, the optimization of Datalog queries against schemas with integrity constraints is being discussed. Note that what Fagin et al. in [23] and we in this thesis call logical equivalence in the Datalog and Logic Programming worlds is known as *uniform equivalence* [42, 52].

Recently, the characteristic features of schema mapping languages have been actually added to Datalog, effectively eliminating the distinction between the two languages. The result of this extension is known as the Datalog$^{\pm}$ family of languages [12, 13]. The research on Datalog$^{\pm}$ so far has been focused on modelling ontology languages and algorithms for query answering and query containment in the situation when the bottom-up evaluation of the Datalog program (the chase procedure) does not terminate. Our work in this thesis is orthogonal to this line of research: We will be only dealing with mappings with the terminating chase property. Our results on schema mapping optimization and equivalence also apply to Datalog$^{\pm}$ programs, with an exception of Chapter 8, where the language of Second-Order tgds is considered.

The question of rewriting of schema mappings have been considered in the literature in the context of *core computation for data exchange*. In [45] and [57], the authors aim at the transformation of a set Σ of s-t tgds into an equivalent set Σ', s.t. chasing a source instance

with Σ' directly yields the core universal solution of the corresponding data exchange problem. In [44], such a transformation of s-t tgds is extended to mappings which comprise also functional dependencies as target dependencies. The transformations in [44, 45, 57] insert negated atoms and/or inequalities in the antecedents of some s-t tgds so as to block certain forms of applying these s-t tgds in the chase. The goal pursued by these transformations is to avoid the expensive core computation by post-processing of the canonical universal solution and to obtain the core directly as the chase result. Normalization and optimization of the mappings are not in the scope of those transformations.

Alternative notions of equivalence

In the area of schema mappings, notions of equivalence other than logical equivalence have been first considered by Fagin et al. in 2008 [23]. For query languages and logic programs such notions have been considered also prior to that. Actually, the standard notion of equivalence for Datalog programs coincides with CQ-equivalence in the terminology of Fagin et al. CQ-equivalence of Datalog programs is undecidable [55], like many optimization problems [28] related to this notion. These results can be directly used to demonstrate the hardness of equivalence testing and optimization of schema mappings with target tgds, with respect to CQ-equivalence: this observation has been made already in [23].

However, the applications of information integration yield for equivalence notions which have not yet been considered in the literature: So is *data exchange equivalence* [23] allowing mappings to differ on non-universal solutions. Since universal solutions have numerous important applications in information integration, this is a quite natural notion in this area.

No existing results on query optimization can be directly applied to data exchange equivalence. However, so far there was no clear understanding, for which situations DE-equivalence is preferable over, e.g. CQ-equivalence. More generally, the alternative notions of equivalence are subject to the following phenomenon: for simpler mappings (e.g., for mappings based on non-recursive tgds relating schemas without integrity constraints) the alternative notions coincide with logical equivalence [23], and thus bring no additional optimization potential. However, already a slight increase in expressiveness of mappings, such as allowing key constraints over the target schema, leads to the undecidability of testing for all equivalence notions but the most restrictive logical one. It is thus important to identify a class of mappings for which relaxed notions of equivalence are both decidable and more powerful than logical equivalence.

Second-Order tgds

No results concerning testing equivalence of Second-Order tgds have been published until the work [27], partly reflected upon in Chapter 8. In this chapter we show that the CQ-equivalence of mappings based on Second-Order tgds is undecidable, if the source schema allows for key dependencies. Without this assumption, however, the decidability of testing for CQ-equivalence remains an interesting open problem: On the one hand, logical equivalence of Second-Order tgds

is undecidable even for schemas without integrity constraints. On the other hand, in their recent paper [21], Fagin and Kolaitis have shown that testing CQ-equivalence between a Second-Order tgd and a set of embedded dependencies is decidable.

Summary

In spite of the fact that many notions of equivalence have been studied in the context of query languages and logic programming, the following problems have not been sufficiently treated in the literature and will be studied in this thesis:

- The language of schema mappings has features not present in traditional query and logic programming languages for which the optimization problem has been considered in the literature. This asks for a new formulation of optimization criteria and new optimization algorithms.
- The applications in data exchange make the question of normalization of schema mappings important. Is there a unique normal form? Is this form optimal?
- The alternative notions of equivalence, especially that of data exchange equivalence, raise a number of open problems. In particular, it is desirable to find classes of mappings for which the relaxed notions of equivalence offer more optimization potential than the standard notion of logical equivalence, and yet do not lead to undecidability of equivalence testing problem.
- Complexity (actually, even decidability) of testing equivalence of Second-Order dependencies is not known.

In the next section, we describe the structure of this thesis and outline our contributions towards closing the above gaps.

1.3 Organization of the thesis and summary of results

This thesis is organized in the following way: In Chapter 2, we recall some basic notions. A conclusion and an outlook to future work are given in Chapter 9. The rest of the thesis is divided in two parts.

Part I is dedicated to optimization and normalization of schema mappings under logical equivalence. The main results of the first part of the thesis are detailed in the Chapters 3–5, namely:

• *Optimization and normalization of sets of s-t tgds.* In Chapter 3, we give a formal definition of the above mentioned optimality criteria for sets of s-t tgds and we present rewrite rules to transform any set of s-t tgds into an optimal one (i.e., minimal w.r.t. to these criteria). We shall also show that the normal form obtained by our rewrite rules is unique up to variable renaming. Moreover, we show that, if the length of each s-t tgd is bounded by a constant, then this normal form can be computed in polynomial time.

• *Extension to target egds.* In Chapter 4, the rewrite rule system for s-t tgds is then extended to schema mappings comprising target egds. Several non-trivial extensions (like the introduction of source egds) are required to arrive at a unique normal form again. The extended splitting and simplification rules will have to be defined very carefully so as not destroy this uniqueness.

• *Semantics of aggregate operators.* In Chapter 5, we discuss in detail the application of our normalization of schema mappings to the definition of a unique semantics of aggregate operators in data exchange.

Part II of the thesis deals with alternative notions of schema mapping equivalence, namely with data exchange equivalence and conjunctive query equivalence.

• *Optimization of source-to-target dependencies in the presence of fixed target constraints under the relaxed notions of equivalence* is considered in Chapter 6, followed by an overview of undecidability results concerning testing relaxed notions of equivalence for various classes of mappings. We show that by no means can the practical applicability of DE- and CQ-equivalence be taken for granted: whereas for simple mappings the notions coincide with the logical equivalence, already a slight increase in expressiveness of the mappings results in the undecidable equivalence testing problem.

• *A class of mappings with decidable data exchange equivalence* is identified in Chapter 7. This class includes mappings with target Inclusion and Functional Dependencies, which makes DE-equivalence a practically relevant notion. This is in a sharp difference to CQ-equivalence, which, as pointed out in Chapter 6, is undecidable even for mappings with target Key Dependencies. The problem of testing DE-equivalence is shown to be efficiently solvable under realistic assumptions.

• *Undecidability of conjunctive query equivalence for mappings based on Second Order tgds.* In Chapter 8, we show that if Key Dependencies over the source schema are allowed, CQ-equivalence of SO tgds is undecidable. The proof uses the reduction from the Domino Problem [8]. As a by-product of this proof, we also get the undecidability of logical equivalence of SO tgds without equalities, also known as *plain SO tgds*.

CHAPTER 2

Preliminaries

A *schema* $\mathbf{R} = \{R_1, \ldots, R_n\}$ is a set of relation symbols R_i each of a fixed arity. An *instance* (or *database*) I over a schema $\mathbf{R}$ consists of a relation R_i^I for each relation symbol R_i in $\mathbf{R}$, such that both have the same arity. We only consider finite instances here.

Tuples of the relations (which we also call *facts*) may contain two types of *terms*: *constants* and *variables*. The latter are often also called *labelled nulls* or simply *nulls* for short. Two labelled nulls are equal if they have the same label. For every instance J, we write $dom(J)$, $var(J)$, and $Const(J)$ to denote the set of terms, variables, and constants, respectively, of J. Clearly, $dom(J) = var(J) \cup Const(J)$ and $var(J) \cap Const(J) = \emptyset$. If we have no particular instance J in mind, we write $Const$ to denote the set of all possible constants. We write $\overline{x}$ for a tuple $\langle x_1, x_2, \ldots, x_n \rangle$ and vice versa, denote by x_i an element at the i^{th} position in $\overline{x}$. By a certain abuse of notation, we also refer to the set $\{x_1, \ldots, x_n\}$ as $\overline{x}$. Hence, we may use expressions like $x_i \in \overline{x}$ or $\overline{x} \subseteq \overline{y}$, $\overline{x} \subseteq X$, etc.

Let $\mathbf{S} = \{S_1, \ldots, S_n\}$ and $\mathbf{T} = \{T_1, \ldots, T_m\}$ be schemas with no relation symbols in common. We call $\mathbf{S}$ the *source schema* and $\mathbf{T}$ the *target schema*. We write $\langle \mathbf{S}, \mathbf{T} \rangle$ to denote the schema $\{S_1, \ldots, S_n, T_1, \ldots, T_m\}$. Instances over $\mathbf{S}$ and $\mathbf{T}$ are called *source* and *target instances*, respectively. If I is a source instance and J a target instance, then their combination $\langle I, J \rangle$ is an instance of the schema $\langle \mathbf{S}, \mathbf{T} \rangle$. In this thesis we assume the instances over $\mathbf{S}$ to be ground, that is, $dom(I) = Const(I)$, whereas target instances may contain nulls.

2.1 Homomorphisms and substitutions

Let I, I' be instances. A *homomorphism* $h\colon I \to I'$ is a mapping $dom(I) \to dom(I')$, such that (1) whenever a fact $R(\overline{x}) \in I$, then $R(h(\overline{x})) \in I'$, and (2) for every constant c, $h(c) = c$. If such h exists, we write $I \to I'$. Moreover, if both $I \to I'$ and $I \leftarrow I'$ holds, abbreviated as $I \leftrightarrow I'$, then we say that I and I' are *homomorphically equivalent*. In contrast, if $I \to I'$ but not vice versa, we say that I is *more general* than I', and I' is *more specific* than I.

If $h\colon I \to I'$ is invertible, s.t. h^{-1} is a homomorphism from I' to I, then h is called an *isomorphism*, and the instances I, I' *isomorphic*, denoted $I \cong I'$. An *endomorphism* is a homomorphism $I \to I$. An endomorphism is *proper* if it is not surjective (for finite instances, this is equivalent to being not injective), i.e., if it reduces the domain of I.

If I is an instance, and $I' \subseteq I$ is such that $I \to I'$ holds but for no other $I'' \subset I'\colon I \to I''$ (that is, I' cannot be further "shrunk" by a proper endomorphism), then I' is called a *core* of I [24, 33]. The core is unique up to isomorphism [33]. Hence, we may speak about *the* core of I. Cores have the following important property: for arbitrary instances J and J', $J \leftrightarrow J'$ *iff* $core(J) \cong core(J')$ [33].

A *substitution* σ is a mapping which sends variables to other domain elements (i.e., variables or constants). We write $\sigma = \{x_1 \leftarrow a_1, \ldots, x_n \leftarrow a_n\}$ if σ maps each x_i to a_i and σ is the identity outside $\{x_1, \ldots, x_n\}$. The application of a substitution is usually denoted in postfix notation, e.g., $x\sigma$ denotes the image of x under σ. For an expression $\varphi(\overline{x})$, we write $\varphi(\overline{x}\sigma)$ to denote the result of replacing every occurrence of every variable $x \in \overline{x}$ by $x\sigma$. We will sometimes identify σ with the tuple $\overline{a} = \langle a_1, \ldots, a_n\rangle$ of respective images for the variables $\langle x_1, \ldots, x_n\rangle$ and speak of the *assignment* $\overline{a}$ *for* $\overline{x}$. In this case, $\varphi(\overline{a})$ is a shorthand for $\varphi(\overline{x}\sigma)$.

2.2 Conjunctive queries

A *conjunctive query* (CQ) over a schema $\mathbf{S}$ is a formula $\varphi(\overline{x}) = R_1(\overline{x}_i) \wedge \ldots \wedge R_n(\overline{x}_n)$ where $R_i \in \mathbf{S}$ and $\overline{x}_i \subseteq \overline{x}$, for $1 \leq i \leq n$. We write $At(\varphi(\overline{x}))$ or simply $At(\varphi)$ to denote the instance in which the tuples correspond exactly to the set of atoms of $\varphi(\overline{x})$. Thereby, if the variables in $\overline{x}$ are substituted with distinct fresh constants in $At(\varphi(\overline{x}))$ — that is, with constants not occurring in $\varphi(\overline{x})$ — the latter instance is called a *frozen database of* φ. Such an instance is known in the literature as a *canonical database of* $\varphi(\overline{x})$. However, unless otherwise specified, we assume that the elements in $\overline{x}$ are instantiated with distinct labelled nulls in $At(\varphi(\overline{x}))$. Furthermore, in Chapters 3 and 4 we will consider CQs with variables partitioned in two sets, written as $\varphi(\overline{x}_1, \overline{x}_2)$. In $At(\varphi(\overline{x}_1, \overline{x}_2))$, the elements of $\overline{x}_1$ and $\overline{x}_2$ can be instantiated differently, e.g., often we will instantiate $\overline{x}_1$ with distinct fresh constants and $\overline{x}_2$ with distinct fresh labelled nulls. The way of instantiation is specified explicitly in such situations.

We say that an instance I of $\mathbf{S}$ satisfies $\varphi(\overline{x})$, written $I \models \varphi(\overline{x})$, if there exists an assignment $\overline{a} \subseteq dom(I)$ for $\overline{x}$ such that $At(\varphi(\overline{a})) \subseteq I$. For CQs $\varphi(\overline{a})$ without variables, $I \models \varphi(\overline{a})$ obviously coincides with $At(\varphi(\overline{a})) \subseteq I$.

2.3 Schema mappings and data exchange

A *schema mapping* is given by a triple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ where $\mathbf{S}$ is the source schema, $\mathbf{T}$ is the target schema, and Σ is a set of dependencies expressing the relationship between $\mathbf{S}$ and $\mathbf{T}$ and possibly also local constraints on $\mathbf{S}$ and $\mathbf{T}$. The Data Exchange Problem associated with $\mathcal{M}$

is the following: Given a (ground) source instance I, find a target instance J, s.t. $\langle I, J\rangle \models \Sigma$. Such a J is called a *solution for I under $\mathcal{M}$* or, simply, a *solution* if I and $\mathcal{M}$ are clear from the context. The set of all solutions for I under $\mathcal{M}$ is denoted by $Sol(I, \mathcal{M})$. If $J \in Sol(I, \mathcal{M})$ is such that $J \to J'$ holds for any other solution $J' \in Sol(I, \mathcal{M})$, then J is called a *universal solution*. The set of all universal solutions for I under $\mathcal{M}$ is denoted as $UnivSol(I, \mathcal{M})$. Since the universal solutions for a source instance I are homomorphically equivalent, the core of the universal solutions for I is unique up to isomorphism. It will be denoted as $core(I, \mathcal{M})$. The core is the smallest universal solution [24].

In the following, we will often identify a schema mapping $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma\rangle$ with the set of dependencies Σ, without explicitly mentioning the schemas, for the sake of brevity. In such situations, we will also write $core(I, \Sigma)$ instead of $core(I, \mathcal{M})$

2.4 Dependencies

Embedded dependencies [19] over a relational schema $\mathbf{R}$ are first-order formulas of the form

$$\forall \overline{x}\big(\varphi(\overline{x}\,) \to \exists \overline{y}\ \psi(\overline{x}, \overline{y}\,)\big)$$

In case of *tuple-generating dependencies* (tgds), both *antecedent* φ and *conclusion* ψ are conjunctive queries (CQs) over the relation symbols from $\mathbf{R}$ such that all variables in $\overline{x}$ actually do occur in $\varphi(\overline{x}\,)$. *Equality-generating dependencies* (egds) are of the form

$$\forall \overline{x}\,(\varphi(\overline{x}\,) \to x_i = x_j)$$

with $x_i, x_j \in \overline{x}$. Throughout this thesis, we shall omit the universal quantifiers: By convention, all variables occurring in the antecedent are universally quantified (over the entire formula). If there is no existentially quantified variables in the conclusion of a tgd, the latter is called *full tgd*.

We denote the antecedent of an embedded dependency τ as $ant(\tau)$. The instance $At(ant(\tau))$ will be often referred to as *antecedent database of* τ. If τ is a tgd with the conclusion $(\exists \overline{y})\varphi(\overline{x}, \overline{y})$, we will also consider a CQ $\varphi(\overline{x}, \overline{y})$ and a corresponding instance $At(\varphi(\overline{x}, \overline{y}))$ called *conclusion database of* τ. In $At(\varphi(\overline{x}, \overline{y}))$, elements of $\overline{x}$ will be typically instantiated with fresh distinct constants whereas elements of $\overline{y}$ with fresh distinct labelled nulls.

In Chapters 3 and 4, we also split the variables $\overline{x}$ in the antecedent of a dependency in two disjoint sets: one contains the variables which occur in the conclusion of the dependency while the other contains the variables which do not occur in the conclusion. We thus write down the dependency as $\varphi(\overline{x}_1, \overline{x}_2) \to (\exists \overline{y})\ \psi(\overline{x}_1, \overline{y})$. Clearly, all variables in $\overline{x}_1$ are supposed to occur in $\psi(\overline{x}_1, \overline{y})$ in this case.

In the context of data exchange, we are mainly dealing with *source-to-target dependencies* consisting of tuple-generating dependencies (or s-t tgds) over the schema $\langle \mathbf{S}, \mathbf{T}\rangle$ (the antecedent is a CQ over $\mathbf{S}$, the conclusion over $\mathbf{T}$) and *target dependencies* over $\mathbf{T}$. Moreover, in Chapters 4 and 8, we shall also consider *source dependencies* consisting of egds over $\mathbf{S}$.

Second-Order tgds

A *second-order tuple generating dependency (SO-tgd)* [25] has the form

$$\exists \bar{f}\,(\,(\forall \overline{x}_1(\varphi_1 \rightarrow \psi_1)) \wedge \cdots \wedge (\forall \overline{x}_n(\varphi_n \rightarrow \psi_n))\,)\,,$$

where

(1) each member of $\bar{f}$ is a function symbol,

(2) each φ_i is a conjunction of atomic formulas over the source schema $\mathbf{S}$ with terms from $\overline{x}_i$ and equalities of the form $t = t'$, t and t' being terms based on $\overline{x}_i$ and $\bar{f}$,

(3) each ψ_i is a conjunction of atomic formulas over the target schema $\mathbf{T}$ of the form $T(t_1, \ldots, t_\ell)$, $t_1, \ldots, t_\ell$ being terms based on $\overline{x}_i$ and $\bar{f}$, and

(4) each variable in $\overline{x}_i$ appears in some atomic formula of φ_i.

When dealing with instances $\langle I, J\rangle$ in the context of SO tgds, target instances J may contain *functional terms* which can be treated as labelled nulls [25]. The domain of source instances I is assumed to consist of constants only, also in the context of SO tgds.

2.5 Chase

As has already been mentioned in Chapter 1, the data exchange problem can be solved by the *chase* [7, 22], a sequence of *chase steps*, each enforcing a single constraint within some limited set of tuples. More precisely, let Σ contain a tgd $\tau\colon \varphi(\overline{x}) \rightarrow (\exists \overline{y}\,)\psi(\overline{x}, \overline{y}\,)$, such that $I \models \varphi(\overline{a}\,)$ for some assignment $\overline{a}$ on $\overline{x}$. Then the chase step with τ and $\overline{a}$ extends I with facts corresponding to $\psi(\overline{a}, \overline{z}\,)$, where the elements of $\overline{z}$ are fresh labelled nulls. Note that this definition of the chase differs from the definition in [22], where no new facts are added if $I \models \exists \overline{y}\psi(\overline{a}, \overline{y}\,)$ is already fulfilled. Omitting this check is referred to as *oblivious* [36] chase or *naïve* [1] chase. It is the preferred version of chase if the result of the chase should not depend on the order in which the tgds are applied (see e.g., [1, 3, 39]).

Now suppose that Σ contains an egd $\varepsilon\colon \varphi(\overline{x}\,) \rightarrow x_i = x_j$, s.t. $I \models \varphi(\overline{a}\,)$ for some assignment $\overline{a}$ on $\overline{x}$. This egd enforces the equality $a_i = a_j$. At the chase step with ε and $\overline{a}$ we thus choose a null a' among $\{a_i, a_j\}$ and replace *every occurrence* of a' in I by the other term; if $a_i, a_j \in Const(I)$ and $a_i \neq a_j$, the chase halts with *failure*.

The chase proceeds until all combinations of dependencies $\varphi(\overline{x}) \rightarrow (\exists \overline{y})\ \psi(\overline{x}, \overline{y})$ and assignments $\overline{a}$ for $\overline{x}$, consisting of values from $dom(I)$ such that $I \models \varphi(\overline{a})$, have been tried. A *chase sequence* is a sequence of chase steps. The *chase* of I with Σ, denoted as $chase(I, \Sigma)$, is a chase sequence including all possible chase steps.

Consider an arbitrary schema mapping $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma\rangle$ with $\Sigma = \Sigma_s \cup \Sigma_{st} \cup \Sigma_t$ where Σ_s is a set of source egds, Σ_{st} is a set of source-to-target tgds and Σ_t is a set of target tgds and egds. If $I \not\models \Sigma_s$, the chase terminates with failure: recall that by our convention I is ground and Σ_s contains of egds, so a unification of two distinct constants is implied. If no source egd is violated, then the solution to a source instance I can be computed as follows: We start off with the instance $\langle I, \emptyset\rangle$, i.e., the source instance is I and the target instance is initially empty.

Chasing $\langle I, \emptyset \rangle$ with Σ_{st} yields the instance $\langle I, J \rangle$, where J is called the *preuniversal instance*. This chase always succeeds since Σ_{st} contains no egds, and can be computed in polynomial time [22]. Then J is chased with Σ_t. This chase may fail on an attempt to unify distinct constants. If the chase succeeds, we end up with $U = chase(J, \Sigma_t)$, which is referred to as the *canonical universal solution* $CanSol(I, \Sigma)$, which, depending on the context, will be also denoted as $CanSol(I, \mathcal{M})$ or simply $CanSol(I)$.

If Σ_t contains tgds, the chase sequence may be infinite. As already mentioned in the Introduction, in this thesis we will not deal with such sets of dependencies. Instead, we assume that Σ belongs to some class of mappings for which the chase sequence is finite, for any source instance. Examples of such classes can be found in [22, 43, 46, 56]. Interestingly, for all known so far classes of mappings having only finite chase sequences, the length of these sequences is polynomially bounded.

For a SO tgd $\exists \bar{f}\,((\forall \overline{x}_1(\varphi_1 \rightarrow \psi_1)) \wedge \cdots \wedge (\forall \overline{x}_n(\varphi_n \rightarrow \psi_n)))$, the chase step consists of an application of a single implicational conjunct $(\forall \overline{x}_i)\ (\varphi_i \rightarrow \psi_i)$ to a given instance. Such a chase step is very similar to a chase step with a tgd, whereby also the equalities in the antecedents have to be taken into account. Formally, we say that a mapping h from a conjunct $C_i = (\forall \overline{x})(\varphi_i \rightarrow \psi_i)$ of an SO tgd to an instance I is a homomorphism if for every relational atom $S(y_1, \ldots, y_k)$ in φ_i the tuple $(h(y_1), \ldots, h(y_k))$ is in S^I and for every equality $t = t'$ we have $h(t) = h(t')$. The chase step with a conjunct of C_i and a homomorphism h can be defined similarly to a chase step with a tgd: The instance I is extended with a tuples $(h(t_1), \ldots, h(t_\ell))$ for each atom $T(t_1, \ldots, t_\ell)$ in ψ_i, where the terms $h(t_i)$ are understood as labelled nulls.

As in the case of embedded dependencies, the chase with SO tgds can be done in polynomial time w.r.t. the size of the source instance and results in a universal solution [25].

2.6 Equivalence of schema mappings

Let $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$ and $\mathcal{M}' = \langle \mathbf{S}, \mathbf{T}, \Sigma' \rangle$ be two schema mappings.

Logical equivalence. $\mathcal{M}$ and $\mathcal{M}'$ are *logically equivalent*, denoted as $\mathcal{M} \equiv \mathcal{M}'$, if, for every source instance I and target instance J, $\langle I, J \rangle \models \Sigma$ iff $\langle I, J \rangle \models \Sigma'$. This is the case if $Sol(I, \mathcal{M}) = Sol(I, \mathcal{M}')$ holds for every source instance I.

Data exchange equivalence (DE-equivalence). $\mathcal{M}$ and $\mathcal{M}'$ are DE-equivalent, denoted as $\mathcal{M} \equiv_{DE} \mathcal{M}'$, if, for every source instance I, the universal solutions coincide, i.e.: the equality $UnivSol(I, \mathcal{M}) = UnivSol(I, \mathcal{M}')$ holds for every source instance I.

Conjunctive query equivalence (CQ-equivalence). To formally define this notion, we need a definition of *certain answers*: For a schema mapping $\mathcal{M}$ defined as above, the set of certain answers to a query q over the schema $\mathbf{T}$ and for a source instance I is a set

$$certain_{\mathcal{M}}(q, I) = \bigcap \{q(J) | J \in Sol(I, \mathcal{M})\}.$$

Now, $\mathcal{M}$ and $\mathcal{M}'$ are CQ-equivalent, denoted as $\mathcal{M} \equiv_{CQ} \mathcal{M}'$ if, for every source instance I, and for any conjunctive query q over $\mathbf{T}$, the equality $certain_{\mathcal{M}}(q, I) = certain_{\mathcal{M}'}(q, I)$.

As mentioned in the previous chapter, the implications $(\mathcal{M}_1 \equiv \mathcal{M}_2) \Rightarrow (\mathcal{M}_1 \equiv_{DE} \mathcal{M}_2) \Rightarrow (\mathcal{M}_1 \equiv_{CQ} \mathcal{M}_2)$ have been proved in [23], whereas the converse of neither implication holds in general. Furthermore, [23] presents an alternative characterization of CQ-equivalence for mappings that satisfy the following property: for every source I if there is a solution for I then there is a universal solution for I. Mappings given by embedded dependencies which do not cause infinite chase sequences, as well as mappings given by the SO-tgds, possess this property: Namely, $chase(I, \Sigma)$ is a desired universal solution for a mapping $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$ and a source instance I [22, 25]. The characterization in [23, Proposition 3.5], which will be used as *the definition of CQ-equivalence in this thesis*, is as follows:

Proposition 1 (CQ-EQUIVALENCE [23])**.** *Let $\mathcal{M}$ and $\mathcal{M}'$ be schema mappings given by sets of embedded dependencies or by SO-tgds. Then, $\mathcal{M} \equiv_{CQ} \mathcal{M}'$ if, for every source instance I, either $Sol(I, \mathcal{M}) = \emptyset = Sol(I, \mathcal{M}')$ or $core(I, \mathcal{M}) = core(I, \mathcal{M}')$.*

Part I

Logical equivalence

CHAPTER 3

Optimization and normalization of mappings defined by s-t tgds

In this chapter, we investigate ways of optimizing sets of s-t tgds relative to logical equivalence. In the first place, we thus formulate some natural optimality criteria. The following parameters of a set of s-t tgds will be needed in the definition of such criteria:

Definition 1. *Let Υ be a set of s-t tgds. Then we define:*

- $|\Upsilon|$ *denotes the number of tgds in* Υ.
- $AntSize(\Upsilon) = \Sigma\{|At(\varphi(\overline{x}))|\colon \varphi(\overline{x}) \rightarrow \exists\overline{y}\ \psi(\overline{x},\overline{y})$ *is a tgd in* $\Upsilon\}$*, i.e.,* $AntSize(\Upsilon)$ *is the total number of atoms in all antecedents of tgds in* Υ.
- $ConSize(\Upsilon) = \Sigma\{|At(\psi(\overline{x},\overline{y}))|\colon \varphi(\overline{x}) \rightarrow \exists\overline{y}\,\psi(\overline{x},\overline{y})$ *is a tgd in* $\Upsilon\}$*, i.e.,* $ConSize(\Upsilon)$ *is the total number of atoms in all conclusions of tgds in* Υ.
- $VarSize(\Upsilon) = \Sigma\{|\overline{y}|\colon \varphi(\overline{x}) \rightarrow \exists\overline{y}\ \psi(\overline{x},\overline{y})$ *is a tgd in in* $\Upsilon\}$*, i.e.,* $VarSize(\Upsilon)$ *is the total number of existentially quantified variables in all conclusions of tgds in* Υ.

We would naturally like to transform any set of s-t tgds into an equivalent one where all the above parameters are minimal. Recall however our discussion on the splitting of s-t tgds from Example 4. As we pointed out there, the splitting of s-t tgds is comparable to normal form decomposition of relational schemas. It should clearly be applied in order to avoid anomalies like the introduction of obviously irrelevant atoms in the canonical universal solution as we saw in Example 4, where the set Σ (with two split s-t tgds) was certainly preferable to Σ' even though $|\Sigma'| < |\Sigma|$ and $AntSize(\Sigma') < AntSize(\Sigma)$ hold. Note that in Example 4, the equality $ConSize(\Sigma') = ConSize(\Sigma)$ holds. Intuitively, the effect of splitting is that the atoms in the conclusion of some s-t tgd are distributed over several strictly smaller s-t tgds. Thus, our goal should be to find an optimal set of s-t tgds (that is, a set where the above mentioned parameters

are minimal) among those sets of s-t tgds for which no further splitting is possible. We now make precise what it means that "no further splitting" is possible and formally define optimality of a set of s-t tgds.

Definition 2. *Let Σ be a set of s-t tgds. We say that Σ is* split-reduced *if there exists no Σ' equivalent to Σ, s.t. $|\Sigma'| > |\Sigma|$ but $ConSize(\Sigma') = ConSize(\Sigma)$.*

Definition 3. *Let Σ be a set of s-t tgds. We say that Σ is* optimal *if it is split-reduced, and if each of the parameters $|\Sigma|$, $AntSize(\Sigma)$, $ConSize(\Sigma)$, and $VarSize(\Sigma)$ is minimal among all split-reduced sets equivalent to Σ.*

Of course, given an arbitrary set Σ of s-t tgds, it is a priori not clear that an optimal set Σ' equivalent to Σ exists, since it might well be the case that some Σ' minimizes some of the parameters while another set Σ'' minimizes the other parameters. The goal of this chapter is to show that optimality in the above sense can always be achieved and to construct an algorithm which transforms any set Σ of s-t tgds into an equivalent optimal one. To this end, we introduce a rewrite system which consists of two kinds of rewrite rules: rules which simplify each s-t tgd individually and rules which are applied to the entire set of s-t tgds. The following example illustrates several kinds of redundancy that a single s-t tgd may contain (and which may be eliminated with our rewrite rules).

Example 10. *Consider the following dependency:*

$$\tau\colon S(x_1,x_3)\wedge S(x_1,x_2) \rightarrow (\exists y_1,y_2,y_3,y_4,y_5)\ P(x_1,y_2,y_1)\wedge R(y_1,y_3,x_2)\wedge R(2,y_3,x_2) \wedge P(x_1,y_4,2)\wedge P(x_1,y_4,y_5)\wedge Q(y_4,x_3)$$

Clearly, τ is equivalent to the set $\{\tau_1,\tau_2\}$ of s-t tgds:

$$\tau_1\colon S(x_1,x_3)\wedge S(x_1,x_2) \rightarrow (\exists y_1,y_2,y_3)\ P(x_1,y_2,y_1)\wedge R(y_1,y_3,x_2)\wedge R(2,y_3,x_2)$$
$$\tau_2\colon S(x_1,x_3)\wedge S(x_1,x_2) \rightarrow (\exists y_4,y_5)\ P(x_1,y_4,2)\wedge P(x_1,y_4,y_5)\wedge Q(y_4,x_3)$$

Now the antecedents of τ_1 and τ_2 can be simplified:

$$\tau_1'\colon S(x_1,x_2) \rightarrow (\exists y_1,y_2,y_3)\ P(x_1,y_2,y_1)\wedge R(y_1,y_3,x_2)\wedge R(2,y_3,x_2)$$
$$\tau_2'\colon S(x_1,x_3) \rightarrow (\exists y_4,y_5)\ P(x_1,y_4,2)\wedge P(x_1,y_4,y_5)\wedge Q(y_4,x_3)$$

Finally, we may also simplify the conclusion of τ_2':

$$\tau_2''\colon S(x_1,x_3) \rightarrow (\exists y_4)P(x_1,y_4,2)\wedge Q(y_4,x_3)$$

In total, τ is equivalent to $\{\tau_1',\tau_2''\}$.

For the simplifications illustrated in Example 10, we define the rewrite rules 1 – 3 in Figure 3.1. Rules 1 and 2 replace an s-t tgd τ by a simpler one (i.e., with fewer atoms) τ', while Rules 3 replaces τ by a set $\{\tau_1,\ldots,\tau_n\}$ of simpler s-t tgds. These rules make use of the following definitions of the *core* and the *components* of CQs.

Rewrite rules to simplify a set of s-t tgds

Rule 1 (Core of the conclusion, see Definition 4).
$\tau\colon \varphi(\overline{x}\,) \to (\exists\overline{y}\,)\psi(\overline{x},\overline{y}\,) \implies$
$\tau'\colon \varphi(\overline{x}\,) \to (\exists\overline{y}\,)\psi(\overline{x},\overline{y}\sigma)$,
s.t. $\psi(\overline{x},\overline{y}\sigma)$ is the core of $\psi(\overline{x},\overline{y})$.

Rule 2 (Core of the antecedent, see Definition 4).
$\tau\colon \varphi(\overline{x}_1,\overline{x}_2) \to (\exists\overline{y}\,)\psi(\overline{x}_1,\overline{y}\,) \implies$
$\tau'\colon \varphi(\overline{x}_1,\overline{x}_2\sigma) \to (\exists\overline{y})\psi(\overline{x}_1,\overline{y})$,
s.t. $\varphi(\overline{x}_1,\overline{x}_2\sigma)$ is the core of $\varphi(\overline{x}_1,\overline{x}_2)$.

Rule 3 (Splitting, see Definition 5).
$\tau\colon\ \varphi(\overline{x}\,) \to (\exists\overline{y}\,)\psi(\overline{x},\overline{y}\,) \implies \{\tau_1,\ldots,\tau_n\}$, s.t.
$\tau_i\colon\ \varphi(\overline{x}\,) \to (\exists\overline{y}_i)\psi_i(\overline{x},\overline{y}_i)$ for $i \in \{1,\ldots,n\}$ and
$\{\psi_1(\overline{x},\overline{y}_1),\ldots,\psi_n(\overline{x},\overline{y}_n)\}$ are the components of $\psi(\overline{x},\overline{y}\,)$

Rule 4 (Implication of an s-t tgd).
$\Sigma \implies \Sigma \setminus \{\tau\}$
if $\Sigma \setminus \{\tau\} \models \tau$.

Rule 5 (Implication of atoms in the conclusion).
$\Sigma \implies (\Sigma \setminus \{\tau\}) \cup \{\tau'\}$
if $\tau\colon \varphi(\overline{x}\,) \to (\exists\overline{y}\,)\psi(\overline{x},\overline{y}\,)$
and $\tau'\colon \varphi(\overline{x}\,) \to (\exists\overline{y}\,')\psi'(\overline{x},\overline{y}\,')$,
s.t. $At(\psi'(\overline{x},\overline{y}\,')) \subset At(\psi(\overline{x},\overline{y}\,))$
and $(\Sigma \setminus \{\tau\}) \cup \{\tau'\} \models \tau$.

Figure 3.1: Redundancy elimination from a set of s-t tgds.

Definition 4. *Let $\chi(\overline{u},\overline{v}\,)$ be a CQ with variables in $\overline{u}\cup\overline{v}$ and let $\mathcal{A}$ denote the structure consisting of the atoms $At(\chi(\overline{u},\overline{v}\,))$, s.t. the variables $\overline{u}$ are considered as constants and the variables $\overline{v}$ as labelled nulls. Let $\mathcal{A}'$ denote the core of $\mathcal{A}$ with $\mathcal{A}' \subseteq \mathcal{A}$, i.e., there exists a substitution $\sigma\colon \overline{v} \to Const \cup \overline{u} \cup \overline{v}$ s.t. $At(\chi(\overline{u},\,\overline{v}\,\sigma)) = \mathcal{A}' \subseteq At(\chi(\overline{u},\overline{v}\,))$. Then we define the* core of $\chi(\overline{u},\overline{v}\,)$ *as the CQ $\chi(\overline{u},\overline{v}\sigma)$.*

Definition 5. *Let $\chi(\overline{u},\overline{v}\,)$ be a CQ with variables in $\overline{u}\cup\overline{v}$. We set up the* dual graph $\mathcal{G}(\tau)$ *as follows: The atoms of $\chi(\overline{u},\overline{v}\,)$ are the vertices of $\mathcal{G}(\tau)$. Two vertices are connected if the corresponding atoms have at least one variable from $\overline{v}$ in common. Let $\{C_1,\ldots,C_n\}$ denote the connected components of $\mathcal{G}(\tau)$. Moreover, for every $i \in \{1,\ldots,n\}$, let $\overline{v}_i$ with $\emptyset \subseteq \overline{v}_i \subseteq \overline{v}$ denote those variables from $\overline{v}$, which actually occur in C_i and let $\chi_i(\overline{u},\overline{v}_i)$ denote the CQ consisting of the atoms in C_i. Then we define the* components of $\chi(\overline{u},\overline{v}\,)$ *as the set $\{\chi_1(\overline{u},\overline{v}_1),\ldots,\chi_n(\overline{u},\overline{v}_n)\}$.*

The splitting rule (i.e., Rule 3 in Figure 3.1) was already applied in Example 4. Rule 2 involving core computation of the antecedent was applied in Example 2, when we reduced $L(x_1,x_2,x_3) \wedge L(x_4,3,x_5) \wedge P(x_5,x_6)$ to its core $L(x_4,3,x_5) \wedge P(x_5,x_6)$. Likewise, in Example 10, the simplification of τ_1 and τ_2 to τ_1' and τ_2' is due to Rule 2. In a similar way, Rule 1 involving core computation of the conclusion allowed us to reduce $L(y_1,y_2,y_3) \wedge L(x_1,3,y_4) \wedge$

$P(y_4, x_2)$ in Example 3 to $L(x_1, 3, y_4) \wedge P(y_4, x_2)$. In Example 10, Rule 1 was applied when we replaced τ_2' by τ_2''.

The following example illustrates that additional rules are required in order to remove an s-t tgd or a part of an s-t tgd whose redundancy is due to the presence of other s-t tgds.

Example 11. *Consider the set $\Sigma = \{\tau_1', \tau_2'', \tau_3\}$, where τ_1' and τ_2'' are the s-t tgds resulting from the simplification steps in Example 10:*

$$\tau_1' \colon S(x_1, x_2) \rightarrow (\exists y_1, y_2, y_3)\ P(x_1, y_2, y_1) \wedge R(y_1, y_3, x_2) \wedge R(2, y_3, x_2)$$
$$\tau_2'' \colon S(x_1, x_3) \rightarrow (\exists y_4)\ P(x_1, y_4, 2) \wedge Q(y_4, x_3)$$
$$\tau_3 \colon\ S(2, x)\ \rightarrow (\exists y)\ R(2, y, x)$$

The tgd τ_3 generates only a part of the atoms that τ_1' does, and fires in strictly fewer cases than τ_1'. Hence, τ_3 may be deleted. Moreover, considering the combined effect of the rules τ_1' and τ_2'', which fire on exactly the same tuples, and a substitution $\{y_1 \leftarrow 2, y_2 \leftarrow y_4\}$, we notice that the first two atoms in the conclusion of τ_1' are in fact redundant, and it is possible to reduce τ_1' to $\tau_1'' \colon\ S(x_1, x_2) \rightarrow (\exists y_3) R(2, y_3, x_2)$. In total, Σ may be replaced by $\Sigma' = \{\tau_1'', \tau_2''\}$.

Rules 4 and 5 in Figure 3.1 allow us to eliminate such redundancies from a set Σ of s-t tgds: By Rule 4, we may delete an s-t tgd τ from Σ, if τ is implied by the others, like τ_3 in Example 11. Rule 5 allows us to replace a rule τ by a strictly smaller rule (with fewer atoms in the conclusion) if τ is implied by τ' together with the remaining s-t tgds in Σ (cf. the replacement of τ_1' with τ_1'' in Example 11 above). Figure 3.2 illustrates the elimination of redundant atoms via Rules 1, 2, 4 and 5 in a set $\{\tau_1, \tau_2', \tau_3\}$ of tgds from Examples 10 and 11.

In principle, the implication of a tgd by a set of dependencies can be tested by a procedural criterion based on the chase [7]. For our purposes, the following, declarative criterion is more convenient.

Lemma 1. *Consider an s-t tgd $\tau \colon \varphi(\overline{x}) \rightarrow (\exists \overline{y}\,)\psi(\overline{x}, \overline{y}\,)$ and a set Σ of s-t tgds. Then $\Sigma \models \tau$ holds iff there exist (not necessarily distinct) s-t tgds $\tau_1, \ldots, \tau_k$ in Σ, such that all s-t tgds $\tau, \tau_1, \ldots, \tau_k$ are pairwise variable disjoint and the following conditions hold:*

(a) For every $i \in \{1, \ldots, k\}$, there exists a substitution $\lambda_i \colon \overline{x}_i \rightarrow Const \cup \overline{x}$, such that $At(\varphi_i(\overline{x}_i\lambda_i)) \subseteq At(\varphi(\overline{x}\,))$.

(b) A substitution $\mu \colon \overline{y} \rightarrow Const \cup \overline{x} \cup \bigcup_{i=1}^{k} \overline{y}_i$ exists, such that the following inclusion holds: $At(\psi(\overline{x}, \overline{y}\mu)) \subseteq \bigcup_{i=1}^{k} At(\psi_i(\overline{x}_i\lambda_i, \overline{y}_i))$.

Proof. For the "$\Rightarrow$"-direction, consider an arbitrary pair $\langle I, J\rangle$ of source and target instance, s.t. $\langle I, J\rangle \models \Sigma$. It is easy to show that, by conditions (a) and (b), then also $\langle I, J\rangle \models \tau$ holds. For the "$\Leftarrow$"-direction, we take the source instance $I = At(\varphi(\overline{x}))$, where we consider the variables $\overline{x}$ as constants. Moreover, let J denote the target instance which results from the *oblivious* chase of I with Σ. Let $\tau_1, \ldots, \tau_k$ denote the (not necessarily distinct) s-t tgds whose antecedent can be mapped into I via substitutions $\lambda_1, \ldots, \lambda_k$. These substitutions satisfy the condition (a). By $\langle I, J\rangle \models \Sigma$ and $\Sigma \models \tau$ we get the desired substitution μ for condition (b). □

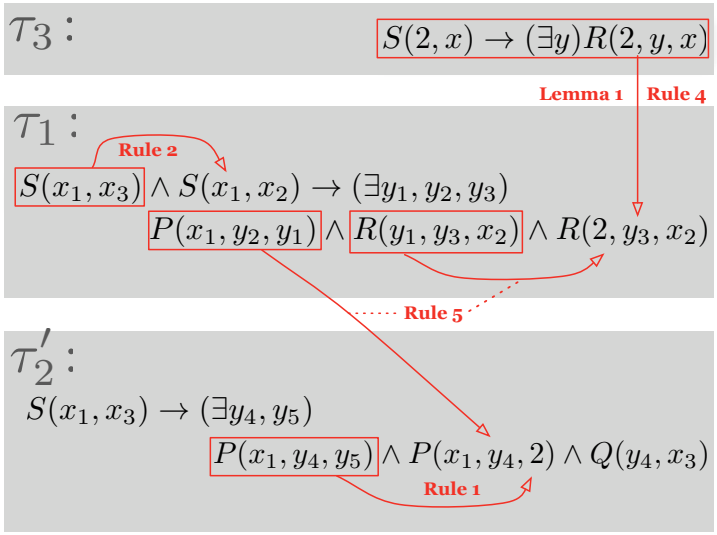

Figure 3.2: Tgd optimization. Rectangles mark eliminated atoms, arrows show justifications for elimination.

Note that Rule 5 generalizes Rule 1 and, in principle, also Rule 4. Indeed, if we restrict Σ in Rule 5 to the singleton $\Sigma = \{\tau\}$, then the replacement of τ by τ' means that we reduce $\psi(\overline{x},\overline{y}\,)$ to its core. Moreover, Rule 5 allows us to eliminate all atoms from the conclusion of τ iff τ may be deleted via Rule 4. Clearly, the deletion of the conclusion of τ essentially comes down to the deletion of τ itself.

The correctness of Rules 1 – 5 can be easily established. For the proof, we will need the following notion of a "proper instance" of an s-t tgd. It will also play an important role for showing that the Rules 1 – 5 lead to a unique normal form. A proper instance of an s-t tgd τ is obtained from τ by eliminating at least one existentially quantified variable in the conclusion of τ, while keeping the antecedent unchanged.

Definition 6. *Let $\tau\colon \varphi(\overline{x}) \rightarrow (\exists\overline{y}\,)\psi(\overline{x},\overline{y}\,)$ be an s-t tgd. We call an s-t tgd τ' a* proper instance *of τ, if there exists a strict subset $\overline{y}\,' \subset \overline{y}$ and a substitution $\sigma\colon \overline{y} \rightarrow Const \cup \overline{x} \cup \overline{y}\,'$, such that τ' is of the form $\tau'\colon\ \varphi(\overline{x}\,) \rightarrow (\exists\overline{y}\,')\psi(\overline{x},\overline{y}\sigma)$.*

Example 12. *In the following three tgds, each next tgd is a proper instance of the previous ones:*

$\tau_1\colon\ S(x_1,x_2) \rightarrow (\exists y_1,y_2)\ Q(x_1,y_1,y_2)$
$\tau_2\colon\ S(x_1,x_2) \rightarrow (\exists y_1)\ Q(x_1,y_1,y_1)$
$\tau_3\colon\ S(x_1,x_2) \rightarrow Q(x_1,x_2,x_2)$

Moreover, observe that $\tau_2 \models \tau_1$ and $\tau_3 \models \tau_2$ holds.

The importance of "proper instances" to our investigations comes from the following properties:

Lemma 2. *Let τ and τ' be s-t tgds, such that τ' is a proper instance of τ. Then the following properties hold:*

(1) $\tau' \models \tau$.

(2) Suppose that τ *is reduced with respect to Rule 1. Then* $\tau \not\models \tau'$.

Proof. (1) Consider two tgds $\tau\colon\ \varphi(\overline{x}\,) \rightarrow (\exists \overline{y}\,)\ \psi(\overline{x}, \overline{y}\,)$ and $\tau'\colon\ \varphi(\overline{x}\,) \rightarrow (\exists \overline{y}\,')\,\psi(\overline{x}, \overline{y}\sigma)$. Let $\langle I, J\rangle$ be an arbitrary pair of source and target instances with $\langle I, J\rangle \models \tau'$ and let $\lambda\colon \overline{x} \rightarrow dom(I)$ be a substitution, such that $At(\varphi(\overline{x}\,\lambda)) \subseteq I$. We show that then also $J \models \psi(\overline{x}\lambda, \overline{y}\,)$ holds. By $\langle I, J\rangle \models \tau'$, we get $J \models \psi(\overline{x}\lambda, \overline{y}\sigma)$, that is, there exists a substitution μ, such that $At(\psi(\overline{x}\lambda, \overline{y}\sigma\mu)) \subseteq J$. But then, for $\nu = \sigma\mu$, we have $At(\psi(\overline{x}, \overline{y}\sigma)) \subseteq J$. Thus, $J \models \psi(\overline{x}\lambda, \overline{y}\,)$ holds.

(2) Suppose that $\tau \models \tau'$ holds. We have to show that then Rule 1 is applicable to τ. Let $\langle I, J\rangle$ denote a pair of source and target instance with $I = At(\varphi(\overline{x}\,))$ and $J = At(\psi(\overline{x}, \overline{y}\,))$. The variables in $\overline{x}$ are thus considered as constants while $\overline{y}$ are labelled nulls. Clearly, $\langle I, J\rangle \models \tau$ and $I \models \varphi(\overline{x}\,)$. Thus, by the assumption $\tau \models \tau'$, also $J \models \psi(\overline{x}, \overline{y}\sigma)$ holds, i.e., there exists a substitution $\mu\colon \overline{y}\,' \rightarrow dom(J)$ such that $At(\psi(\overline{x}, \overline{y}\sigma\mu)) \subseteq J$. Hence, also the following inclusion holds: $At(\psi(\overline{x}, \overline{y}\sigma\mu)) \subseteq At(\psi(\overline{x}, \overline{y}\,))$. Note that $\overline{y}\,' = \overline{y}\sigma \subset \overline{y}$. Hence, also $At(\psi(\overline{x}, \overline{y}\sigma\mu)) \subset At(\psi(\overline{x}, \overline{y}\,))$. But then $At(\psi(\overline{x}, \overline{y}\,))$ is not a core and, therefore, Rule 1 is applicable to τ. □

We are now ready to prove the correctness of our transformation rules.

Lemma 3. *The Rules 1 – 5 in Figure 3.1 are correct, i.e.: Let* Σ *be a set of s-t tgds and* $\tau \in \Sigma$. *Suppose that* Σ *is transformed into* Σ' *by applying one of the Rules 1 – 5 to* τ, *that is:*

- τ *is replaced by a single s-t tgd* τ' *(via Rule 1,2,5),*
- τ *is replaced by s-t tgds* $\tau_1, \ldots, \tau_n$ *(via Rule 3),*
- *or* τ *is deleted (via Rule 4).*

Then Σ *and* Σ' *are equivalent.*

Proof.
Rule 1. Suppose that an s-t tgd τ is replaced by τ' via Rule 1. Then the s-t tgds τ and τ' are of the form $\tau\colon\ \varphi(\overline{x}) \rightarrow (\exists \overline{y}\,)\psi(\overline{x}, \overline{y}\,)$ and $\tau'\colon \varphi(\overline{x}\,) \rightarrow (\exists \overline{y}\,)\psi(\overline{x}, \overline{y}\sigma)$, s.t. $At(\psi(\overline{x}, \overline{y}\sigma)) \subset At(\psi(\overline{x}, \overline{y}))$. In particular, τ' is a "proper instance" of τ according to Definition 6. Hence, by Lemma 2, $\tau' \models \tau$ holds.

On the other hand, let $\langle I, J\rangle$ be an arbitrary pair of source and target instance with $\langle I, J\rangle \models \tau$ and let $\lambda\colon \overline{x} \rightarrow dom(I)$ be a substitution, s.t. $At(\varphi(\overline{x}\,\lambda) \subseteq I$. We have to show that then also $J \models \psi(\overline{x}\lambda, \overline{y}\sigma)$ holds. By assumption, $\langle I, J\rangle \models \tau$. Hence, $J \models \psi(\overline{x}\lambda, \overline{y}\,)$, i.e., there exists a substitution μ, s.t. $At(\psi(\overline{x}\lambda, \overline{y}\mu)) \subseteq J$. But then, since $At(\psi(\overline{x}, \overline{y}\sigma)) \subseteq At(\psi(\overline{x}, \overline{y}\,))$ holds, we also have $At(\psi(\overline{x}\lambda, \overline{y}\sigma\mu)) \subseteq J$. Thus, $\tau \models \tau'$ indeed holds.

Rule 2. Suppose that $\tau\colon\ \varphi(\overline{x}_1, \overline{x}_2) \rightarrow (\exists \overline{y}\,)\psi(\overline{x}_1, \overline{y}\,)$ is replaced by τ' via Rule 2. Then τ' must be of the form $\tau'\colon\ \varphi(\overline{x}_1, \overline{x}_2\sigma) \rightarrow (\exists \overline{y}\,)\psi(\overline{x}_1, \overline{y}\,)$, with $At(\varphi(\overline{x}_1, \overline{x}_2\sigma)) \subset At(\varphi(\overline{x}_1, \overline{x}_2))$. We show both implications $\tau \models \tau'$ and $\tau' \models \tau$ separately.

$[\tau \models \tau']$ Let $\langle I, J\rangle$ be a pair of source and target instance with $\langle I, J\rangle \models \tau$. If $I \not\models \varphi(\overline{x}_1, \overline{x}_2\sigma)$ then $\langle I, J\rangle \models \tau'$ holds vacuously. It remains to consider the case that $I \models \varphi(\overline{x}_1, \overline{x}_2\sigma)$ holds, i.e., there exists a substitution λ', s.t. $At(\varphi(\overline{x}_1\lambda', \overline{x}_2\sigma\lambda')) \subseteq I$. Consider the substitution $\lambda\colon \overline{x}_1 \cup \overline{x}_2 \to dom(I)$, s.t. $x\lambda = x\lambda'$ for every $x \in \overline{x}_1$ and $x\lambda = x\sigma\lambda'$ for every $x \in \overline{x}_2$. Then $At(\varphi(\overline{x}_1\lambda, \overline{x}_2\lambda)) = At(\varphi(\overline{x}_1\lambda', \overline{x}_2\sigma\lambda')) \subseteq I$ holds. Thus, we conclude that $J \models (\exists\overline{y}\,)\psi(\overline{x}_1\lambda, \overline{y}\,)$, since $\langle I, J\rangle \models \tau$ holds. Hence, since λ and λ' coincide on $\overline{x}_1$ also $J \models (\exists\overline{y}\,)\psi(\overline{x}_1\lambda', \overline{y}\,)$ and, therefore, $\langle I, J\rangle \models \tau'$.

$[\tau' \models \tau]$ Now let $\langle I, J\rangle$ be a pair of source and target instance with $\langle I, J\rangle \models \tau'$ and $J \models \varphi(\overline{x}_1, \overline{x}_2)$, i.e., there exists a substitution λ, s.t. $At(\varphi(\overline{x}_1\lambda, \overline{x}_2\lambda)) \subseteq I$. By definition of Rule 2, the inclusion $At(\varphi(\overline{x}_1, \overline{x}_2\sigma)) \subseteq At(\varphi(\overline{x}_1, \overline{x}_2))$ holds. Hence, $At(\varphi(\overline{x}_1\lambda, \overline{x}_2\sigma\lambda)) \subseteq I$ is true as well. But then, by $\langle I, J\rangle \models \tau'$, we know that also $J \models (\exists\overline{y}\,)\psi(\overline{x}_1\lambda, \overline{y}\,)$ holds and, therefore, $\langle I, J\rangle \models \tau$.

Rule 3. This rule is based on two general equivalences in first-order logic:

- $(\exists\overline{z}\,)(A(\overline{z}\,) \wedge B(\overline{z}\,)) \equiv (\exists\overline{z}_1)A(\overline{z}_1) \wedge (\exists\overline{z}_2)B(\overline{z}_2)$ if the variables $\overline{z}_1$ actually occurring in A and the variables $\overline{z}_2$ actually occurring in B are disjoint.
- We clearly have the equivalence $A \to (B_1 \wedge B_2) \equiv (A \to B_1) \wedge (A \to B_2)$.

Rule 4. Suppose that τ is deleted from Σ via Rule 4, i.e., Σ is transformed into Σ' with $\Sigma' = \Sigma \setminus \{\tau\}$ and $\Sigma' \models \tau$. Hence, $\Sigma' \models \Sigma$ holds. Moreover, $\Sigma \models \Sigma'$ holds by the monotonicity of "$\models$". Hence, $\Sigma \equiv \Sigma'$ clearly holds.
Rule 5. Suppose that $\tau \in \Sigma$ is replaced by τ' via Rule 5. Then τ' is of the form $\tau'\colon \varphi(\overline{x}) \to (\exists\overline{y}\,')\psi'(\overline{x}, \overline{y}\,')$, s.t. $At(\psi'(\overline{x}, \overline{y}\,')) \subset At(\psi(\overline{x}, \overline{y}\,))$. By the latter condition, $\tau \models \tau'$ clearly holds. Hence, Σ is equivalent to $(\Sigma \cup \{\tau'\})$. By the definition of Rule 5, $((\Sigma \cup \{\tau'\}) \setminus \{\tau\}) \models \tau$ holds. Hence, Σ is also equivalent to $(\Sigma \cup \{\tau'\}) \setminus \{\tau\}$. □

We will now move on to showing that our Rules 1 – 5 lead to a unique normal form. Two auxiliary lemmas, Lemma 4 and 5, will help us to arrive at this result, formulated as Theorem 1.

Lemma 4. *Let τ be an s-t tgd reduced w.r.t. the Rules 1 and 3 and let Σ be a set of s-t tgds. If $\Sigma \models \tau$, then one of the following two conditions is fulfilled: Either*

- *there exists a single s-t tgd $\tau_0 \in \Sigma$, s.t. $\tau_0 \models \tau$, or*
- *there exists a proper instance τ' of τ, s.t. $\Sigma \models \tau'$.*

Proof. Let $\{\tau_1, \ldots, \tau_k\} \subseteq \Sigma \setminus \{\tau\}$ with $\{\tau_1, \ldots, \tau_k\} \models \tau$. Suppose that k is minimal with this property and that $k \geq 2$. We show that then $\{\tau_1, \ldots, \tau_k\} \models \tau'$ holds for some proper instance τ' of τ. For $i \in \{1, \ldots, k\}$, let τ_i's be pairwise variable disjoint and have the form $\tau_i\colon\ \varphi_i(\overline{x}_i) \to (\exists\overline{y}_i\,)\psi_i(\overline{x}_i, \overline{y}_i)$. By Lemma 1 and the definition of Rule 4, the τ_i's fulfil the following properties:

(a) For every $i \in \{1, \ldots, k\}$, there exists a substitution $\lambda_i\colon \overline{x}_i \to \mathit{Const} \cup \overline{x}$, such that $At(\varphi_i(\overline{x}_i\lambda_i)) \subseteq At(\varphi(\overline{x}\,))$.

(b) A substitution $\mu\colon \overline{y} \to \mathit{Const} \cup \overline{x} \cup \bigcup_{i=1}^{k} \overline{y}_i$ exists, such that the following inclusion holds: $At(\psi(\overline{x}, \overline{y}\mu)) \subseteq \bigcup_{i=1}^{k} At(\psi_i(\overline{x}_i\lambda_i, \overline{y}_i))$.

Let $At(\psi(\overline{x}, \overline{y})) = \{A_1, \ldots, A_n\}$. Clearly, $n \geq k \geq 2$. Suppose that $\{A_1\mu, \ldots, A_\alpha\mu\} \subseteq At(\psi_1(\overline{x}_1\lambda_1, \overline{y}_1))$ while $\{A_{\alpha+1}\mu, \ldots, A_n\mu\} \subseteq \bigcup_{i=2}^{k} At(\psi_i(\overline{x}_i\lambda_i, \overline{y}_i))$. By assumption, τ is reduced w.r.t. Rule 3, i.e., the conclusion of τ either consists of a single atom without variables from $\overline{y}$ or of atoms forming a single connected component of the dual graph $\mathcal{G}(\tau)$. By $n \geq k \geq 2$, the former case can be excluded. Hence, the atoms in $\{A_1, \ldots, A_\alpha\}$ and $\{A_{\alpha+1}, \ldots, A_n\}$ share at least one variable $y \in \overline{y}$, i.e., y occurs in some atom A_i with $i \in \{1, \ldots, \alpha\}$ and in some atom A_j with $j \in \{\alpha + 1, \ldots, n\}$. Let $\ell \neq 1$ denote the index, s.t. $A_j\mu \in At(\psi_\ell(\overline{x}_\ell\lambda_\ell, \overline{y}_\ell))$. In total, we thus have $A_i\mu \in At(\psi_1(\overline{x}_1\lambda_1, \overline{y}_1))$ and, therefore, $y\mu \in \mathit{Const} \cup \overline{x} \cup \overline{y}_1$. On the other hand, $A_j\mu \in At(\psi_\ell(\overline{x}_\ell\lambda_\ell, \overline{y}_\ell))$ and, therefore, $y\mu \in \mathit{Const} \cup \overline{x} \cup \overline{y}_\ell$. By assumption, $\overline{y}_1$ and $\overline{y}_\ell$ are disjoint. Thus, $y\mu \in \mathit{Const} \cup \overline{x}$.

We construct the desired proper instance τ' of τ as follows: Let $\overline{y}\,' := \overline{y} \setminus \{y\}$ and define the substitution $\sigma\colon \overline{y} \to \mathit{Const} \cup \overline{x} \cup \overline{y}\,'$, s.t. $y\sigma = y\mu$ and σ maps all other variables in $\overline{y}$ onto themselves. Then we have $\sigma\mu = \mu$, i.e., for every $y_i \in \overline{y}$, $y_i\sigma\mu = y_i\mu$. Clearly, $\{\tau_1, \ldots, \tau_k\} \models \tau\mu$ and, therefore, also $\{\tau_1, \ldots, \tau_k\} \models \tau\sigma$ holds. But then τ' is the desired proper instance of τ. □

Lemma 5. *Let Σ be a set of s-t tgds and suppose that an s-t tgd $\tau \in \Sigma$ is reduced w.r.t. Rules 1 and 3 and that τ cannot be deleted via Rule 4. If there exists a proper instance τ' of τ, s.t. $\Sigma \models \tau'$ holds, then there exists an s-t tgd τ'', s.t. τ may be replaced by τ'' via Rule 5.*

Proof. Let $\tau'\colon \varphi(\overline{x}) \to (\exists\overline{y}\,')\psi'(\overline{x}, \overline{y}\,')$ and suppose that $\Sigma \models \tau'$ holds. Then there exist s-t tgds $\tau_1, \ldots, \tau_k$ in Σ of the form $\tau_i\colon \varphi_i(\overline{x}_i) \to (\exists\overline{y}_i)\psi_i(\overline{x}_i, \overline{y}_i)$, s.t. the conditions (a) and (b) of Lemma 1 are fulfilled, i.e.:

(a) For every $i \in \{1, \ldots, k\}$, there exists a substitution $\lambda_i\colon \overline{x}_i \to \mathit{Const} \cup \overline{x}$, such that $At(\varphi_i(\overline{x}_i\lambda_i)) \subseteq At(\varphi(\overline{x}))$.

(b) There exists a substitution $\mu\colon \overline{y} \to \mathit{Const} \cup \overline{x} \cup \bigcup_{i=1}^{k} \overline{y}_i$, such that the following inclusion holds: $At(\psi'(\overline{x}, \overline{y}\,'\mu)) \subseteq \bigcup_{i=1}^{k} At(\psi_i(\overline{x}_i\lambda_i, \overline{y}_i))$.

Let $\tau\colon \varphi(\overline{x}) \to (\exists\overline{y})\psi(\overline{x}, \overline{y})$. We claim that at least one of the τ_i coincides with τ (up to variable renaming). Suppose to the contrary that $\tau_i \in \Sigma \setminus \{\tau\}$ holds for every $i \in \{1, \ldots, k\}$. Then, the above conditions (a) and (b) imply that $\Sigma \setminus \{\tau\} \models \tau'$ holds by Lemma 1. Moreover, $\tau' \models \tau$ holds by Lemma 2, part (1). Thus, $\Sigma \setminus \{\tau\} \models \tau$ and τ could be deleted by Rule 4, which is a contradiction.

Let $I = \{i \mid 1 \leq i \leq k$, s.t. τ_i is obtained from τ via variable renaming$\}$. We define the CQ $\psi''(\overline{x}, \overline{y}\,'')$ of the s-t tgd $\tau''\colon \varphi(\overline{x}) \to (\exists\overline{y}\,'')\psi''(\overline{x}, \overline{y}\,'')$ as follows:

$\Theta = \{A(\overline{x}, \overline{y}) \mid A(\overline{x}, \overline{y}) \in At(\psi(\overline{x}, \overline{y}))$ and $\exists i$, s.t. $i \in I$ and $A(\overline{x}\lambda_i, \overline{y}) \in At(\psi'(\overline{x}, \overline{y}\,'\mu)) \cap At(\psi_i(\overline{x}_i\lambda_i, \overline{y}_i))\}$.

Moreover, we set $\psi''(\overline{x}, \overline{y}\,'') = \bigwedge_{A(\overline{x},\overline{y})\in\Theta} A(\overline{x}, \overline{y})$.

Clearly, $(\Sigma \setminus \{\tau\}) \cup \{\tau''\} \models \tau'$ by Lemma 1. Thus, also $(\Sigma \setminus \{\tau\}) \cup \{\tau''\} \models \tau$, by Lemma 2, part (1). We claim that $At(\psi''(\overline{x}, \overline{y}\,'')) \subset At(\psi(\overline{x}, \overline{y}\,))$ holds. Suppose to the contrary that $At(\psi''(\overline{x}, \overline{y}\,'')) = At(\psi(\overline{x}, \overline{y}\,))$. Then, by definition of Θ and by Lemma 1, $\tau \models \tau'$ would hold. By Lemma 2, part (2), this implies that τ is not reduced w.r.t. Rule 1, which is a contradiction. Hence, τ'' is indeed the desired s-t tgd, s.t. τ may be replaced by τ'' via Rule 5. □

We now define a normal form of s-t tgds via the rewrite rules of this chapter. We will then show that this normal form is unique up to isomorphism in the sense defined below.

Definition 7. *Let Σ be a set of s-t tgds and let Σ' be the result of applying the Rules 1 – 5 of Figure 3.1 exhaustively to Σ. Then Σ' is the* normal form *of Σ.*

Definition 8. *Let $\tau_1\colon\ \varphi_1(\overline{x}_1) \rightarrow (\exists \overline{y}_1\,)\psi_1(\overline{x}_1, \overline{y}_1)$ and $\tau_2\colon\ \varphi_2(\overline{x}_2) \rightarrow (\exists \overline{y}_2\,)\psi_2(\overline{x}_2, \overline{y}_2)$ be two tgds. We say that τ_1 and τ_2 are* isomorphic *if τ_2 is obtained from τ_1 via variable renamings $\eta\colon \overline{x}_1 \rightarrow \overline{x}_2$ and $\vartheta\colon \overline{y}_1 \rightarrow \overline{y}_2$.*

Let Σ_1 and Σ_2 be two sets of tgds. We say that Σ_1 and Σ_2 are isomorphic *if $|\Sigma_1| = |\Sigma_2|$, every $\tau_1 \in \Sigma_1$ is isomorphic to precisely one $\tau_2 \in \Sigma_2$ and every $\tau_2 \in \Sigma_2$ is isomorphic to precisely one $\tau_1 \in \Sigma_1$.*

We start by showing for two single s-t tgds τ_1 and τ_2 that logical equivalence and isomorphism coincide, provided that the s-t tgds are reduced via our rewrite rules. This result will then be extended to sets Σ_1 and Σ_2 of s-t tgds.

Lemma 6. *Let τ_1 and τ_2 be two s-t tgds that are reduced w.r.t. Rules 1 – 3. Then τ_1 and τ_2 are isomorphic, iff τ_1 and τ_2 are equivalent.*

Proof. The "⇒"-direction is an immediate consequence of Lemma 1. For the "⇐"-direction, consider two equivalent s-t tgds

$\tau_1\colon \varphi_1(\overline{x}_1, \overline{x}_2) \rightarrow (\exists \overline{y}\,)\psi_1(\overline{x}_1, \overline{y}\,)$ and
$\tau_2\colon\ \varphi_2(\overline{u}_1, \overline{u}_2) \rightarrow (\exists \overline{v}\,)\psi(\overline{u}_1, \overline{v}\,)$.

Observe that the antecedents of τ_1 and τ_2 must be homomorphically equivalent, i.e., by Lemma 1, there exist substitutions λ and ρ, s.t.

$\lambda\colon \overline{x}_1 \cup \overline{x}_2 \rightarrow \mathit{Const} \cup \overline{u}_1 \cup \overline{u}_2$, and
$\rho\colon \overline{u}_1 \cup \overline{u}_2 \rightarrow \mathit{Const} \cup \overline{x}_1 \cup \overline{x}_2$, such that
$At(\varphi_1(\overline{x}_1\lambda, \overline{x}_2\lambda)) \subseteq At(\varphi_2(\overline{u}_1, \overline{u}_2))$ and
$At(\varphi_2(\overline{u}_1\rho, \overline{u}_2\rho)) \subseteq At(\varphi_1(\overline{x}_1, \overline{x}_2))$.

We show that the antecedents of τ_1 and τ_2 are in fact isomorphic. In particular, we claim that there exist substitutions λ and ρ, s.t. the above inclusions are equalities, i.e.,

$\lambda\colon \overline{x}_1 \cup \overline{x}_2 \rightarrow \mathit{Const} \cup \overline{u}_1 \cup \overline{u}_2$, and
$\rho\colon \overline{u}_1 \cup \overline{u}_2 \rightarrow \mathit{Const} \cup \overline{x}_1 \cup \overline{x}_2$, such that
$At(\varphi_1(\overline{x}_1\lambda, \overline{x}_2\lambda)) = At(\varphi_2(\overline{u}_1, \overline{u}_2))$ and
$At(\varphi_2(\overline{u}_1\rho, \overline{u}_2\rho)) = At(\varphi_1(\overline{x}_1, \overline{x}_2))$.

Assume the converse is true, and, w.l.o.g., the antecedent of τ_1 cannot be mapped onto the entire antecedent of τ_2, i.e., for every substitution $\lambda_i \colon \overline{x}_1 \cup \overline{x}_2 \to \mathit{Const} \cup \overline{u}_1 \cup \overline{u}_2$, the inclusion $S_{\lambda_i} = \mathit{At}(\varphi_1(\overline{x}_1\lambda_i, \overline{x}_2\lambda_i)) \subset \mathit{At}(\varphi_2(\overline{u}_1, \overline{u}_2))$ holds. For every such λ_i, consider the s-t dependency of the form $\tau_1^i \colon \varphi_1(\overline{x}_1\lambda_i, \overline{x}_2) \to (\exists \overline{y}\,)\psi_1(\overline{x}_1\lambda_i, \overline{y}\,)$.

Let T denote the complete set of all such τ_1^i. It is easy to see that $\tau_1 \models \tau_2$ iff $T \models \tau_2$. Moreover, by construction of T, $\tau_1 \models T$ and, therefore, we get $\{\tau_1\} \equiv \{\tau_2\} \equiv T$.

Note also that the antecedent database of each $\tau_1^i \in T$ is an endomorphic image of the antecedent database of τ_2. Indeed, assume that for some j, there is no homomorphism projecting $\varphi_2(\overline{u}_1, \overline{u}_2)$ onto the antecedent $\varphi_1^j(\overline{x}_1^j, \overline{x}_2^j)$ of $\tau_1^j \in T$. Then, the combined instance $\langle \mathit{At}(\varphi_1^j(\overline{x}_1^j, \overline{x}_2^j), \emptyset\rangle$ satisfies τ_2 but not T, which is a contradiction.

By Lemma 4 we know that the following two cases are possible: either (i) some $\tau_1^i \models \tau_2$, or (ii) T implies a proper instance of τ_2.

The latter case contradicts Lemma 2, part (2), since we immediately get that τ_2 implies a proper instance of itself. We may therefore assume that (i) is the case: There exists some rule $\tau' \in T$ such that $\tau' \equiv \tau_2$ and moreover, the antecedent of τ' is a proper endomorphic image of the antecedent $\varphi_2(\overline{u}_1, \overline{u}_2)$ of τ_2.

We also assume that τ' is the smallest "endomorphic" (w.r.t. the antecedent of τ_2) dependency possible. Indeed, let τ' itself admit an equivalent s-t tgd with the proper endomorphically equivalent antecedent: then we just focus on this smaller s-t tgd instead of τ'. Thus, we consider the dependency τ^* whose antecedent $\varphi^*(\overline{u}_1^*, \overline{u}_2^*)$ is minimal in the following sense: no dependency logically equivalent to τ_2 (and thus to τ^*) exists, with the antecedent database being a proper endomorphic instance of $\mathit{At}(\varphi^*(\overline{u}_1^*, \overline{u}_2^*))$.

We show that also the conclusion of τ^* must be smaller than the conclusion of τ_2: that is, the inequality $|\mathit{At}(\varphi_2(\overline{u}_1, \overline{u}_2))| > |\mathit{At}(\varphi^*(\overline{u}_1^*, \overline{u}_2^*))|$ holds. Let σ be a substitution, such that $\varphi_2(\overline{u}_1\sigma, \overline{u}_2\sigma) = \varphi^*(\overline{u}_1^*, \overline{u}_2^*)$ holds. Since τ^* is minimal, there must also exist a substitution μ: $\mathit{At}(\psi^*(\overline{u}_1^*, \overline{v}^*\mu)) \subseteq \mathit{At}(\psi(\overline{u}_1\sigma, \overline{v}))$.

Note that σ necessarily "lumps together" two elements of $\overline{u}_1$: otherwise, τ_2 would be not reduced w.r.t. Rule 2 (Core of the antecedent). But then also the inequality $|\psi^*(\overline{u}_1^*, \overline{v}^*\mu)| < |\psi(\overline{u}_1\sigma, \overline{v})|$ holds. This means that there exists a dependency equivalent to τ_2 but with fewer conclusion atoms. This contradicts the assumption that $\{\tau_2\}$ is reduced w.r.t. Rule 5.

That is, we have shown that the antecedents of τ_2 and τ_1 are isomorphic. But since these dependencies are equivalent and reduced w.r.t. Rule 1 (Core of the conclusion), we also have that the entire τ_1 and τ_2 must be isomorphic as well. □

Theorem 1. *Let Σ_1 and Σ_2 be equivalent sets of s-t tgds, i.e., $\Sigma_1 \models \Sigma_2$ and $\Sigma_2 \models \Sigma_1$. Let Σ_1' and Σ_2' denote the normal form of Σ_1 and Σ_2, respectively. Then Σ_1' and Σ_2' are isomorphic.*

Proof. Let Σ_1 and Σ_2 be equivalent. Moreover, let Σ_1' and Σ_2' denote the normal form of Σ_1 and Σ_2, respectively. By the correctness of our rewrite rules 1 – 5, of course, also Σ_1' and Σ_2' are equivalent.

We first show that every s-t tgd in Σ_1' is isomorphic to some s-t tgd in Σ_2' and vice versa. Suppose to the contrary that this is not the case. W.l.o.g., we assume that there exists a $\tau \in \Sigma_1'$ which is not isomorphic to any s-t tgd in Σ_2'. By the equivalence of Σ_1' and Σ_2', the implication $\Sigma_2' \models \tau$ clearly holds. By Lemma 4, either $\tau_0 \models \tau$ for a single s-t tgd $\tau_0 \in \Sigma_2'$ or there exists a proper instance τ' of τ, s.t. $\Sigma_2' \models \tau'$.

We start by considering the case that $\tau_0 \models \tau$ for a single s-t tgd $\tau_0 \in \Sigma_2'$. By the equivalence of Σ_1' and Σ_2', the implication $\Sigma_1' \models \tau_0$ holds and we can again apply Lemma 4, i.e., either $\tau_1 \models \tau_0$ for a single s-t tgd $\tau_1 \in \Sigma_1'$ or there exists a proper instance τ_0' of τ_0, s.t. $\Sigma_1' \models \tau_0'$. Again we consider first the case that a single s-t tgd is responsible for the implication. Actually, if τ_1 were identical to τ then we had the equivalence $\tau_1 \models \tau$ and $\tau \models \tau_1$. Since both τ and τ_1 are reduced w.r.t. Rules 1 – 3, this would mean (by Lemma 6) that τ_1 and τ are isomorphic. This contradicts our original assumption that τ is not isomorphic to any s-t tgd in Σ_2'. Hence, the case that $\tau_1 \models \tau_0$ for a single s-t tgd $\tau_1 \in \Sigma_1'$ means that τ_1 is different from τ. In total, we thus have $\tau_1 \models \tau_0$ and $\tau_0 \models \tau$ and, therefore, $\tau_1 \models \tau$ for a s-t tgd $\tau_1 \in \Sigma_1' \setminus \{\tau\}$. Hence, τ can be deleted from Σ_1' via Rule 4, which contradicts the normal form of Σ_1'.

It thus remains to consider the cases that there exists a proper instance τ' of τ, s.t. $\Sigma_2' \models \tau'$ or there exists a proper instance τ_0' of τ_0, s.t. $\Sigma_1' \models \tau_0'$. We only show that the first one leads to a contradiction. The second case is symmetric. So suppose that there exists a proper instance τ' of τ, s.t. $\Sigma_2' \models \tau'$. By the equivalence of Σ_1' and Σ_2', we have $\Sigma_1' \models \Sigma_2'$ and, therefore, also $\Sigma_1' \models \tau'$. But then τ can be replaced in Σ_1' by τ' via Rule 5. Hence, by Lemma 5, τ can be replaced in Σ_1' by some s-t tgd τ'' via Rule 5. But this contradicts the assumption that Σ_1' is in normal form.

Hence, it is indeed the case that every s-t tgd in Σ_1' is isomorphic to *some* s-t tgd in Σ_2' and vice versa. We claim that every s-t tgd in Σ_1' is isomorphic to *precisely one* s-t tgd in Σ_2' and vice versa. Suppose to the contrary that there exists a s-t tgd τ which is isomorphic to two s-t tgds τ_1 and τ_2 in the other set. W.l.o.g., $\tau \in \Sigma_1'$ and $\tau_1, \tau_2 \in \Sigma_2'$. Clearly, τ_1 and τ_2 are isomorphic since they are both isomorphic to τ. Hence, $\tau_1 \models \tau_2$ and, therefore, $\Sigma_2' \setminus \{\tau_2\} \models \tau_2$, i.e., Rule 4 is applicable to Σ_2', which contradicts the assumption that Σ_2' is in normal form. □

We now consider the complexity of computing the normal form of a set of s-t tgds. Of course, the application of any of the Rules 1, 2, 4, and 5 is NP-hard, since they involve CQ answering. However, below we show that if the length of each s-t tgd (i.e., the number of atoms) is bounded by a constant, then the normal form according to Definition 7 can be obtained in polynomial time.

Note that a constant upper bound on the length of the s-t tgds is a common restriction in data exchange since, otherwise, even the most basic tasks like, computing a target instance fulfilling all s-t tgds, would be intractable.

Theorem 2. *Suppose that the length (i.e., the number of atoms) of the s-t tgds under consideration is bounded by some constant b. Then there exists an algorithm which reduces an arbitrary set Σ of s-t tgds to normal form in polynomial time w.r.t. the total size $||\Sigma||$ of (an appropriate representation of) Σ.*

Proof. Let constant b limit the number of atoms in each s-t tgd and let $||\Sigma|| = n$ denote the number of s-t tgds in Σ. Moreover, let α denote the maximum arity of the relation symbols in the source and target schema. We define the following simple algorithm:

1. Obtain Σ' by applying Rules 1 – 3 exhaustively to $\{\tau\}$, for each $\tau \in \Sigma$.
2. For each τ in the current set Σ' of s-t tgds do
   - Try to delete τ via Rule 4.
   - If τ was not deleted, try to replace τ by some τ' via Rule 5.
   - If Rule 5 was applicable, apply Rule 3 to τ'.

Note that the Rules 1 and 2 do not become applicable anymore after an application of Rule 5 provided that we replace τ by an s-t tgd τ' such that the set of atoms in the conclusion of τ' is minimal.

In order to establish the polynomial-time upper bound, we proceed in 2 steps. That is, we prove (1) an upper bound on the total number of rule applications and (2) an upper bound on the cost of each single rule application.

(1) *Total number of rule applications.* Rule 4 deletes an s-t tgd. Hence, it can be applied at most n times. The Rules 1, 2, and 5 delete at least one atom from an s-t tgd. Hence, in total, these rules can be applied at most $b * n$ times. Finally, Rule 3 splits the conclusion of an s-t tgd in 2 or more parts. Hence, also the total number of applications of Rule 3 is bounded by $b * n$. We thus get the upper bound $O(b * n)$ on the total number of applications of any rule. Moreover, it should be noted that at no stage of the algorithm, the current set Σ' of s-t tgds contains more than $b * n$ s-t tgds.

(2) *Cost of a single rule application.* In Rules 1 and 2, we compute the core of the atoms in the conclusion resp. in the antecedent. Rules 1 and 2 thus essentially come down to CQ answering of a query with $\leq b$ atoms over a database with $\leq b$ atoms. The cost of a single application of these rules is therefore in $O(\alpha b^b)$.

Rule 3 is the cheapest one in that it only requires the computation of the connected components of a graph with $\leq b$ vertices. For setting up this graph, we have to inspect at most $\alpha * b$ variable occurrences in an s-t tgd. The cost of an application of Rule 3 is thus in $O(\alpha b^2)$.

To apply Rule 4 to an s-t tgd τ_ℓ in the current set Σ', we compare $\tau_\ell \colon \varphi(\overline{x}\,) \rightarrow (\exists \overline{y}\,)\psi(\overline{x}, \overline{y}\,)$ with every $\tau_i \in \Sigma'$, such that $i \neq \ell$. With $\tau_i \colon \varphi_i(\overline{x}_i) \rightarrow (\exists \overline{y}_i\,)\psi_i(\overline{x}_i, \overline{y}_i)$, we proceed as follows:

1. For each $i \neq \ell$, compute all possible substitutions λ_{ij}, s.t. $At(\varphi_i(\overline{x}_i\lambda_{ij})) \subseteq At(\varphi(\overline{x}\,))$.

   Every such λ_{ij} is uniquely determined by an assignment of the $\leq b$ atoms of $\varphi_i(\overline{x}_i)$ to the $\leq b$ atoms of $\varphi(\overline{x}\,)$. Hence, for every i, there are at most b^b possible substitutions λ_{ij}.

2. For all i, j, compute $\mathcal{A}_{ij} = At(\psi_i(\overline{x}_i\lambda_{ij}, \overline{y}_{ij}))$, where $\overline{y}_{ij}$ is a set of fresh variables, i.e., we apply the substitutions λ_{ij} computed in the first step to the conclusion of τ_i and rename the variables $\overline{y}_i$ apart.

3. Let $\mathcal{A} = \bigcup_{i \neq \ell} \bigcup_j \mathcal{A}_{ij}$. Try to find a substitution μ, s.t. $At(\psi(\overline{x}, \overline{y}\mu)) \subseteq \mathcal{A}$. If such a μ exists, delete τ_ℓ.

In Step 1, we compute all solutions of a CQ with $\leq b$ atoms over a database with $\leq b$ atoms. In total, we apply this step to at most b^2n^2 pairs (τ_ℓ, τ_i), which is feasible in total time $O(b^2n^2\alpha b^b)$. As a result, we get $\mathcal{A}$ as the union of at most $b^2n^2b^b$ sets $\mathcal{A}_{ij}$, each consisting of $\leq b$ atoms. Hence, $\mathcal{A}$ contains $\leq n^2b^{b+3}$ atoms. Step 2 is then feasible in time $O(\alpha n^2 b^{b+3})$. Finally, in Step 3, we have to evaluate a Boolean CQ with $\leq b$ atoms over a database $\mathcal{A}$ consisting of $\leq n^2b^{b+3}$ atoms. This is feasible in $O(\alpha(n^2b^{b+3})^b)$. In total, the entire computation required for an application of Rule 4 thus fits into time $O(||\Sigma||^{f(b)})$ for some function $f(.)$, which depends only on b but not on the size of the input.

An application of Rule 5 is very similar to Rule 4. The first two steps above are identical. Only in Step 3 we do not search for a μ with $At(\psi(\overline{x}, \overline{y}\mu)) \subseteq \mathcal{A}$. At this stage, we know that such a μ does not exist. Instead, we search for a μ, s.t. $At(\psi(\overline{x}, \overline{y}\mu)) \subseteq \mathcal{A} \cup At(\psi(\overline{x}, \overline{y}))$ and $At(\psi(\overline{x}, \overline{y}\mu)) \cap \mathcal{A} \neq \emptyset$. If such a μ exists, we choose μ, s.t. $At(\psi(\overline{x}, \overline{y}\mu)) \cap \mathcal{A}$ is maximal (w.r.t. set inclusion). Note that the desired s-t tgd τ' for replacing τ is obtained as $\tau'\colon \varphi(\overline{x}\,) \rightarrow (\exists \overline{y}\,')\psi'(\overline{x}, \overline{y}\,')$, s.t. $At(\psi'(\overline{x}, \overline{y}\,')) = At(\psi(\overline{x}, \overline{y}\,)) \setminus \{A \mid A \in At(\psi(\overline{x}, \overline{y}\,)) \text{ and } A\mu \in \mathcal{A}\}$. In other words, the set of atoms in the conclusion of τ' becomes minimal if $At(\psi(\overline{x}, \overline{y}\mu)) \cap \mathcal{A}$ is maximized. Clearly, Steps 1 and 2 above do not have to be repeated for Rule 5. Step 3 of an application of Rule 5 boils down to essentially the same kind of CQ evaluation as for Rule 4. We thus end up again with an upper bound of $O(||\Sigma||^{g(b)})$ on the computation time, where $g(.)$ is a function, which depends only on b but not on the size of the input. □

The restriction on the number of atoms in each s-t tgd is used in the above proof only in order to show that each rule application is feasible in polynomial time. The argument that the total number of rule applications is bounded by the total number of atoms in all s-t tgds in Σ applies to any set Σ of s-t tgds. We thus get:

Corollary 1. *The rewrite rule system consisting of Rules 1 – 5 is terminating, i.e., given an arbitrary set Σ of s-t tgds, the non-deterministic, exhaustive application of the Rules 1 – 5 terminates.*

It can be shown that the unique normal form produced by our rewrite rules is indeed *optimal*.

Theorem 3. *A set Σ of s-t tgds in normal form is optimal according to Definition 3.*

Proof. The proof proceeds in three stages:

(1) Σ is split-reduced. Suppose to the contrary that it is not. Then there exists a set Σ' with $\Sigma \equiv \Sigma'$, $|\Sigma| < |\Sigma'|$ and $\mathit{ConSize}(\Sigma) = \mathit{ConSize}(\Sigma')$. Let Σ^* denote the result of exhaustively applying our rewrite rules to Σ'. By Theorem 1, Σ and Σ^* are isomorphic. Hence, we have $\mathit{ConSize}(\Sigma) = \mathit{ConSize}(\Sigma^*)$ and $|\Sigma| = |\Sigma^*|$. An inspection of the Rules 1 – 5 reveals that they may possibly decrement the value of $\mathit{ConSize}()$ (by either deleting an atom in the conclusion or deleting an entire s-t tgd) but they never increment the value of $\mathit{ConSize}()$. By $\mathit{ConSize}(\Sigma) = \mathit{ConSize}(\Sigma')$ together with $\mathit{ConSize}(\Sigma) = \mathit{ConSize}(\Sigma^*)$, we immediately have $\mathit{ConSize}(\Sigma') = \mathit{ConSize}(\Sigma^*)$. Hence, when transforming Σ' into Σ^*, we never decrement $\mathit{ConSize}()$ and, thus, we never delete an s-t tgd. But then $|\Sigma'| = |\Sigma^*|$ and, therefore, $|\Sigma'| = |\Sigma|$, which contradicts the assumption that $|\Sigma| < |\Sigma'|$ holds.

(2) $\mathit{ConSize}(\Sigma)$ and $\mathit{VarSize}(\Sigma)$ are minimal. It is easy to verify that by no application of any of the Rules 1 – 5 the parameters $\mathit{ConSize}()$ or $\mathit{VarSize}()$ can increase, i.e., if a set Υ of s-t tgds is obtained from some set Υ' by an application of one of the Rules 1 – 5, then $\mathit{ConSize}(\Upsilon) \leq \mathit{ConSize}(\Upsilon')$ and $\mathit{VarSize}(\Upsilon) \leq \mathit{VarSize}(\Upsilon')$.

Now let Σ' be a set of s-t tgds equivalent to Σ, and let Σ^* denote the result of exhaustively applying our rewrite rules to Σ'. By Theorem 1, Σ and Σ^* are isomorphic. Hence, we have the following relations: $\mathit{ConSize}(\Sigma) = \mathit{ConSize}(\Sigma^*) \leq \mathit{ConSize}(\Sigma')$ and also $\mathit{VarSize}(\Sigma) = \mathit{VarSize}(\Sigma^*) \leq \mathit{VarSize}(\Sigma')$.

(3) $|\Sigma|$ and $\mathit{AntSize}(\Sigma)$ are minimal. Let Σ' be an arbitrary split-reduced set of s-t tgds equivalent to Σ. We first show that $|\Sigma| \leq |\Sigma'|$. Suppose to the contrary that $|\Sigma| > |\Sigma'|$. We derive a contradiction by showing that then Σ' is not split-reduced. By (2), we know that $\mathit{ConSize}(\Sigma) \leq \mathit{ConSize}(\Sigma')$ holds. Analogously to the proof of Lemma 7, we can transform Σ into $\bar{\Sigma}$ with $\mathit{ConSize}(\bar{\Sigma}) = \mathit{ConSize}(\Sigma')$ simply by choosing an s-t tgd τ in Σ and inflating its conclusion by sufficiently many atoms of the form $P(u_1, \ldots, u_k)$. In total, we then have $\bar{\Sigma} \equiv \Sigma'$, $\mathit{ConSize}(\bar{\Sigma}) = \mathit{ConSize}(\Sigma')$, and $|\bar{\Sigma}| > |\Sigma'|$. Hence, Σ' is not split-reduced.

It remains to prove $\mathit{AntSize}(\Sigma) \leq \mathit{AntSize}(\Sigma')$ as the final inequality. It is easy to verify that the parameter $\mathit{AntSize}()$ can never increase when one of the Rules 1, 2, 4, or 5 is applied. Moreover, by Lemma 7, we know that Rule 3 is never applicable when we transform a split-reduced set of s-t tgds into normal form. Now let Σ^* denote the normal form of Σ'. By Theorem 1, Σ and Σ^* are isomorphic. Hence, we have $\mathit{AntSize}(\Sigma) = \mathit{AntSize}(\Sigma^*) \leq \mathit{AntSize}(\Sigma')$. □

An important motivation for seeking a conclusion-minimal mapping Σ is to keep the redundancies in the target instance small when using Σ in data exchange. The following theorem establishes that our normal form indeed serves this purpose.

Theorem 4. *Let $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$ be a schema mapping where Σ is a set of s-t tgds and Σ is in normal form. Moreover, let Σ' be another set of s-t tgds, s.t. Σ and Σ' are equivalent and let I be an arbitrary source instance. Then there exists a variable renaming λ on the variables in the canonical universal solution $\mathit{CanSol}(I, \Sigma)$, s.t. $\mathit{CanSol}(I, \Sigma)\lambda \subseteq \mathit{CanSol}(I, \Sigma')$ holds, i.e., the canonical instance produced by Σ is subset-minimal up to variable renaming.*

Proof. It is easy to verify that every application of any of the Rules 1 – 5 either leaves the corresponding canonical universal solution unchanged or prevents the introduction of some atoms in the canonical universal solution, i.e., let the set Υ of s-t tgds be obtained from some set Υ' by an application of one of the Rules 1 – 5, then there exists a substitution μ, s.t. $CanSol(I, \Upsilon)\mu \subseteq CanSol(I, \Upsilon')$ holds. The theorem follows by an induction argument. □

Before we conclude this chapter, two remarks on the splitting rule are in order:

(1) The purpose of the splitting rule is to enable a further simplification of the antecedents of the resulting s-t tgds. Of course, it may happen that no further simplification is possible. As an example, consider a schema mapping $\Sigma = \{R(x, y) \wedge R(y, z) \rightarrow S(x, z) \wedge T(z, x)\}$. Splitting yields $\Sigma' = \{R(x, y) \wedge R(y, z) \rightarrow S(x, z); R(x, y) \wedge R(y, z) \rightarrow T(z, x)\}$, which cannot be further simplified. In cases like this, one may either "undo" the splitting or simply keep track of s-t tgds with identical (possibly up to variable renaming) antecedents in order to avoid multiple evaluation of the same antecedent by the chase.

(2) Definition 2 gives a "semantical" definition of "split reduced" while the splitting rule is a "syntactical" criterion. The following lemma establishes the close connection between them.

Lemma 7. *Let Σ be a split-reduced set of s-t tgds and let Σ^* denote the normal form of Σ. Then, for every possible sequence of rewrite rule applications, this normal form Σ^* is obtained from Σ without ever applying Rule 3 (i.e., splitting).*

Proof. Suppose to the contrary that there exists a sequence of rewrite rule applications including the splitting rule on the way from Σ to Σ^*. An inspection of the rewrite rules shows that an application of Rule 4 (i.e., deletion of an s-t tgd) is never required as a precondition in order to be able to apply another rule. Hence, w.l.o.g., we may assume that Rule 4 is applied at the very end of the transformation of Σ into Σ^*, so that Rule 4 does not precede the application of any other rule. Let $\Sigma_0, \ldots, \Sigma_n$ with $\Sigma_0 = \Sigma$ and $\Sigma_n = \Sigma^*$ denote the sequence of intermediate results along this transformation of Σ into Σ^*. Then there exists an $i \in \{1, \ldots, n\}$, s.t. Σ_i is obtained from Σ_{i-1} by an application of Rule 3. Moreover, suppose that this is the first application of Rule 3 along this transformation of Σ into Σ^*. Since we are assuming that all applications of Rule 4 occur at the very end of this transformation from Σ to Σ^*, we have $|\Sigma_{i-1}| = |\Sigma|$ and, therefore, $|\Sigma_i| > |\Sigma|$. An inspection of the Rules 1, 2, 3, and 5 reveals that they may possibly decrement the value of $ConSize()$ (by deleting an atom in the conclusion via Rule 1 or 5) but they never increment the value of $ConSize()$. Hence, we have $ConSize(\Sigma_i) \leq ConSize(\Sigma)$. We derive a contradiction by constructing a set Σ' equivalent to Σ_i (and, hence, to Σ), with $ConSize(\Sigma') = ConSize(\Sigma)$ and $|\Sigma'| > |\Sigma|$. In other words, we show that Σ is not split-reduced.

Let τ with $\tau\colon \varphi(\overline{x}) \rightarrow \exists \overline{y}\ \psi(\overline{x}, \overline{y})$ be an arbitrary s-t tgd in Σ_i and let $P(z_1, \ldots, z_k)$ with $\{z_1, \ldots, z_k\} \subseteq \overline{x} \cup \overline{y}$ be an atom in the conclusion of τ. Clearly, we may add atoms of the form $P(u_1, \ldots, u_k)$ for fresh, existentially quantified variables $u_1, \ldots, u_k$ to the conclusion without

changing the semantics of Σ. Indeed, any such atom could be removed again by our Rule 1 (Core of the conclusion). Then we transform Σ_i into Σ' with $ConSize(\Sigma') = ConSize(\Sigma_i)$ simply by inflating the conclusion of τ in Σ_i by sufficiently many atoms of the form $P(u_1, \ldots, u_k)$. In total, we then have $\Sigma' \equiv \Sigma_i \equiv \Sigma$, $ConSize(\Sigma') = ConSize(\Sigma)$, and $|\Sigma'| = |\Sigma_i| > |\Sigma'|$. Hence, Σ is not split-reduced, which contradicts the assumption of this lemma. □

If a mapping Σ contains redundancies in the sense that one of the Rules 1, 4, 5 is applicable, then the notion of "split-reduced" according to Definition 2 and the non-applicability do not necessarily coincide as the following example illustrates. However, if Rules 1, 4, 5 are not applicable, then Definition 2 is exactly captured by the splitting rule (Rule 3), see Lemma 8.

Example 13. *Consider the set $\Sigma = \{\tau\}$ of s-t tgds with $\tau\colon P(x_1, x_2) \rightarrow (\exists y_1, y_2)R(x_1, x_2, y_1) \wedge R(x_1, y_1, y_2)$. On the one hand, Rule 3 is not applicable because the conclusion of τ consists of a single connected component. On the other hand, τ may be also simplified (via Rule 1 or Rule 5) to the form $\tau'\colon P(x_1, x_2) \rightarrow (\exists y)R(x_1, x_2, y)$. Now let Σ' consist of two "copies" of τ', i.e., $\Sigma' = \{\tau', \tau''\}$ with $\tau''\colon P(z_1, z_2) \rightarrow (\exists y)R(z_1, z_2, y)$. Then we have the equivalence $\Sigma \equiv \Sigma'$. Moreover, $|\Sigma'| > |\Sigma|$ and $ConSize(\Sigma') = ConSize(\Sigma)$. Hence, Σ is not split-reduced in the sense of Definition 2.*

Lemma 8. *Let Σ be a set of s-t tgds, s.t. Σ is reduced w.r.t. Rules 1, 4, 5. Then the following equivalence holds: Σ is split-reduced (according to Definition 2) iff Rule 3 (i.e., splitting) is not applicable.*

Proof. If Rule 3 is applicable to Σ, then Σ can obviously be transformed into an equivalent set Σ' with $|\Sigma'| > |\Sigma|$ and $ConSize(\Sigma') = ConSize(\Sigma)$, i.e., Σ is not split-reduced according to Definition 2.

Now suppose that the splitting rule is not applicable to Σ. We have to show that then Σ is split-reduced. Suppose to the contrary that it is not split-reduced, i.e., there exists an equivalent set Σ' with $|\Sigma'| > |\Sigma|$ and $ConSize(\Sigma') = ConSize(\Sigma)$. We derive a contradiction by exploiting Theorem 1 (i.e., the uniqueness of the normal form according to Definition 7).

First, we observe that the normal form Σ^* of Σ can be obtained via Rule 2 only. Indeed, by assumption, none of Rules 1, 3, 4, 5 is applicable to Σ. Hence, either Σ already is in normal form (i.e., Rule 2 is not applicable either) or Σ can be simplified via Rule 2. Clearly, an application of Rule 2 does not enable the application of any of the other rules. Hence, Σ^* is obtained by iterated applications of Rule 2 only. Note that Rule 2 has no influence on the cardinality and on the conclusion-size of a mapping. Hence, we have $|\Sigma| = |\Sigma^*|$ and $ConSize(\Sigma) = ConSize(\Sigma^*)$.

Second, let us transform Σ' into normal form. By Theorem 1, this normal form is unique up to isomorphism. Hence, w.l.o.g., this normal form of Σ' is Σ^*. As far as the cardinality of the involved mappings is concerned, we have $|\Sigma'| > |\Sigma|$ and $|\Sigma| = |\Sigma^*|$. Hence, during the transformation of Σ' into $|\Sigma^*|$, eventually Rule 4 or 5 must be applied thus reducing the conclusion-size. Hence, we have $ConSize(\Sigma') > ConSize(\Sigma^*)$. But this is a contradiction to the above equalities $ConSize(\Sigma') = ConSize(\Sigma)$ and $ConSize(\Sigma) = ConSize(\Sigma^*)$. □

CHAPTER 4

Normalization in the presence of target egds

We now extend our rewrite rule system to schema mappings with both s-t tgds and target egds. Several additional considerations and measures are required to arrive at a unique normal form and a basis for the s-t tgd optimization also in this case. The outline of this chapter is as follows:

(1) We have already seen in Example 6 that the presence of egds may have an effect on the equivalence between two sets of s-t tgds. We shall therefore first present a method of "propagating" the effects of the egds into the s-t tgds in Section 4.1.

(2) Splitting has played an important role in all our considerations so far. It will turn out that splitting via Rule 3 as in the tgd-only case is not powerful enough if egds are present. We shall therefore present a generalization of the notion of "split-reduced" and of the splitting rule to the case when also egds are present. This will lead to the notion of "egd-split-reduced" mappings in Section 4.2.

(3) The intuition of "egd-split-reduced" mappings is that it is not possible to generate the atoms in the conclusion of some tgd by means of several tgds. The antecedent may thus possibly be left unchanged. It can easily be shown that, in general, there does not exist a unique "egd-split-reduced" normal form. Therefore, in Section 4.3 we restrict this notion to "antecedent-split-reduced" mappings, i.e.: a tgd is replaced by new tgds only if the new tgds have strictly smaller antecedents than the original one. With this concept, we shall manage to prove that there always exists a unique (up to isomorphism) normal form also in the presence of egds.

(4) Finally, we leave aside the considerations on splitting and concentrate on the optimization of the set of s-t tgds according to the criteria of Chapter 3. We shall show in Section 4.4 that grouping the s-t tgds by homomorphically equivalent antecedents is the key to any optimization tasks in this area.

(5) We also look at the operation opposite to splitting: namely, merge of s-t tgds with homomorphically equivalent antecedents. As we will show, unlike the s-t tgds only case, in the presence of target dependencies the splitting of s-t tgds can cause an increase in the total number of conclusion atoms. Hence, for some cases the merge operation can be a reasonable alternative. We will show, however, that with respect to the unique normal form, the "merged" form of the mappings is no more useful than the "egd-split-reduced" form.

4.1 Propagating the effect of egds into s-t tgds

An important complication introduced by the egds has already been hinted at in Chapter 1, namely the equivalence of two sets of s-t tgds may be affected by the presence of egds:

Example 14. *(Example 6 slightly extended).*

$$\begin{aligned}\Sigma_{st} = \{&C(x_1,x_2,x_3) \rightarrow (\exists y_1,y_2)\ P(y_1,y_2,y_2) \wedge P(y_1,x_2,x_3)\\ &C(x_1,x_2,x_3) \rightarrow (\exists y_1) P(y_1,x_3,x_2)\\ &C(x_1,x_2,x_2) \rightarrow Q(x_1)\}\end{aligned}$$

$$\begin{aligned}\Sigma'_{st} = \{&C(x_1,x_2,x_3) \rightarrow (\exists y_1)\ P(y_1,x_2,x_3)\\ &C(x_1,x_2,x_3) \rightarrow Q(x_1)\}\end{aligned}$$

$$\Sigma_t = \{P(x_1,x_2,x_3) \rightarrow x_2 = x_3\}$$

We have $\Sigma_{st} \cup \Sigma_t \equiv \Sigma'_{st} \cup \Sigma_t$. Moreover, both Σ_{st} and Σ'_{st} are in normal form w.r.t. the Rules 1 – 5 from Chapter 3. However, $\Sigma_{st} \not\equiv \Sigma'_{st}$ holds.

In contrast, the equivalence of two sets of target egds is not influenced by the presence of s-t tgds, as the following lemma shows.

Lemma 9. *Suppose that $\Sigma = \Sigma_{st} \cup \Sigma_t$ and $\Upsilon = \Upsilon_{st} \cup \Upsilon_t$ are two logically equivalent sets of s-t tgds and target egds. Then, Σ_t and Υ_t are equivalent.*

Proof. W.l.o.g. assume that there exists an $\varepsilon : \varphi(\overline{x}) \rightarrow \sigma(\overline{x}) \in \Upsilon_t$ s.t. $\Sigma_t \not\models \varepsilon$. That is, the set $L = chase(At(\varphi(\overline{x})), \Sigma_t)$ of atoms of the antecedent of ε chased with Σ_t does not satisfy ε. However, it does satisfy Σ_t. Now, consider the pair of instances $\langle \emptyset, L \rangle$. Since $L \models \Sigma_t$, $\langle \emptyset, L \rangle \models \Sigma$ and $\langle \emptyset, L \rangle \not\models \Upsilon$, which is a contradiction. □

In order to work with logical equivalence, we need a way to test logical implication of schema mappings. However, since we are now dealing with s-t tgds and egds, the declarative implication criterion from Lemma 1 no longer works. Instead, we take the chase-based procedure by Beeri and Vardi [7], applicable to any embedded dependencies that cannot cause an infinite chase (which is clearly the case when all tgds are s-t tgds).

Lemma 10. *[7] Let Σ be a set of tgds and egds and let δ be either a tgd or an egd. Let $\varphi(\overline{x}\,)$ denote the antecedent of δ and let T denote the database obtained by chasing $At(\varphi(\overline{x}\,))$ with Σ. The variables in $\overline{x}$ are considered as labelled nulls. Then $\Sigma \models \delta$ iff $T \models \delta$ holds.*

Procedure PROPAGATE
Input: Mapping $\Sigma = \Sigma_{st} \cup \Sigma_t$, conjunction $\varphi(\bar{x})$
Output: Sets of dependencies $\Delta_s^{\Sigma}(\varphi(\bar{x}))$ and $\Delta_{st}^{\Sigma}(\varphi(\bar{x}))$

/* 1. chase with $\Sigma = \Sigma_{st} \cup \Sigma_t$ */
$I := At(\varphi(\bar{x}\,))$;
$J := chase(I, \Sigma)$;
if chase fails having to unify $c_1 = c_2$, two distinct constants occurring in $\varphi(\bar{x})$ or in Σ **then**
return $(\{\varphi(\bar{x}) \rightarrow c_1 = c_2\}, \emptyset)$

/* 2. compute s-t tgd τ */
let $J = J_S \cup J_T$, s.t. J_S is an instance over $\mathbf{S}$ and J_T is an instance over $\mathbf{T}$;
let $J^* = core(J_T)$, where the variables that occur in J_S are considered as constants.
$\tau := \big(\bigwedge_{A \in J_S} A\big) \rightarrow (\exists \bar{y}\,) \bigwedge_{B \in J^*} B$;
$\Delta_{st} := \{\tau\}$;

/* 3. compute source egds */
$\Delta_s := \emptyset$;
Compute a substitution λ s.t. $At(\varphi(\bar{x}\lambda)) = J_S$;
for each pair of variables $x_j, x_k \in \bar{x}$ **do**
if $x_j\lambda = x_k\lambda$ **then**
$\Delta_s := \Delta_s \cup \{\varphi(\bar{x}\,) \rightarrow x_j = x_k\}$;

/* 4. output result */
return (Δ_s, Δ_{st});

Figure 4.1: PROPAGATE Procedure.

Analogously to Rule 4 in Figure 3.1, we also need a rule for deleting redundant tgds in the presence of target egds. We shall refer to this rule as the Rule E1 in the rewrite rule system to be constructed in this chapter, which is specified in Figure 4.2 (Section 4.3). As in the tgd-only case, the primary goal of such a rewrite rule system is the definition of a unique normal form of the s-t tgds – but now taking also the target egds into account. The first step towards this goal is to incorporate the effects of egds into s-t tgds. As we have already pointed out in Chapter 1, this may require the introduction of source egds. Since we only consider source instances containing no variables, there will be no source chase. The source egds are only meant to capture the failure conditions which cannot be detected otherwise after the rewriting of the s-t tgds.

In Figure 4.1, we present the procedure PROPAGATE, which allows to incorporate, to some extent, the effect of the target egds into the s-t tgds and thereby possibly generates source egds. The idea of this procedure is that, given an s-t tgd τ, we identify all egds in a given mapping Σ that will be applicable in the chase with Σ whenever τ is. Moreover, we want that all equalities enforced by these egds should already be enforced in the s-t tgd.

Definition 9 (Rewriting with PROPAGATE)**.** *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ be a set of source-to-target and*

target embedded dependencies. We say that the following set $\Sigma^* = \Sigma_s \cup \Sigma^*_{st} \cup \Sigma_t$ *is the* rewriting *of* Σ *with* PROPAGATE *if:*

- Σ_s *is a set of all source egds produced by applying* PROPAGATE *to all pairs* $\langle \varphi_i, \Sigma \rangle$*, where* φ_i *is the antecedent of an s-t tgd in* Σ_{st}*,*
- Σ^*_{st} *is a set of all s-t tgds produced by applying* PROPAGATE *to all pairs* $\langle \varphi_i, \Sigma \rangle$*, where* φ_i *is the antecedent of an s-t tgd in* Σ_{st}*.*

Note that the chase in step 1 of PROPAGATE is not the usual chase in data exchange. Here, in order to propagate backwards the effect of the target egds, we chase the database $I = At(\varphi(\overline{x}))$ with labelled nulls, which instantiate the variables from $\overline{x}$.

Example 15. *We apply the* PROPAGATE *procedure to each s-t tgd in the set of dependencies* $\Sigma = \Sigma_{st} \cup \Sigma_t$ *from Example 14. We start with the first tgd of* Σ_{st}*:*

$\tau\colon\ C(x_1, x_2, x_3) \rightarrow (\exists y_1, y_2)\ P(y_1, y_2, y_2) \wedge P(y_1, x_2, x_3).$

(a) $I := \{C(x_1, x_2, x_3)\}$*. (We now consider every* x_i *as a labelled null).*

(b) Chasing $\langle I, \emptyset \rangle$ *with* Σ_{st} *yields the instance*

$I' = \{C(x_1, x_2, x_3), P(y'_1, y'_2, y'_2), P(y'_1, x_2, x_3), P(y''_1, x_3, x_2)\}.$

The egd of Σ_t *is then applied, resulting in*

$I'' = \{C(x_1, x_2, x_2), P(y'_1, y'_2, y'_2), P(y'_1, x_2, x_2), P(y''_1, x_2, x_2)\}.$

Note, that the egd application affected the "source" atom C*. Now, the third tgd in* Σ_{st} *becomes applicable, producing the ultimate instance*

$J = \langle J_S, J_T \rangle = chase(\langle I, \emptyset \rangle, \Sigma) =$
$\{C(x_1, x_2, x_2), P(y'_1, y'_2, y'_2), P(y'_1, x_2, x_2), P(y''_1, x_2, x_2), Q(x_1)\}.$

(c) We got instances

$J_S = \{C(x_1, x_2, x_2)\}$ *and*

$J_T = \{P(y'_1, y'_2, y'_2), P(y'_1, x_2, x_2), P(y''_1, x_2, x_2), Q(x_1)\}.$

Core computation of J_T *yields*

$J^* = \{P(y'_1, x_2, x_2), Q(x_1)\}.$

The s-t tgd τ *is thus transformed into the following tgd*

$\tau'\colon\ C(x_1, x_2, x_2) \rightarrow (\exists y'_1)\ P(y'_1, x_2, x_2) \wedge Q(x_1).$

(d) We compute the substitution $\lambda = \{x_3 \leftarrow x_2\}$*, which maps the only atom* $C(x_1, x_2, x_3)$ *in* $\varphi(\overline{x}\,)$ *onto instance* $J_S = \{C(x_1, x_2, x_2)\}$*. Hence, we get one source egd*

$C(x_1, x_2, x_3) \rightarrow x_2 = x_3.$

Finally, we obtain the sets

$\Delta^{\Sigma}_{st}(C(x_1, x_2, x_3)) = \{C(x_1, x_2, x_2) \rightarrow (\exists y'_1)\, P(y'_1, x_2, x_2) \wedge Q(x_1)\}$ *and*

$\Delta^{\Sigma}_{s}(C(x_1, x_2, x_3)) = \{C(x_1, x_2, x_3) \rightarrow x_2 = x_3\}.$

Since the antecedents of the other two s-t tgds in Σ_{st} *coincide either with* $ant(\tau)$ *or with the antecedent of the tgd in* $\Delta^{\Sigma}_{st}(C(x_1, x_2, x_3))$*, the set* $\Delta^{\Sigma}_{s}(C(x_1, x_2, x_3)) \cup \Delta^{\Sigma}_{st}(C(x_1, x_2, x_3)) \cup \Sigma_t$ *is the rewriting of* Σ *with* PROPAGATE*.*

Rewriting with the PROPAGATE procedure never increases the size of the antecedents of s-t tgds. Hence, the cost of the join-operations when computing the canonical universal solution is not affected. On the other hand, the size of the conclusions is normally increased by this procedure. Note however that all atoms thus accumulated in the conclusion of some s-t tgd τ would be generated in a target instance anyway, whenever τ fires. Deletion of redundant atoms from the conclusion of the s-t tgds (via a rule similar to Rule 5 from Figure 3.1) will be a topic of the next sections.

At this point, we will prove the correctness of rewriting with the PROPAGATE procedure. The proof will work even for the case when the set of target constraints contains tgds, under the assumption that these tgds *never cause infinite chase sequences*. We will start with showing an auxiliary lemma.

Lemma 11. *Let $\Delta_s^\Sigma(\varphi(\bar{z})) \cup \Delta_{st}^\Sigma(\varphi(\bar{z}))$ result from applying* PROPAGATE *to a CQ $\varphi(\bar{z})$ and a set of embedded dependencies Σ. Assume that $\Delta_{st}^\Sigma(\varphi(\bar{z}))$ is not empty and thus contains a tgd $\varphi(\bar{x}) \rightarrow (\exists \bar{y})\ \psi(\bar{x}, \bar{y})$, where $\bar{x} = \bar{z}\lambda$ for some substitution λ, and let I_φ denote a frozen database of $At(\varphi(\bar{x}))$. Then, the chase of I_φ with Σ succeeds.*

Proof. Note that $\Delta_{st}^\Sigma(\varphi(\bar{z}))$ being not empty witnesses the fact that the chase of $\varphi(\bar{z})$ does not fail due to unification of some constants present in φ or in Σ (see step 1 of PROPAGATE). Moreover, all equalities which are enforced by Σ on the elements of $\overline{z}$ are accumulated in the substitution λ. It suffices to note that $\overline{x} = \overline{z}\lambda$, and that I_φ has been produced from $\varphi(\overline{x})$ by assigning a fresh distinct constant to each variable in $\overline{x}$. Thus, all unifications implied by Σ have been already implemented in I_φ by construction. □

Lemma 12 (Soundness). *For an arbitrary CQ $\varphi(\overline{x})$, and a set of embedded dependencies Σ, $\Sigma \models \Delta_s^\Sigma(\varphi(\bar{x})) \cup \Delta_{st}^\Sigma(\varphi(\bar{x}))$.*

Proof. We prove $\Sigma \models \delta$ for $\delta \in \Delta_s^\Sigma(\varphi(\bar{x}))$ and for $\delta \in \Delta_{st}^\Sigma(\varphi(\bar{x}))$ separately:

$\delta \in \Delta_s^\Sigma(\varphi(\bar{x}))$. Let $\langle K_S, K_T\rangle$ be a pair of source and target instance with $\langle K_S, K_T\rangle \models \Sigma$. We have to show that then $\langle K_S, K_T\rangle \models \delta$ holds. It suffices to show that, if $\langle K_S, K_T\rangle \not\models \delta$, then the chase of $\langle K_S, \emptyset\rangle$ with Σ fails, i.e., rather than showing that K_T is not a solution to K_S under the mapping Σ, we show that K_S has no solution at all under Σ.

By construction, δ has the form $\varphi(\bar{x}) \rightarrow x_i = x_j$, where x_i, x_j are either variables from $\bar{x}$ or constants. By $\langle K_S, K_T\rangle \not\models \delta$, there exists a substitution λ with $At(\varphi(\bar{x}\lambda)) \subseteq K_S$, i.e., λ defines a homomorphism from $At(\varphi(\bar{x}))$ to K_S. Since CQs are closed under homomorphisms, we can replay on K_S the chase of $At(\varphi(\bar{x}))$ with Σ (as carried out by the PROPAGATE procedure). Since $\delta \in \Delta_s^\Sigma(\varphi(\bar{x}))$, this chase sequence contains the application of some egd ε which enforces the equality $x_i = x_j$. Consider the chase on K_S up to the corresponding application of the egd ε. Suppose that the chase on K_S is successful up to this egd application. Now the application of ε enforces the equality $x_i\lambda = x_j\lambda$ in K_S. However, since $\langle K_S, K_T\rangle \not\models \delta$, $x_i\lambda \neq x_j\lambda$ holds in K_S, i.e., the chase of $\langle K_S, \emptyset\rangle$ with Σ fails and, therefore, K_S has no solution at all under Σ.

$\delta \in \Delta_{st}^{\Sigma}(\varphi(\bar{x}))$. Let $J = J_S \cup J_T$ denote the result of chasing $At(\varphi(\bar{x}))$ with Σ (according to step 1 of PROPAGATE). By the computation of $\Delta_{st}^{\Sigma}(\varphi(\bar{x}))$ in step 2, δ is of the form $\varphi(\bar{x}\lambda) \rightarrow \exists\bar{y}\psi(\bar{x}\lambda, \bar{y})$ for some substitution λ.

We prove $\Sigma \models \delta$ by the implication criterion of Beeri and Vardi recalled in Lemma 10: We can chase the antecedent $\varphi(\bar{x}\lambda)$ of δ by applying the same chase order as in the chase of $At(\varphi(\bar{x}))$ in step 1 of PROPAGATE. Clearly, this chase of $At(\varphi(\bar{x}\lambda))$ yields exactly the same result J_T as the chase of $At(\varphi(\bar{x}))$ with Σ, i.e., after replaying the chase of step 1 in PROPAGATE, no additional dependencies fire. Note that $At(\psi(\bar{x}\lambda, \bar{y})) \subseteq J_T$, since $At(\psi(\bar{x}\lambda, \bar{y}))$ is obtained from J_T by core computation (in step 2 of PROPAGATE). Hence, there is a trivial homomorphism from the conclusion $\psi(\bar{x}\lambda, \bar{y})$ of δ to the result J_T of the chase, namely the identity. Thus, by Lemma 10, $\Sigma \models \delta$ holds. □

Lemma 13 (Completeness). *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ and $\sigma \in \Sigma_{st}$, s.t. $\varphi(\bar{x})$ is the antecedent of σ and let $\Sigma^* = (\Sigma \setminus \{\sigma\}) \cup \Delta_s^{\Sigma}(\varphi(\bar{x})) \cup \Delta_{st}^{\Sigma}(\varphi(\bar{x}))$. Then $\Sigma^* \models \Sigma$ holds.*

Proof. Let $\sigma = \varphi(\bar{x}) \rightarrow \exists\bar{y}\psi(\bar{x}, \bar{y}\,)$ and let $\langle K_S, K_T\rangle$ be a pair of source and target instance with $\langle K_S, K_T\rangle \models \Sigma^*$. Moreover, suppose that there exists a substitution ν on the variables $\overline{x}$ with $At(\varphi(\bar{x}\nu)) \subseteq K_S$. We have to show that there exists a substitution ν_y on the variables $\overline{y}$ with $At(\psi(\bar{x}\nu, \bar{y}\nu_y)) \subseteq K_T$.

We first note that the chase of $At(\varphi(\bar{x}))$ with Σ succeeds. That is, it forces no equality between distinct constant terms occurring either in $\varphi(\bar{x})$ or in Σ. For, according to the initial step of the PROPAGATE procedure, in this case $\Delta_s^{\Sigma}(\varphi(\bar{x}))$ would contain a dependency with $\varphi(\bar{x})$ as antecedent and an unsatisfiable equality in the conclusion. Since, by our assumption, there is a homomorphism from $\varphi(\overline{x})$ to K_S and CQs are closed under homomorphisms, all steps of the chase of $At(\varphi(\overline{x}))$ performed by PROPAGATE can be replayed on K_S. Therefore, the assumption $\langle K_S, K_T\rangle \models \Sigma^*$ implies that the chase of $At(\varphi(\bar{x}))$ with Σ succeeds, and thus $\Delta_{st}^{\Sigma}(\varphi(\bar{x}))$ contains a tgd $\tau = \varphi(\bar{x}\lambda) \rightarrow \exists\bar{z}\chi(\bar{x}\lambda, \bar{z}\,)$ for some substitution λ.

Since K_S satisfies all source egds produced by PROPAGATE, there exists a substitution μ on the variables in $\varphi(\bar{x}\lambda)$, s.t. $\nu = \lambda\mu$ and, therefore, $\varphi(\bar{x}\nu) = \varphi(\bar{x}\lambda\mu)$. We can replay on $At(\varphi(\bar{x}\lambda))$ the chase of $At(\varphi(\bar{x}))$ with Σ (as carried out by the PROPAGATE procedure). Moreover, it can be easily verified that the chase of $At(\varphi(\bar{x}\lambda))$ with Σ yields exactly the same result as the chase of $At(\varphi(\bar{x}))$ with Σ, i.e.: we get $J = J_S \cup J_T$ with $J_S = At(\varphi(\bar{x}\lambda))$ as in the PROPAGATE procedure. In the chase of $At(\varphi(\bar{x}\lambda))$, eventually also the s-t tgd σ fires and produces the atoms $At(\psi(\bar{x}\lambda, \bar{y}\,'))$ for fresh labelled nulls $\overline{y}'$. In the further course of the chase of $At(\varphi(\bar{x}\lambda))$, some equalities may be enforced on the labelled nulls $\overline{y}'$. Hence, there exists a substitution η on $\overline{y}$, s.t. $At(\psi(\bar{x}\lambda, \bar{y}\eta)) \subseteq J_T$. By the construction of τ in step 2 of PROPAGATE, $At(\chi(\bar{x}\lambda, \bar{z}))$ is the core of J_T. Hence, there exists a substitution η^* on $\overline{y}$, s.t. $At(\psi(\bar{x}\lambda, \bar{y}\eta^*)) \subseteq At(\chi(\bar{x}\lambda, \bar{z}))$.

Clearly, $\langle K_S, K_T\rangle \models \tau$ since $\tau \in \Sigma^*$ and $\langle K_S, K_T\rangle \models \Sigma^*$. Moreover, $At(\varphi(\bar{x}\nu)) \subseteq K_S$ and $\nu = \lambda\mu$. Therefore, we have $At(\varphi((\bar{x}\lambda)\mu)) \subseteq K_S$. Hence, there exists a substitution μ_z on $\overline{z}$ with $At(\chi(\bar{x}\lambda\mu, \bar{z}\mu_z)) \subseteq K_T$. But then $\nu_y = \eta^*\mu_z$ is the desired substitution on $\overline{y}$ with $At(\psi(\bar{x}\nu, \bar{y}\nu_y)) \subseteq K_T$. □

4.2 Splitting in the presence of egds

The following example illustrates that the splitting rule (i.e., Rule 3 in Figure 3.1) does not suffice to detect the possibility of splitting a "bigger" tgd into smaller ones in the presence of target egds:

Example 16. *Consider the following mapping* $\Sigma = \{\tau, \epsilon\}$

$$\tau\colon\ S(x, z_1) \wedge S(x, z_2) \rightarrow (\exists y)\ R(z_1, y) \wedge Q(z_2, y)$$

$$\epsilon\colon\ R(x_1, y_1) \wedge Q(x_2, y_2) \rightarrow y_1 = y_2$$

It is easy to check that Σ *is equivalent to the mapping* $\Sigma' = \{\tau_1, \tau_2, \epsilon\}$ *with the same target egd and two s-t tgds, each containing only a subset of the antecedent and conclusion atoms of* τ*:*

$$\tau_1\colon\ S(x, z_1) \rightarrow (\exists y)\ R(z_1, y)$$

$$\tau_2\colon\ S(x, z_2) \rightarrow (\exists y)\ Q(z_2, y)$$

The Rule 3 from Chapter 3 does not allow such a splitting, however.

In some sense, the splitting in the above example still had significant similarities with splitting in the absence of egds, namely: the basic idea of distributing the conclusion atoms over several dependencies is still present when target egds have to be taken into account. However, we have to deal with a significant extension here: Without egds it would never be possible to split the connected component (w.r.t. the existential variables) of the conclusion of a tgd. As we have seen in the above example, egds may allow us to tear a connected component apart. Moreover, splitting in the presence of egds is not merely distributing atoms of the conclusion of some dependency over several ones. The following example illustrates that we may also have to copy atoms in order to further split the conclusion of a tgd.

Example 17. *Consider the following mapping* Σ

$$\tau : S(x_1, x_2) \wedge S(x_1, x_3) \rightarrow (\exists y)\ R(x_2, y) \wedge P(y, x_2) \wedge Q(y, x_3)$$

$$\epsilon : R(x, y_1) \wedge R(x, y_2) \rightarrow y_1 = y_2$$

The s-t tgd τ *can be rewritten in the following way:*

$$\tau_1 : S(x_1, x_2) \wedge S(x_1, x_3) \rightarrow (\exists y)\ R(x_2, y) \wedge Q(y, x_3)$$

$$\tau_2 : S(x_1, x_2) \rightarrow (\exists y)\ R(x_2, y) \wedge P(y, x_2)$$

Both τ_1 *and* τ_2 *must contain an* R*-atom.*

We observe that the total number of atoms in all conclusions in the resulting mapping in Example 17 has increased. But compared with the original mapping τ, each conclusion is strictly smaller than the original one. (i.e., is obtained by deletion of at least one atom and possibly the renaming of some variable occurrences). We thus generalize the notion of *split-reduced* mappings from the s-t tgd-only case:

Definition 10. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ be a schema mapping. The tgd $\tau\colon \varphi(\overline{x}) \rightarrow (\exists \overline{y})\, \psi(\overline{x}, \overline{y}) \in \Sigma_{st}$ is* egd-split-reduced*, if it is not possible to replace it by a set of new s-t tgds τ_i with antecedent φ_i and conclusion ψ_i, s.t. $At(\varphi_i) \subseteq At(\varphi)$ and $|At(\psi_i)| < |At(\psi)|$. Σ_{st} is said to be egd-split-reduced if each dependency in it is.*

The above notion of egd-split-reduced mappings generalizes the notion of split-reduced mappings from Definition 2 to mappings with target egds. The connection between the two notions of splitting is formalized by the following lemma:

Lemma 14. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ be a schema mapping with $\Sigma_t = \emptyset$ and suppose that Σ_{st} cannot be simplified by any of the Rules 1,4, and 5 from Figure 3.1 (i.e., the rules which would reduce $ConSize(\Sigma_{st})$ are not applicable). Then the following equivalences hold: Σ_{st} is egd-split-reduced iff Σ_{st} is split-reduced iff Rule 3 (i.e., splitting) cannot be applied.*

Proof. The second equivalence was already shown in Lemma 8. Below we show that Σ_{st} is egd-split-reduced iff Rule 3 (i.e., splitting) cannot be applied.

First suppose that Rule 3 (i.e., splitting) actually can be applied to Σ_{st}. Then some $\tau \in \Sigma_{st}$ can be replaced by tgds $\tau_1, \ldots, \tau_n$, s.t. the antecedent of each τ_i coincides with the antecedent of τ and the conclusion of each τ_i is a proper subset of the conclusion of τ. Hence, Σ_{st} is not egd-split-reduced.

Now suppose that Σ_{st} is not egd-split-reduced. We have to show that then Rule 3 can be applied. Suppose to the contrary that Rule 3 cannot be applied. We derive a contradiction by showing that then one of the Rules 1, 4, 5 is applicable to Σ: Since Σ_{st} is not egd-split-reduced, there exists a $\tau \in \Sigma_{st}$ with antecedent φ which can be replaced by a set of new tgds $\{\tau_1, \ldots, \tau_n\}$, s.t. for every i, $At(\varphi_i) \subseteq At(\varphi)$ and $|At(\psi_i)| < |At(\psi)|$ hold, where φ_i and ψ_i respectively denote the antecedent and conclusion of τ. Moreover, $\Sigma \equiv \Sigma'$ holds with $\Sigma' = (\Sigma \setminus \{\tau\}) \cup \{\tau_1, \ldots, \tau_n\}$. In particular, $\Sigma' \models \tau$. By Lemma 4, then either (1) $\Sigma' \models \tau'$ holds for some proper instance τ' of τ (see Definition 6) or (2) τ is already implied by a single tgd $\sigma \in \Sigma'$.

In case (1), we clearly also have $\Sigma \models \tau'$ for the proper instance τ' of τ. But then, by Lemma 5, Rule 5 is applicable to τ, which is a contradiction. It remains to consider case (2). Clearly, $\sigma \notin \Sigma \setminus \{\tau\}$ since otherwise τ could be deleted from Σ via Rule 4. So let $\sigma = \tau_j$ for some j. We thus have $\bar{\Sigma} \models \tau$ with $\bar{\Sigma} = (\Sigma \setminus \{\tau\}) \cup \tau_j$ and, therefore, also $\bar{\Sigma} \equiv \Sigma$. Moreover, $|At(\psi_j)| < |At(\psi)|$ holds, which implies $ConSize(\bar{\Sigma}) < ConSize(\Sigma)$. Now suppose that we transform both Σ and $\bar{\Sigma}$ into the unique (up to isomorphism) normal form Σ^* according to Definition 7. By assumption, none of the Rules 1, 3, 4, and 5 is applicable to Σ. Hence, by the same considerations as in the proof of Lemma 8, Σ^* is obtained by successive applications of Rule 2, which leaves the conclusions of the tgds unchanged. Hence, we have $ConSize(\Sigma^*) = ConSize(\Sigma)$. On the other hand, if we transform $\bar{\Sigma}$ into the normal form Σ^*, then we never increase the conclusion size. Hence, from the inequality $ConSize(\bar{\Sigma}) <$

$ConSize(\Sigma)$ we may infer $ConSize(\Sigma^*) < ConSize(\Sigma)$, which contradicts the equality that we have just derived. □

In Figure 4.2 we present the Rule ES, whose exhaustive application obviously transforms any mapping into an egd-split-reduced one. Alas, the following example shows that we cannot hope to get a unique egd-split-reduced mapping.

Example 18. *Consider the schema mapping Σ consisting of a single s-t tgd and a number of target egds:*

$$S(1,x) \wedge S(1,2) \wedge S(y,2) \rightarrow (\exists x,v,w)\, T(x,y,z) \wedge P(x,z) \wedge R(y,z) \wedge Q(z,v,w)$$

$$T(x,y,z) \wedge P(x,z) \wedge Q(z,v,w) \rightarrow v = w$$
$$T(x,y,z) \wedge R(y,z) \wedge Q(z,v,w) \rightarrow v = w$$

$$T(x,y,z_1) \wedge P(x,z_2) \wedge Q(w,v,v) \rightarrow z_1 = z_2$$
$$T(x,y,z_1) \wedge R(y,z_2) \wedge Q(w,v,v) \rightarrow z_1 = z_2$$

This mapping is not in the egd-split-reduced form, since the antecedent of the s-t tgd can be shrunk by extracting either the P or the Q atom from the conclusion. Σ'_{st} and Σ''_{st} are two possible transformations of Σ into egd-split-reduced form via the Rule ES.

$$\Sigma'_{st} = \{\ S(1,x) \wedge S(1,2) \wedge S(y,2) \rightarrow (\exists z,v,w) T(x,y,z) \wedge R(y,z) \wedge Q(z,v,w),$$
$$S(1,x) \wedge S(1,2) \rightarrow (\exists z)\ P(x,z)\} \text{ and}$$

$$\Sigma''_{st} = \{\ S(1,x) \wedge S(1,2) \wedge S(y,2) \rightarrow (\exists z,v,w)\ T(x,y,z) \wedge P(x,z) \wedge Q(z,v,w),$$
$$S(1,2) \wedge S(y,2) \rightarrow (\exists z)\ R(y,z)\}.$$

Clearly, the problem in Example 18 is not just due to the definition of the Rule ES. Instead, it is an intrinsic problem of the notion of egd-split-reduced mappings. Apparently this extent of splitting is too strong. We shall therefore relax the notion of egd-split-reduced. This will be the topic of the next section.

4.3 Antecedent-split-reduced mappings

In Example 18 we observed that certain antecedent atoms may be freely distributed between several tgds, if the idea of splitting from Chapter 3 is directly adopted in the setting with target constraints. Therefore, in order to arrive at an intuitive definition of a unique normal form, we shift our focus to the antecedents:

Definition 11. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ be a schema mapping. The s-t tgd $\tau\colon \varphi(\overline{x}) \rightarrow (\overline{y})\ \psi(\overline{x},\overline{y})$ in Σ_{st} is* antecedent-split-reduced*, if it is not possible to replace it with a set of new s-t tgds τ_i each having strictly smaller antecedent, i.e., for the antecedents φ_i, we get $|At(\varphi_i)| < |At(\varphi)|$. Σ_{st} is said to be antecedent-split-reduced if each dependency in it is.*

Rewrite Rules in the Presence of Egds

Rule E1 (General implication).
$\Sigma \Longrightarrow \Sigma \setminus \{\tau\}$
if $\Sigma \setminus \{\tau\} \models \tau$.

Rule E2 (Restriction of an antecedent to endomorphic images).
$\Sigma \Longrightarrow (\Sigma \setminus \{\tau\}) \cup \{\tau_1, \ldots, \tau_n\}$
if $\tau\colon \varphi(\overline{x}\,) \rightarrow (\exists \overline{y}\,)\psi(\overline{x}, \overline{y}\,)$
and $(\Sigma \setminus \{\tau\}) \cup \{\tau_1, \ldots, \tau_n\} \models \tau$
and for each $i \in \{1, \ldots, n\}$
$\tau_i\colon \varphi_i(\overline{x}_i) \rightarrow (\exists \overline{y}_i)\psi_i(\overline{x}_i, \overline{y}_i)$,
s.t. $\exists \lambda$, $At(\varphi(\overline{x}\lambda)) = At(\varphi_i(\overline{x}_i)) \subset At(\varphi(\overline{x}))$
and $\psi_i(\overline{x}_i, \overline{y}_i) = \mathit{core}(At(\varphi_i(\overline{x}_i)), \Sigma)$.

Rule E3 (Implication of atoms in the conclusion).
$\Sigma \Longrightarrow (\Sigma \setminus \{\tau\}) \cup \{\tau'\}$
if $\tau\colon \varphi(\overline{x}\,) \rightarrow (\exists \overline{y}\,)\psi(\overline{x}, \overline{y}\,)$
and $\tau'\colon \varphi(\overline{x}\,) \rightarrow (\exists \overline{y}\,')\psi'(\overline{x}, \overline{y}\,')$,
s.t. $At(\psi'(\overline{x}, \overline{y}\,')) \subset At(\psi(\overline{x}, \overline{y}\,))$
and $(\Sigma \setminus \{\tau\}) \cup \{\tau'\} \models \tau$.

Rule ES (generalized splitting in the presence of egds).
$\Sigma \Longrightarrow (\Sigma \setminus \{\tau\}) \cup \{\tau_1, \ldots, \tau_n\}$
if $\tau\colon \varphi(\overline{x}\,) \rightarrow (\exists \overline{y}\,)\psi(\overline{x}, \overline{y}\,)$
and $(\Sigma \setminus \{\tau\}) \cup \{\tau_1, \ldots, \tau_n\} \models \tau$
and for each $i \in \{1, \ldots, n\}$
$\tau_i\colon \varphi_i(\overline{x}_i) \rightarrow (\exists \overline{y}_i)\psi_i(\overline{x}_i, \overline{y}_i)$,
s.t. $\emptyset \subset At(\varphi_i(\overline{x}_i)) \subseteq At(\varphi(\overline{x}))$
and $At(\psi_i(\overline{x}_i, \overline{y}_i\mu_i)) \subset At(\psi(\overline{x}, \overline{y}))$ for a substitution μ_i.

Rule E2-eager (Restriction of an antecedent to subsets).
$\Sigma \Longrightarrow (\Sigma \setminus \{\tau\}) \cup \{\tau_1, \ldots, \tau_n\}$
if $\tau\colon \varphi(\overline{x}\,) \rightarrow (\exists \overline{y}\,)\psi(\overline{x}, \overline{y}\,)$
and $(\Sigma \setminus \{\tau\}) \cup \{\tau_1, \ldots, \tau_n\} \models \tau$
and for each $i \in \{1, \ldots, n\}$
$\tau_i\colon \varphi_i(\overline{x}_i) \rightarrow (\exists \overline{y}_i)\psi_i(\overline{x}_i, \overline{y}_i)$,
s.t. $\emptyset \subset At(\varphi_i(\overline{x}_i)) \subset At(\varphi(\overline{x}))$
and $\psi_i(\overline{x}_i, \overline{y}_i) = \mathit{core}(At(\varphi_i(\overline{x}_i)), \Sigma)$.

Figure 4.2: Rewrite rules in the presence of egds.

In order to transform a mapping into an antecedent-split-reduced one, we define the rule E2-eager in Figure 4.2. It can be shown that any normal form under a rule rewrite system containing Rule E2-eager is antecedent-split-reduced and vice versa. In this rule, we have to inspect all subsets of the antecedent database of each tgd. Actually, we will show that it suffices to check all subsets φ_i of an antecedent $\varphi(\overline{x})$, such that φ_i is a proper endomorphic image of $\varphi(\overline{x})$. This is what the Rule E2 in Figure 4.2 does. Clearly, the number of endomorphic images is, in general, far smaller than the number of all subsets. In particular, we never have to check antecedents smaller than the core of the already present antecedents (here we mean the core of

the conjunctive query – without distinguishing two groups of variables as we did in the definition of the Rules 1 and 2). The following Theorem shows that both, the rule E2-eager and the rule E2 exactly capture the notion of antecedent-split-reduced mappings.

Theorem 5. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ be a schema mapping in which no tgd can be deleted via the Rule E1. Then the following properties are equivalent:*

1. *Σ is antecedent-split-reduced.*
2. *Σ is reduced w.r.t. Rule E2-eager.*
3. *Σ is reduced w.r.t. Rule E2.*

Proof. (1) $\Leftrightarrow$ (2) follows directly from the definition of antecedent-split-reduced form.

(2) $\Rightarrow$ (3) is trivial, as every proper endomorphic image of a set of atoms $At(\varphi)$ is a proper subset of $At(\varphi)$.

(3) $\Rightarrow$ (2). Consider an application of Rule E2-eager, in which it substitutes some s-t tgd τ in Σ with a set of s-t tgds T, s.t. $\Sigma \equiv \Sigma \setminus \{\tau\} \cup T$. Let now φ be the antecedent of τ, and φ_i be the antecedent of some tgd $\tau_i \in T$. By definition of E2-eager, $At(\varphi_i) \subset At(\varphi)$. We have to show that $At(\varphi_i)$ is an endomorphic image of $At(\varphi)$. Suppose to the contrary that it is not. We show that then τ_i is "superfluous" in T: Namely, the property $(\Sigma \setminus \{\tau\}) \cup T \setminus \{\tau_i\} \equiv (\Sigma \setminus \{\tau\}) \cup T$ holds. Of course, if T only contains such tgds which are superfluous, then the tgd τ itself can be deleted by Rule E1, which is a contradiction to the assumption of this theorem. On the other hand, if T is non-empty and contains only tgds whose antecedent is an endomorphic image of φ then also Rule E2 is applicable.

To show that τ_i is superfluous in T, consider the following two cases:

(a) There exists no homomorphism $At(\varphi) \to At(\varphi_i)$. Then, τ_i is superfluous in T, in a sense that the property $(\Sigma \setminus \{\tau\}) \cup T \setminus \{\tau_i\} \models \tau$ holds. Indeed, according to E2-eager, the conclusion of τ_i was created by chasing $At(\varphi_i)$ with Σ. Since $At(\varphi) \not\to At(\varphi_i)$, τ played no role in that chase, and thus $\Sigma \setminus \{\tau\} \models \tau_i$ holds, and hence τ_i is indeed superfluous.

(b) There exists a homomorphism $At(\varphi) \to At(\varphi_i)$. Let Λ denote the set of all homomorphisms $At(\varphi) \to At(\varphi_i)$. Then we define $T_\varphi \subset T$, s.t. $T_\varphi = \{\tau_j \in T \mid At(\varphi_j) = At(\varphi(\overline{x}\lambda)) \text{ for some } \lambda \in \Lambda\}$, i.e., the antecedents of the tgds in T_φ are subsets of φ_i and, at the same time, endomorphic images of φ. Similarly to the previous case, one can show that $(\Sigma \setminus \{\tau\}) \cup T_\varphi \models \tau_i$ holds. Thus τ_i is superfluous.

By construction of τ_i, we have $\Sigma \models \tau_i$. Indeed, consider the implication test of Lemma 10, in which the database $At(\varphi_i)$, obtained from the antecedent of τ_i, is chased by Σ. The effect of τ in this chase is exactly the generation of conclusion atoms by instantiating the existentially quantified variables $\overline{y}$ in $\psi(\overline{x}\lambda, \overline{y})$, where $\psi(\overline{x}, \overline{y})$ is the conclusion of τ and λ is an endomorphism transforming the antecedent $\varphi(\overline{x})$ of τ into its subformula $\varphi_i(\overline{x}) = \varphi(\overline{x}\lambda)$, which is the

antecedent of τ_i. But then, by definition of Rule E2, there exists a substitution μ such that $\psi(\overline{x}\lambda_i, \overline{y}\mu)$ is a subformula of the conclusion of some $\tau_\varphi \in T_\varphi$: Indeed, Rule E2 takes exactly all endomorphic images of φ and performs the chase with Σ to derive the conclusion of an s-t tgd in T_φ. Note that the substitution μ captures the effect of egds possibly fired by the chase which derives the conclusion of τ_φ from its antecedent $\varphi(\overline{x}\lambda)$; these egds will surely be also fired in the chase of $At(\varphi_i)$.

Let the chase of $At(\varphi_i)$ with τ yield the target instance J, and the chase of $At(\varphi_i)$ with T_φ yield J'. Then, it is easy to see that a homomorphism $J \rightarrow J'$ must exist. But then, $chase(At(\varphi_i), \Sigma) \models \tau_i$ only if $chase(At(\varphi_i), \Sigma') \models \tau_i$ with $\Sigma' = (\Sigma \setminus \{\tau\}) \cup T_\varphi$ Hence, it must be the case that $(\Sigma \setminus \{\tau\}) \cup T_\varphi$ holds and, therefore, τ_i is superfluous. □

There is a close connection between antecedent-split-reduced mappings and the split-reduced form from Definition 2:

Lemma 15. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ be a schema mapping with $\Sigma_t = \emptyset$ and suppose that Σ_{st} cannot be simplified by any of the Rules 1,4, and 5 from Figure 3.1 (i.e., the rules which would reduce $ConSize(\Sigma_{st})$ are not applicable). Then the following equivalence holds: Σ_{st} is antecedent-split-reduced iff Rule 3 (i.e., splitting) cannot be applied in such a way that the antecedents of all resulting dependencies can be further simplified (by Rule 2).*

Proof. First suppose that Rule 3 (i.e., splitting) followed by a simplification of the antecedent of each new tgd is applicable. Then, in the first place, some $\tau \in \Sigma_{st}$ can be replaced by tgds $\tau_1, \ldots, \tau_n$, s.t. the antecedent of each τ_i coincides with the antecedent of τ and the conclusion of each τ_i is a proper subset of the conclusion of τ. Moreover, each τ_i can then be transformed via Rule 2 into τ_i', s.t. the antecedent of τ_i' is a proper subset of the antecedent of τ and, therefore, also of τ. Hence, Σ_{st} is not antecedent-split-reduced.

For the opposite direction, suppose that Σ_{st} is not antecedent-split-reduced. We have to show that then Rule 3 can be applied followed by applying Rule 2 to each of the new tgds. Since Σ_{st} is not antecedent-split-reduced, there exists a $\tau \in \Sigma_{st}$ with antecedent φ which can be replaced by a set of new tgds $\{\tau_1, \ldots, \tau_n\}$, s.t. for every i, $At(\varphi_i) \subset At(\varphi)$ and $|At(\psi_i)| < |At(\psi)|$ hold, where φ_i and ψ_i respectively denote the antecedent and conclusion of τ. Moreover, $\Sigma \equiv \Sigma'$ holds with $\Sigma' = (\Sigma \setminus \{\tau\}) \cup \{\tau_1, \ldots, \tau_n\}$.

Analogously to the proof of Theorem 5, we may assume w.l.o.g., that each of the new antecedents φ_i is an endomorphic image of φ. Moreover, we may assume w.l.o.g., that each ψ_i contains only one connected component since otherwise we simply split τ_i further via Rule 3. We claim that for every connected component of ψ, there is one i, s.t. this connected component corresponds to ψ_i. Suppose to the contrary that there is a connected component χ of ψ which does not have a corresponding ψ_i. Then we derive a contradiction as follows. The tgd τ' obtained from τ by reducing the conclusion ψ to χ is clearly implied by Σ'. Hence, by Lemma 4, either (1) $\Sigma' \models \tau''$ holds for some proper instance τ'' of τ' (see Definition 6) or (2) τ' is already implied by a single tgd $\sigma \in \Sigma'$. In case (1), we thus have $\Sigma \models \tau''$ for the proper instance

τ'' of τ'. But then, also $\Sigma \models \tau^*$, where τ^* denotes the tgd obtained from τ by replacing the connected component χ by the conclusion of τ'' (i.e., a proper instance of χ) and leaving all other connected components unchanged. By Lemma 5, Rule 5 is applicable to $\tau \in \Sigma$, which is a contradiction. Now consider case (2), i.e., τ' is implied by a single tgd $\sigma \in \Sigma'$. Clearly, σ cannot be contained in $\Sigma \setminus \{\tau\}$ since this would mean that the connected component χ of the conclusion of τ could be deleted from Σ via Rule 5. So suppose that $\sigma = \tau_j$ for some j, i.e., we have $\tau_j \models \tau'$. By Lemma 10, this means that the conclusion χ of τ' can be obtained by chasing the antecedent φ of τ' with χ. Note however that χ is a single connected component. Hence, all of χ is obtained in a single chase step, since otherwise we conclude that also a proper instance of τ' is implied by τ_j and we proceed as in case (1). Since χ is obtained in a single chase step, the conclusion of τ_j indeed comprises all of χ.

To conclude the proof, recall the above observation that each of the antecedents φ_i is an endomorphic image of φ. But then we can indeed apply Rule 2 in the reverse direction to extend each φ_i to φ. Let the resulting tgds be called $\{\bar{\tau}_1, \ldots, \bar{\tau}_n\}$. Then we indeed have that $\tau \in \Sigma$ may be replaced by $\{\bar{\tau}_1, \ldots, \bar{\tau}_n\}$ via Rule 3 and each $\bar{\tau}_i$ may be further simplified via Rule 2 to τ_i with strictly smaller antecedent. □

Most importantly, the notion of antecedent-split-reduced mappings allows us to define a unique (up to isomorphism) normal form of the set of s-t tgds. To this end, we consider the transformation of an arbitrary mapping consisting of s-t tgds and target egds by the PROPAGATE procedure from Figure 4.1 followed by exhaustive application of the rules E1 and E2 from Figure 4.2. Below we show that the resulting normal form is indeed unique up to isomorphism:

Lemma 16. *The rewrite rules E1 and E2 in Figure 4.2 are correct, i.e.: Let Σ be a set of s-t tgds and target egds and let Σ' be the result of applying one of the rules E1 or E2 to Σ. Then $\Sigma \equiv \Sigma'$.*

Proof. The correctness follows directly from the fact that a logical implication test is built into the rules E1 and E2. □

Theorem 6. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ and $\Upsilon = \Upsilon_{st} \cup \Upsilon_t$ be two logically equivalent schema mappings consisting of s-t tgds and target egds and let $\langle \Sigma_s, \Sigma^*_{st}, \Sigma_t \rangle$ and $\langle \Upsilon_s, \Upsilon^*_{st}, \Upsilon_t \rangle$ be obtained by first rewriting Σ resp. Υ with the* PROPAGATE *procedure and then exhaustively applying Rules E1 and E2 to the rewritten sets of s-t tgds in these mappings. Then $\Sigma_t \equiv \Upsilon_t$ holds and Σ^*_{st} and Υ^*_{st} are isomorphic.*

Proof. The equivalence $\Sigma_t \equiv \Upsilon_t$ was shown in Lemma 9. It remains to show that Σ^*_{st} and Υ^*_{st} are isomorphic.

Let $\sigma \in \Sigma^*_{st}$ be an arbitrary s-t tgd in Σ^*_{st}. We have to show that it has an isomorphic analogue in Υ^*_{st} (and vice versa). Let $\bar{\Sigma}_\sigma$ denote the set of s-t tgds whose antecedents are the proper subsets of σ and whose conclusions are obtained by chasing the corresponding antecedent database with $\Sigma^*_{st} \cup \Sigma_t$ (i.e., we get s-t tgds analogous to the τ_i's in Rule E2). By Lemma 11, the chase of the frozen antecedent database of σ with Σ succeeds. The same must then hold for any

set of tuples from this database: the antecedent of σ and all its subformulas satisfy Σ_s. Then, by Lemmas 12 and 13, the conclusion obtained by chasing $\Sigma_{st}^* \cup \Sigma_t$ must have the same core as if it were obtained by chasing with Σ.

By $\Upsilon \models \sigma$, there exists a subset $T \subseteq \Upsilon_s \cup \Upsilon_{st}^* \cup \Upsilon_t$, s.t. $T \cup \Sigma_s \cup \bar{\Sigma}_\sigma \cup \Sigma_t \cup \Sigma_{st}^* \setminus \{\sigma\} \models \sigma$. We claim that there even exists a set $T_\sigma \subseteq \Upsilon_{st}^*$ with $T_\sigma \cup \Sigma_s \cup \bar{\Sigma}_\sigma \cup \Sigma_t \cup \Sigma_{st}^* \setminus \{\sigma\} \models \sigma$, s.t. every $\tau \in T_\sigma$ fulfils the following properties:

1. The antecedents $\varphi_\tau(\overline{x}_\tau)$ of τ and $\varphi_\sigma(\overline{x}_\sigma)$ of σ are homomorphically equivalent;
2. there exists a substitution λ, s.t. $\varphi_\tau(\overline{x}_\tau\lambda) = \varphi_\sigma(\overline{x}_\sigma)$. That is, the antecedent of τ can be mapped onto the entire antecedent of σ;
3. τ is not equivalent to any dependency in $\Sigma_{st} \setminus \{\sigma\}$

In order to prove this claim, we start with a set $T \subseteq \Upsilon_s \cup \Upsilon_{st}^* \cup \Upsilon_t$, s.t. $T \cup \Sigma_s \cup \bar{\Sigma}_\sigma \cup \Sigma_t \cup \Sigma_{st}^* \setminus \{\sigma\} \models \sigma$ and remove all parts from T until a subset $T_\sigma \subseteq T$ with the desired properties is obtained. It is convenient to write Σ^* as a short-hand for $\Sigma_s \cup \bar{\Sigma}_\sigma \cup \Sigma_t \cup \Sigma_{st}^* \setminus \{\sigma\}$.

(a) *Eliminate Υ_s from T.* This is justified by the fact that $\Upsilon_{st}^* \cup \Upsilon_t \models \sigma$ holds. Suppose to the contrary that σ is not implied by $\Upsilon_{st}^* \cup \Upsilon_t$. Let I be an instance over the schema $\langle \mathbf{S}, \mathbf{T} \rangle$, in which the only non-empty relations are those of the antecedent database $At(\varphi_\sigma(\overline{x}_\sigma))$ of σ. Then, $chase(I, \Upsilon_{st} \cup \Upsilon_t) \not\models \sigma$, whereas $chase(I, \Upsilon_s \cup \Upsilon_{st} \cup \Upsilon_t) \models \sigma$. Since source dependencies in Υ_s are only applicable to relations of the source schema, it must hold that Υ_s modifies $At(\varphi_\sigma(\overline{x}_\sigma))$; otherwise there would be no difference between the two chase results. That is, $At(\varphi_\sigma(\overline{x}_\sigma)) \not\models \Upsilon_s$. By Lemma 13, this means that the chase of $At(\varphi_\sigma(\overline{x}_\sigma))$ with Υ fails. Thus, also the chase with Σ fails, which contradicts Lemma 11.

(b) *Eliminate Υ_t from T.* The correctness of this step follows immediately from the equivalence $\Upsilon_t \equiv \Sigma_t$ that we showed in Lemma 9.

(c) *Eliminate every tgd τ from T which is equivalent to some $\sigma' \in \Sigma_{st}^* \setminus \{\sigma\}$.* Clearly, after such a reduction, we still have $T \cup \Sigma^* \models \sigma$ with $\Sigma^* = \Sigma_s \cup \bar{\Sigma}_\sigma \cup \Sigma_t \cup \Sigma_{st}^* \setminus \{\sigma\}$.

(d) *Eliminate from T all dependencies with the antecedent $\varphi_i(\overline{x}_i)$ which is not homomorphically equivalent to the antecedent $\varphi_\sigma(\overline{x}_\sigma)$ of σ.* Indeed, for every s-t tgd $\tau_i \in \Upsilon_{st}^*$ with the antecedent "more specific" than $\varphi_\sigma(\overline{x}_\sigma)$, we may conclude that for arbitrary Σ', such that $\Sigma' \models \sigma$, it holds that $\Sigma' \setminus \{\tau_i\} \models \sigma$. For every τ_j with the antecedent "more general" than $\varphi_\sigma(\overline{x}_\sigma)$, we have that $\Sigma^* \setminus \{\sigma\} \models \tau_j$, and therefore, τ_j is redundant in $T \cup \Sigma^* \setminus \{\sigma\}$.

(e) *Eliminate from T all s-t tgds with the antecedents $\varphi_k(\overline{x}_k)$ such that there exists no variable substitution λ:* $\varphi_k(\overline{x}_k\lambda) = \varphi_\sigma(\overline{x}_\sigma)$, where $\varphi_\sigma(\overline{x}_\sigma)$ again denotes the antecedent of σ. First observe that there are no dependencies in T whose antecedents under any variable substitution are supersets of $\varphi_\sigma(\overline{x}_\sigma)$, since they are "more specific" than $\varphi_\sigma(\overline{x}_\sigma)$ and have therefore been removed in the previous step.

Now consider the substitutions λ_{ki}, s.t. $\varphi_k(\overline{x}_k\lambda_{ki}) \subset \varphi_\sigma(\overline{x}_\sigma)$ and the corresponding s-t tgds $\tau_k \in T$. We claim that the following property holds:

For any set of dependencies K such that $\tau_k \in K$, $K \models \sigma$ iff $(K \setminus \{\tau_k\}) \cup K_{\tau_k} \models \sigma$, where K_{τ_k} is the set of all instantiations of τ_k with λ_{ki}: $\tau_{ki} = \varphi_k(\overline{x}_k\lambda_{ki}) \rightarrow \exists\overline{y}_k\ \psi(\overline{x}_k\lambda_{ki}, \overline{y}_k)$.

The claim follows from the consideration of the implication test by Beeri and Vardi [7]: to chase the antecedent database $At(\varphi_\sigma(\overline{x}_\sigma))$ of σ, τ_k is instantiated by every λ_{ki} and thus has the same effect in the chase as K_{τ_k}. Hence, every τ_k in T whose antecedent cannot be projected onto the entire $\varphi_\sigma(\overline{x}_\sigma)$ may be replaced by the respective instantiations K_{τ_k}.

We now recall that the antecedents of the s-t tgds $\rho_l \in \bar{\Sigma}_\sigma \subset \Sigma^*$ range over all possible subsets of $\varphi_\sigma(\overline{x}_\sigma)$. That is, for each τ_{ki} with the antecedent $\varphi_k(\overline{x}_k\lambda_{ki})$ there exists ρ_{ki} with the identical antecedent and with the conclusion obtained by chasing $\varphi_k(\overline{x}_k\lambda_{ki})$ with Σ. Since Σ and Υ are equivalent, we conclude that $\rho_{ki} \models \tau_{ki}$, and thus $\Sigma^* \models K_{\tau_k}$ for every τ_k. Hence, it is indeed allowed to eliminate from T all s-t tgds with the antecedents $\varphi_k(\overline{x}_k)$ such that there exists no variable substitution λ: $\varphi_k(\overline{x}_k\lambda) = \varphi_\sigma(\overline{x}_\sigma)$.

After the above five elimination steps, T is indeed reduced to a set T_σ of the desired form. Note that T_σ is non-empty. This can be seen as follows: The s-t tgd σ is reduced w.r.t. rules E1 and E2. Hence, $\Sigma_s \cup \bar{\Sigma}_\sigma \cup \Sigma_t \cup \Sigma^*_{st} \setminus \{\sigma\} \not\models \sigma$ and, therefore, T_σ must be non-empty.

By obvious symmetry reasons, the same holds for any s-t tgd $\tau \in \Upsilon^*_{st}$ as well: each τ must also have such a corresponding non-empty set $S_\tau \subseteq \Sigma^*_{st}$, with elements satisfying the conditions 1–3.

We now construct a directed bipartite graph $G = (V_1, V_2, E)$ as follows: We associate the s-t tgds in Σ^*_{st} and Υ^*_{st} with the vertices, s.t. $V_1 = \Sigma^*_{st}$ and $V_2 = \Upsilon^*_{st}$. Moreover, whenever $\tau \in T_\sigma$ (resp. $\sigma \in S_\tau$), then there is an edge from τ to σ (resp. from σ to τ).

The conditions 1–3 of T_σ and S_τ translate into the following properties of the graph G:

a. Every vertex has an incoming edge, since the sets T_σ and S_τ are non-empty.

b. Cycles in G have length at most 2. Indeed, by property 2, an edge from τ to σ implies that the size of the antecedent of τ is no less than the size of σ. But then all s-t tgds associated to the vertices in a cycle must have antecedents of equal size. By properties 1 and 2, all such antecedents are isomorphic. This means that the conclusions are isomorphic as well, since they are obtained as cores of the chase of isomorphic source instances with equivalent sets of dependencies (procedure PROPAGATE).

c. Vertices that participate in such a two-edge cycle are disconnected from the rest of the graph. This follows from the fact that the corresponding s-t tgds are equivalent, and thus any other edge would contradict the property 3.

We obtained a graph, of which each vertex should be connected by an incoming path to a cycle: there is only a finite number of vertices, and from each vertex an infinite incoming path can be traced, by the property (a). Considering (c), this is only possible if each vertex itself belongs to a cycle, and, by (b), G must consist of connected components of size 2. In total, this means, that every s-t tgd $\sigma \in \Sigma^*_{st}$ has an isomorphic counterpart $\tau \in \Upsilon^*_{st}$ and vice versa. □

The question now is how to further simplify the set of s-t tgds. Due to the egds, we could strengthen Rule 5 from Chapter 3 (i.e., deletion of redundant atoms from some conclusion) to the Rule E3 in Figure 4.2. Unfortunately, this would again lead to a non-unique normal form as the following example illustrates.

Example 19. *Consider the mapping consisting of two s-t tgds and one egd:*

$S(x, y) \rightarrow (\exists z)\ P(x, z) \wedge Q(x, z)$
$S(x, y) \rightarrow (\exists z)\ R(x, z) \wedge Q(x, z)$
$P(x, z_1) \wedge R(x, z_2) \rightarrow z_1 = z_2$

It is easy to verify that the atom Q can be eliminated by the rule E3 from the conclusion of any of the two tgds, but not from both.

However, if we content ourselves with the simplifications from the s-t tgds only case (i.e,. Rules 1 – 5 from Chapter 3), then we get an intuitive normal form which is simplified to a large extent and which is guaranteed to be unique up to isomorphism. As was mentioned earlier, it is sometimes important in data exchange to arrive at a unique canonical universal solution (this is in particular the case for defining the semantics of queries in a way that the semantics does not not depend on the syntax of the dependencies). In these situations, the normal form defined below should be chosen.

Definition 12. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ be a schema mapping consisting of s-t tgds and target egds and let the result of rewriting of Σ with* PROPAGATE *be the schema mapping $\Sigma_s \cup \Sigma'_{st} \cup \Sigma_t$. Moreover, let Σ^*_{st} denote the set of s-t tgds resulting from $\Sigma_{st}{}'$ by exhaustive application of Rules E1, E2 as well as Rules 1–5 from Chapter 3 and let Σ^*_s denote the result of exhaustive reduction of Σ_s via rule E1. Then we call $\langle \Sigma^*_s, \Sigma^*_{st}, \Sigma_t \rangle$ the* normal form *of Σ.*

Theorem 7. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ and $\Upsilon = \Upsilon_{st} \cup \Upsilon_t$ be equivalent sets consisting of s-t tgds and target egds and let $\langle \Sigma^*_s, \Sigma^*_{st}, \Sigma_t \rangle$ and $\langle \Upsilon^*_s, \Upsilon^*_{st}, \Upsilon_t \rangle$ be the corresponding normal forms. Then Σ^*_{st} and Υ^*_{st} are isomorphic. Moreover, $\Sigma_t \equiv \Upsilon_t$ holds.*

Proof. The fact that Σ^*_{st} and Υ^*_{st} are isomorphic follows immediately from Theorems 1 and 6. The equivalence $\Sigma_t \equiv \Upsilon_t$ was proved in Lemma 9. □

4.4 Homomorphically equivalent components

The normal form obtained by the PROPAGATE procedure followed by the Rules E1 and E2 is not optimal in all respects yet. In particular, both the PROPAGATE procedure and the Rule E2 may have introduced more atoms than needed in the conclusion of s-t tgds. Moreover, by the Rule E2, we may have split the antecedents of tgds into several smaller ones, such that the total number of atoms in the antecedents is increased. Of course, we may now simply apply the rules from Figure 3.1 to further simplify the set of s-t tgds. However, in the final part of this chapter,

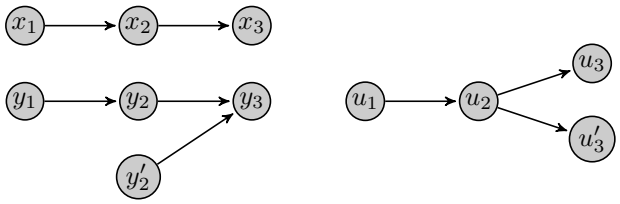

Figure 4.3: Antecedents of τ_1 (left) and τ_2 (right), Example 20

we want to look in a principled way at further optimizations of the normal form of s-t tgds in the presence of egds. The following concept is crucial.

Definition 13. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ be a schema mapping. We say that two tgds τ_1 and τ_2 in Σ_{st} are* homomorphically equivalent *if their antecedents are. Moreover, we say two sets S, S' of tgds are homomorphically equivalent if the tgds in one set and the tgds in the other set are pairwise homomorphically equivalent.*

Obviously, homomorphical equivalence is indeed an equivalence relation on Σ_{st}. We refer to the equivalence classes of this relation as the HE-components *of Σ_{st}.*

We now define a partial order on the HE-components of a set of s-t tgds by considering a "more general" component as greater than a "more specific" one (i.e., there are homomorphisms from the more general one into the "more specific" one but not vice-versa). Moreover, we also consider the closure under the greater-than relation. Below, we show that the closure of each HE-component is unique up to logical equivalence.

Definition 14. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ be a schema mapping and let $\mathcal{S} = \{S_1, \ldots, S_m\}$ denote the HE-components of Σ_{st}. We define a* partial order *as follows: for any pair of indices i, j, we define $S_i \geq S_j$ if for every antecedent $\varphi(\overline{x})$ of the tgds in S_i and every antecedent $\chi(\overline{z})$ of the tgds in S_j, $At(\varphi(\overline{x})) \rightarrow At(\chi(\overline{z}))$ holds (i.e., there is a homomorphism from $\varphi(\overline{x})$ to $\chi(\overline{z})$). If $S_j \not\geq S_i$, S_i is said to be strictly greater than S_j, $S_i > S_j$.*

For $i \in \{1, \ldots, n\}$, we define the closure of S_i above *as $Cl_{\geq}(S_i, \Sigma) = \{\tau \mid \tau \in S_j$ for some j with $S_j \geq S_i\}$.*

Example 20. *Consider a source schema consisting of a single relation symbol $P(\cdot, \cdot)$, a target schema with relational symbols $Q(\cdot, \cdot), T(\cdot, \cdot)$ and a schema mapping $\Sigma = \{\tau_1, \tau_2, \tau_3, \tau_4\}$, where the τ_i's are defined as follows:*

$$\tau_1 : P(x_1, x_2) \wedge P(x_2, x_3) \wedge P(y_1, y_2) \wedge P(y_2, y_3) \wedge P(y_2', y_3) \rightarrow Q(x_1, y_3)$$
$$\tau_2 : P(u_1, u_2) \wedge P(u_2, u_3) \wedge P(u_2, u_3') \rightarrow Q(u_3, u_3')$$
$$\tau_3 : P(v_1, v_2) \rightarrow T(v_1, v_2)$$
$$\tau_4 : P(v_1, v_1) \rightarrow Q(v_1, v_1)$$

Intuitively, the binary relation symbol $P(\cdot,\cdot)$ can be thought of as defining edges of a directed graph. Then the antecedent of the tgd τ_1 consists of two connected components: two paths of length two, one having an additional edge pointing to the peak. The antecedent of the tgd τ_2 corresponds to a Y-shaped graph (see Figure 4.3). The antecedent of τ_3 consists of a single edge, and the antecedent of τ_4 consists of a single self-loop.

The antecedents of τ_1 and τ_2 have the same cores (a path of length 2) and thus are homomorphically equivalent. Hence, τ_1 and τ_2 are part of the same HE-component S_1. The tgd τ_3 belongs to a different HE-component S_2 with $S_2 > S_1$. Indeed, there is a homomorphism sending $P(v_1, v_2)$ either to the antecedent of τ_1 or the antecedent of τ_2, but not vice versa. For the same reason, τ_4 gives rise to yet another HE-component S_3 with $S_1 > S_3$. In total, Σ has three HE-components. As far as the "closure above" is concerned, we thus have $Cl_{\geq}(S_1, \Sigma) = \{\tau_1, \tau_2, \tau_3\}$, $Cl_{\geq}(S_2, \Sigma) = \{\tau_3\}$, and $Cl_{\geq}(S_3, \Sigma) = \Sigma$.

Lemma 17. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ and $\Upsilon = \Upsilon_{st} \cup \Upsilon_t$ be two logically equivalent schema mappings. Moreover, let S be an HE-component in Σ_{st} and let T be an HE-component in Υ_{st}, s.t. S and T are homomorphically equivalent. Then $Cl_{\geq}(S, \Sigma) \cup \Sigma_t \equiv Cl_{\geq}(T, \Upsilon) \cup \Upsilon_t$ holds.*

Proof. By Lemma 9 we have $\Sigma_t \equiv \Upsilon_t$. It remains to show that, for every $\tau \in Cl_{\geq}(T, \Upsilon)$, the implication $Cl_{\geq}(S, \Sigma) \cup \Sigma_t \models \tau$ holds. The implication $Cl_{\geq}(T, \Upsilon_{st}) \cup \Upsilon_t \models \sigma$ for every $\sigma \in S$ follows by symmetry.

By $\Sigma \equiv \Upsilon$, we clearly have $\Sigma \models \tau$. Let $\varphi(\overline{x})$ denote the frozen antecedent of τ and let $I = At(\varphi(\overline{x}))$. Now consider the instance $chase(I, \Sigma_{st})$: Clearly, only those tgds $\sigma \in \Sigma_{st}$ fire, for which there is a homomorphism from the antecedent of σ to $\varphi(\overline{x})$. These are precisely the tgds in $Cl_{\geq}(S, \Sigma_{st})$. Hence, we have $chase(I, \Sigma_{st}) = chase(I, Cl_{\geq}(S, \Sigma))$. But then, by the implication criterion of [7] recalled in Lemma 10, $\Sigma \models \tau$ holds iff $Cl_{\geq}(S, \Sigma) \cup \Sigma_t \models \tau$ holds. □

The following lemma shows that, unless a schema mapping contains redundant dependencies, the HE-components of a schema mapping are in a sense invariant under logical equivalence. Moreover, HE-components may be exchanged between logically equivalent mappings.

Lemma 18. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ and $\Upsilon = \Upsilon_{st} \cup \Upsilon_t$ be two logically equivalent schema mappings, s.t. Rule E1 is not applicable to them. Let $\mathcal{S} = \{S_1, \ldots, S_m\}$ denote the HE-components of Σ_{st} and $\mathcal{T} = \{T_1, \ldots, T_n\}$ the HE-components of Υ_{st}. Then the following properties hold: $n = m$ and for every $S_i \in \mathcal{S}$, there exists exactly one j, s.t. the tgds in S_i are homomorphically equivalent to the tgds in T_j.*

Proof. W.l.o.g. suppose that there exists an HE-component S_i of Σ_{st} which is not homomorphically equivalent to any HE-component T_j of Υ_{st}. By assumption, $\Sigma \equiv \Upsilon$. Hence, $\Upsilon \models S_i$. Let $\mathcal{T}^* \subseteq \mathcal{T}$ with $\mathcal{T}^* = \bigcup\{T_j \mid T_j \geq S_i\}$. By the same considerations as in the proof of Lemma 17, only the HE-components in $\mathcal{T}^*$ are used to test the implication $\Upsilon \models S_i$ via Lemma 10. Hence, we have $\mathcal{T}^* \models S_i$.

On the other hand, also $\Sigma \models \mathcal{T}^*$. Now define $\mathcal{S}^* = \bigcup\{S_k \mid S_k \geq T_j \text{ for some } T_j \in \mathcal{T}^*\}$. Again, we may conclude $\mathcal{S}^* \models \mathcal{T}^*$ and, therefore, also $\mathcal{S}^* \models S_i$ By assumption, Υ does not contain an HE-component whose tgds are homomorphically equivalent to S_i. Therefore, all HE-components in $\mathcal{T}^*$ are strictly greater than S_i. But then, all HE-components S_k in $\mathcal{S}^*$ are also strictly greater than S_i. Thus, $\Sigma \setminus S_i \models S_i$. In other words, every dependency in S_i can be removed from Σ_{st} by Rule E1, which contradicts the assumption that Rule E1 is not applicable. □

Lemma 19. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ and $\Upsilon = \Upsilon_{st} \cup \Upsilon_t$ be two logically equivalent schema mappings, s.t. Rule E1 is not applicable to them. Moreover, let S be an HE-component in Σ_{st} and let T be an HE-component in Υ_{st}, s.t. S and T are homomorphically equivalent. Then the logical equivalence $\Sigma \equiv (\Sigma_{st} \setminus S) \cup T \cup \Sigma_t$ holds (i.e., we may replace the HE-component S from Σ by the corresponding HE-component T from Υ).*

Proof. Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ denote the HE-components of Σ_{st} and $\mathcal{T} = \{T_1, \ldots, T_n\}$ the HE-components of Υ_{st}. By Lemma 18, we may assume w.l.o.g., that every S_i is homomorphically equivalent to T_i. Now let S and T of this lemma correspond to S_j and T_j, for some $j \in \{1 \ldots n\}$.

We apply Lemma 17 to all HE-components that are strictly greater than S_j resp. T_j: Let $I = \{i \mid S_i > S_j\}$. Clearly, $I = \{i \mid T_i > T_j\}$. For every $i \in I$, we have $Cl_{\geq}(S_i, \Sigma) \cup \Sigma_t \equiv Cl_{\geq}(T_i, \Upsilon) \cup \Upsilon_t$ by Lemma 17. Then also $(\bigcup_{i \in I} Cl_{\geq}(S_i, \Sigma)) \cup \Sigma_t \equiv (\bigcup_{i \in I} Cl_{\geq}(T_i, \Sigma)) \cup \Upsilon_t$ holds, i.e.: $(Cl_{\geq}(S_j, \Sigma) \setminus S_j) \cup \Sigma_t \equiv (Cl_{\geq}(T_j, \Sigma) \setminus T_j) \cup \Upsilon_t$, i.e., the HE-components *strictly* greater than S_j and T_j lead to logical equivalence.

Now if we apply Lemma 17 to S_j and T_j, we may conclude $Cl_{\geq}(S_j, \Sigma) \cup \Sigma_t \equiv Cl_{\geq}(T_j, \Upsilon) \cup \Upsilon_t$. By the above considerations, we may exchange in $Cl_{\geq}(T_j, \Upsilon)$ all HE-components that are *strictly* greater than T_j by the corresponding HE-components from Σ. That is, $Cl_{\geq}(S_j, \Sigma) \cup \Sigma_t \equiv (Cl_{\geq}(S_j, \Sigma) \setminus S_j) \cup T_j \cup \Upsilon_t$. By adding all remaining HE-components of Σ to both sides of the equivalence, we get the desired equivalence $\Sigma \equiv (\Sigma_{st} \setminus S) \cup T \cup \Sigma_t$. □

HE-components will turn out to be crucial for optimizing the s-t tgds. Indeed we show that for all optimization criteria considered here, local optimization inside every HE-component yields a global optimum.

Definition 15. *An optimization problem on sets of dependencies is called a* sum-minimization problem *if the goal of the optimization is to minimize a function F with the following property: (1) $F(\Sigma) \geq 0$ holds for every set of dependencies Σ and (2) for any two sets of dependencies Σ, Σ' with $\Sigma \cap \Sigma' = \emptyset$, we have $F(\Sigma \cup \Sigma') = F(\Sigma) + F(\Sigma')$.*

Clearly, all optimization criteria studied here (like cardinality-minimality, antecedent-minimality, conclusion-minimality, and variable-minimality, see Definition 1) are sum-minimization problems.

Definition 16. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ be a schema mapping, s.t. Rule E1 is not applicable to it, i.e., Σ contains no s-t tgd that may be deleted. Now consider a sum-minimization problem whose goal is to minimize some function F over sets of s-t tgds.*

We say that Σ is globally optimal *(or simply* optimal*) if, for every mapping $\Upsilon = \Upsilon_{st} \cup \Upsilon_t$ with $\Sigma \equiv \Upsilon$, we have $F(\Sigma) \leq F(\Upsilon)$.*

We say that Σ is locally optimal *if the following conditions are fulfilled: let $\Upsilon = \Upsilon_{st} \cup \Upsilon_t$ be an arbitrary mapping with $\Sigma \equiv \Upsilon$. Moreover, let S be an arbitrary HE-component of Σ and let T be the corresponding HE-component of Υ_{st}, s.t. S and T are homomorphically equivalent. Then $F(S) \leq F(T)$ holds.*

Theorem 8. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ be a schema mapping, s.t. Rule E1 is not applicable to it. Now consider a sum-minimization problem whose goal is to minimize some function F over sets of s-t tgds. Then Σ is* globally optimal *iff it is* locally optimal.

Proof. Let $\Upsilon = \Upsilon_{st} \cup \Upsilon_t$ be an arbitrary mapping with $\Sigma \equiv \Upsilon$. By Lemma 18, there exist sets of s-t tgds $\mathcal{S} = \{S_1, \ldots, S_n\}$ and $\mathcal{T} = \{T_1, \ldots, T_n\}$, s.t. $\mathcal{S}$ denotes the set of HE-components of Σ_{st}, $\mathcal{T}$ denotes the set of HE-components of Υ_{st}, and for every i, the tgds in S_i are homomorphically equivalent to the tgds in T_i.

First suppose that Σ is *globally optimal*. We have to show that then Σ is also locally optimal. Assume to the contrary that $F(S_i) > F(T_i)$ holds for some i. We define $\Sigma' = (\Sigma_{st} \setminus S_i) \cup T_i \cup \Sigma_t$. By Lemma 19, $\Sigma \equiv \Sigma'$. Moreover, since we are considering a sum-minimization problem, we clearly have: $F(\Sigma_{st}) = F(\Sigma_{st} \setminus S_i) + F(S_i) > F(\Sigma_{st} \setminus S_i) + F(T_i) = F((\Sigma_{st} \setminus S_i) \cup T_i) = F(\Sigma')$. This contradicts the assumption that Σ is globally optimal.

Now suppose that Σ is *locally optimal*. We have to show that then Σ is also globally optimal The local optimality implies that $F(S_i) \leq F(T_i)$ holds for every i. Since F defines a sum-minimization problem, we have $F(\Sigma_{st}) = \sum_{i=1}^{n} F(S_i)$ and $F(\Upsilon_{st}) = \sum_{i=1}^{n} F(T_i)$. But then also $F(\Sigma_{st}) \leq F(\Upsilon_{st})$ holds, i.e., Σ is globally optimal. □

Theorem 8 says, that for the optimization of an HE-component, it does not matter how and if other HE-components have already been optimized. However, this does not mean that one can optimize a single HE-component in isolation. In particular, the closure above must be considered.

As demonstrated by the Examples 18 and 19, aggressive splitting and conclusion optimization lead to a non-unique normal form. In the rest of the section, we consider an operation opposite to splitting: Namely, merging of multiple s-t tgds, to enforce cardinality-minimality. As we will see, also this approach leads to non-unique normal forms. The following theorem contains a property that any merge operation must fulfil:

Theorem 9. *Consider a schema mapping $\Sigma = \Sigma_s \cup \Sigma_{st} \cup \Sigma_t$ resulting from a rewriting with the* PROPAGATE *procedure, and additionally reduced by the rules E1 and E2. Assume that dependency $\tau \in \Sigma_{st}$ with the antecedent φ can be substituted by the dependency τ' with the*

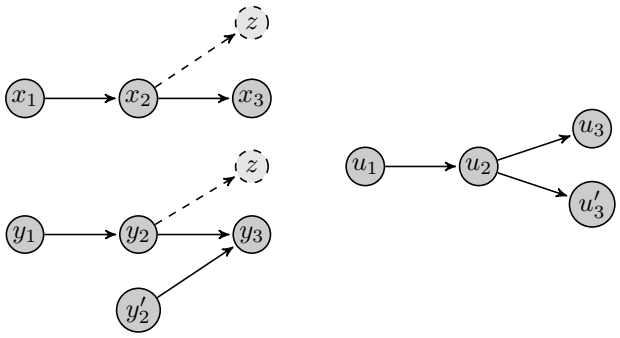

Figure 4.4: Possible merges of antecedents φ_1, φ_2, Example 21

antecedent φ', such that $\Sigma \cup \{\tau'\} \setminus \{\tau\} \equiv \Sigma$ holds, and $At(\varphi')$ does not cause a chase failure under Σ. Then, φ must coincide (up to isomorphism) with some endomorphic image of φ'.

Proof. By Theorem 6, exhaustive application of the rules E1 and E2 allows us to obtain a unique normal form of s-t dependencies. Hence, if the mapping $\Sigma' = \Sigma \cup \{\tau'\} \setminus \tau$ is logically equivalent to Σ, it is possible to bring it back in the form isomorphic to Σ by applying the procedure PROPAGATE, followed by the rules E1 and E2.

Since $At(\varphi')$ does not cause a chase failure, we know that PROPAGATE does not affect φ' in any way. Moreover, all the remaining rules in $\Sigma \setminus \{\tau\}$ remain unchanged after Σ' is transformed by E1 and E2. Hence, it must be the case that one can obtain τ from τ' by (possibly successive) applications of E2, and hence φ has to be among the endomorphic images of φ'. □

To achieve cardinality-minimality, we will replace each HE-component with a single tgd. As Theorem 9 suggests, the antecedent of this tgd must contain every antecedent from the original HE-component as an endomorphic image. The following example illustrates that there is no unique minimal "merged" antecedent.

Example 21. *Recall the schema mapping Σ from Example 20, with the HE-component S_1 containing tgds τ_1, τ_2:*

$$\tau_1 : P(x_1, x_2) \wedge P(x_2, x_3) \wedge \\ P(y_1, y_2) \wedge P(y_2, y_3) \wedge P(y'_2, y_3) \rightarrow Q(x_1, y_3)$$

$$\tau_2 : P(u_1, u_2) \wedge P(u_2, u_3) \wedge P(u_2, u'_3) \rightarrow Q(u_3, u'_3)$$

Let φ_1, φ_2 denote the antecedents of τ_1 and τ_2, respectively. Recall the graphical representation of φ_1 and φ_2 that was given in Figure 4.3. Obviously, φ_1 and φ_2 are not isomorphic.

Now, there are two ways of adding a single edge to φ_1 in order to get a minimum conjunctive query containing both antecedents as its endomorphic images namely, $\varphi'_1 = \varphi_1 \wedge P(x_2, z)$ and $\varphi''_1 = \varphi_1 \wedge P(y_2, z)$, see Figure 4.4. Clearly, the resulting antecedents φ'_1 and φ''_1 are not isomorphic.

Procedure MERGE

Input. A set Φ of homomorphically equivalent CQs;
Output. CQ φ having endomorphism onto each $\varphi_i \in \Phi$.

while $|\Phi| > 1$ **do**
  Choose distinct $\varphi_i, \varphi_j \in \Phi$.
  Find an endomorphism e_i for φ_i and e_j for φ_j, such that
    $e_i(\varphi_i) \cong e_j(\varphi_j)$ and $|e_i(\varphi_i)|$ is maximized.
  Let λ be a variable renaming $e_i(\varphi_i) \to e_j(\varphi_j)$.
  Set $\varphi_{ij} := \varphi_i\lambda \wedge \varphi_j$.
  Set $\Phi := \Phi \cup \{\varphi_{ij}\} \setminus \{\varphi_i, \varphi_j\}$.
od;
return the element remaining in Φ;

Procedure MERGETGDS

Input. A schema mapping $\Sigma = \Sigma_s \cup \Sigma_{st} \cup \Sigma_t$, a CQ χ;
Output. A set Δ_s^Σ of source egds and a set $\Delta_{st}^\Sigma = \{\tau\}$ with an s-t tgd,
  s.t. the equivalence $\Sigma \equiv (\Sigma \setminus \Sigma[\chi]) \cup \Delta_s^\Sigma \cup \Delta_{st}^\Sigma$ holds.

/* (a) collect and merge the antecedents of $\Sigma[\chi]$ */
  Set $\Phi = \{\varphi_i | \, (\varphi_i(\overline{x}_i) \to (\exists \overline{y}_i)\, \psi(\overline{x}_i, \overline{y}_i) \in \Sigma_{st}) \wedge \varphi_i \leftrightarrow \chi\}$;
  $I := chase(At(\text{MERGE}\ (\Phi)), \Sigma_s)$

/* (b) initialize τ' */
  $J := chase(I, \Sigma_{st})$.
  Let $\overline{y}$ be a tuple of all labelled nulls from $var(J) \setminus var(I)$
  $\tau' := \bigwedge_{A \in I} A \to (\exists \overline{y})\ \bigwedge_{B \in J} B$.

/* (c) propagate Σ_t into $ant(\tau')$ */
  return $\langle \Delta_s^\Sigma, \Delta_{st}^\Sigma \rangle :=$ PROPAGATE $(ant(\tau'), \Sigma)$.

Figure 4.5: Procedures MERGE and MERGETGDS.

Example 21 shows that there is no unique optimal way of merging s-t tgds from a single HE-component. Notably, egds play no role here. On the other hand, an obvious unique (though hardly optimal) way of merging would be to take a conjunction of all antecedents in a HE-component of a schema mapping resulting from the exhaustive application of the rules E1 and E2, and renaming apart the variables in distinct tgds.

We conclude this discussion by presenting a procedure that merges several homomorphically equivalent conjunctive queries in one, of reasonable size and satisfying the condition of Theorem 9. At every iteration, the procedure MERGE takes two conjunctive queries φ_i and φ_j, finds a greatest common (up to isomorphism) endomorphic image in them, which in the worst case is the core, and renames the variables of φ_i in such a way that it is stitched to φ_j along this greatest common endomorphic subquery. The resulting query φ_{ij} is thus sure to have an endomorphism to φ_i as well as to φ_j.

This operation is then used in the Procedure MERGETGDS, which produces an s-t tgd to substitute a given HE-component in a mapping Σ. To build the conclusion of such a merged tgd, MERGETGDS uses the PROPAGATE procedure, which chases the merged antecedent with Σ and then takes the conjunction of atoms in the core of the resulting target instance as the conclusion.

Definition 17. *Let Σ be a set of s-t tgds and let χ be a conjunctive query. Then we write $\Sigma[\chi]$ to denote the HE-component of those tgds in Σ, whose antecedents are homomorphically equivalent to χ.*

Theorem 10. *Let $\Sigma = \Sigma_s \cup \Sigma_{st} \cup \Sigma_t$ be a schema mapping consisting of source egds, s-t tgds and target egds, Σ_s and Σ_{st} resulting from a rewriting with the* PROPAGATE *procedure, and let χ be a CQ. Moreover, let $\langle \Delta_s^\Sigma, \Delta_{st}^\Sigma \rangle$ be the output of* MERGETGDS *(Σ, χ). Then, the following equivalence holds: $\Sigma \equiv (\Sigma \setminus \Sigma[\chi]) \cup \Delta_s^\Sigma \cup \Delta_{st}^\Sigma$.*

Proof (Sketch). Consider an s-t tgd τ' which has been created in step (b) of MERGETGDS by chasing the merged antecedent I with Σ. By construction of τ', $\Sigma \models \tau'$ holds and thus $\Sigma \equiv \Sigma \cup \{\tau'\}$. Then, $\Sigma \models \Sigma \cup \Delta_s^\Sigma \cup \Delta_{st}^\Sigma$ follows by Lemma 12, as both Δ_s^Σ and Δ_{st}^Σ have been produced by applying PROPAGATE to the antecedent of τ'.

In the other direction, we show that $\tau' \models \tau_i$ for any $\tau_i \in \Sigma[\chi]$. Then, by Lemma 13 we immediately obtain $\Delta_s^\Sigma \cup \Delta_{st}^\Sigma \models \tau_i$. In order to prove $\tau' \models \tau_i$ we will use the implication criterion from Lemma 1.

Let τ' and τ_i have the form $\varphi(\overline{x}) \rightarrow (\exists \overline{y})\ \psi(\overline{x}, \overline{y})$ and $\varphi_i(\overline{x}_i) \rightarrow (\exists \overline{y}_i)\ \psi_i(\overline{x}_i, \overline{y}_i)$, respectively. Then, the following two claims hold.

1. there exists a substitution $\sigma_1 : \overline{x} \rightarrow \overline{x}_i \cup \mathit{Const}$ s.t. $At(\varphi(\overline{x}\sigma_1)) = At(\varphi_i(\overline{x}_i))$;
2. there exists a variable renaming $\sigma_2 : \overline{x}_i \rightarrow \overline{x}$ s.t. $At(\varphi_i(\overline{x}\sigma_2)) \subseteq At(\varphi(\overline{x}))$.

Both claims follow directly from the definition of the procedure MERGE. Moreover, combining the two claims it is straightforward to show that there exists some endomorphic image K_φ of $At(\varphi(\overline{x}))$ that can be transformed in $At(\varphi_i(\overline{x}_i))$ by renaming of nulls. Let λ denote a respective variable renaming, and let λ_0 be the endomorphism s.t. $At(\varphi(\overline{x}\lambda_0)) = K_\varphi$.

Note that the conclusion of τ' has been produced in step (b) of MERGETGDS by chasing $At(\varphi(\overline{x}))$ with Σ. Since $\tau_i \in \Sigma$, the chase step with τ_i and λ^{-1} has been part of that chase, where λ^{-1} is the inverse of λ. Hence, the conclusion of τ' contains the conclusion of τ_i under the isomorphism $\lambda^{-1} \cup \mu$, where $\mu : \overline{y}_i \rightarrow \overline{y}'$, with $\overline{y}' \subseteq \overline{y}$, captures the assignment instantiating the existentially quantified variables $\overline{y}_i$ of τ_i with fresh labelled nulls, in the chase step with τ_i and λ^{-1}. In total, we have a situation when the mappings $\lambda' : \overline{x} \rightarrow \overline{x}_i \cup \mathit{Const}$ and $\mu^{-1} : \overline{y}_i \rightarrow \overline{y}$ exist s.t. $At(\varphi(\overline{x}\lambda')) = At(\varphi_i(\overline{x}_i))$ and $At(\psi_i(\overline{x}_i, \overline{y}_i\mu^{-1})) \subseteq At(\psi(\overline{x}\lambda', \overline{y}))$, where $\lambda' = \lambda_0 \circ \lambda$. This satisfies the preconditions of Lemma 1, and thus $\tau \models \tau_i$. □

Example 22. *Recall the tgds τ_1 and τ_2 with the antecedents φ_1 and φ_2 from Example 21. As illustrated by that example, there are two possible ways to merge φ_1 and φ_2, resulting in two merged antecedents $\varphi_1' = \varphi_1 \wedge P(x_2, z)$ and $\varphi_1'' = \varphi_1 \wedge P(y_2, z)$. The corresponding outputs of the procedure* MERGETGDS *are*

$$\tau_1' : P(y_1, y_2) \wedge P(y_2, y_3) \wedge P(y_2', y_3) \wedge \\ P(x_1, x_2) \wedge P(x_2, x_3) \wedge P(x_2, z) \rightarrow Q(x_1, y_3) \wedge Q(x_3, z)$$

and

$$\tau_1'' : P(y_1, y_2) \wedge P(y_2, y_3) \wedge P(y_2', y_3) \wedge P(y_2, z) \wedge$$
$$P(x_1, x_2) \wedge P(x_2, x_3) \rightarrow Q(x_1, y_3) \wedge Q(y_3, z),$$

respectively.

4.5 Summary

The following lessons have been learned from our analysis of the normalization and optimization of s-t tgds in the presence of egds: In contrast to the tgd-only case, we have seen that one has to be very careful with the definition of splitting and optimization so as not to produce a non-unique normal form: If we aim at a strict generalization of the splitting rule from Chapter 3 via the Rule ES in Figure 4.2, then there does not exist a unique normal form. This also happens if we aim at a strict generalization of the Rule 5 (deletion of redundant atoms from the conclusion of a tgd) via the Rule E3 in Figure 4.2. For most purposes, we therefore consider the transformation of an arbitrary mapping (consisting of s-t tgds and target egds) via the PROPAGATE procedure and exhaustive application of the rules E1 and E2 from Figure 4.2 followed by the Rules 1 – 5 from Chapter 3 as the best choice: The resulting normal form is unique up to isomorphism and incorporates a reasonable amount of splitting and simplification. From the splitting point of view, the resulting normal form is referred to as "antecedent-split-reduced". This corresponds to a restriction of the splitting rule in the tgd-only case to those situations where subsequent antecedent simplifications of all resulting dependencies are possible. Such a restriction is justifiable by the fact that one of the main motivations for splitting is indeed to further reduce the antecedents. From the optimization point of view, the Rules 1–5 guarantee that we do not perform worse than in the tgd-only case. But of course, this leaves some additional potential of further optimization in the presence of egds (in particular the Rule E3) unexploited.

We have also identified the HE-components (components of tgds with homomorphically equivalent antecedents) as an important handle for the most common optimization tasks on the s-t tgds (in particular, for all optimization criteria according to Definition 1). We have seen that a global optimum according to the optimization criteria studied here is obtained by locally optimizing the s-t tgds inside each HE-component. In particular, this allowed us to define a simple procedure which transforms a schema mapping into an equivalent one with the smallest possible number of s-t tgds. Of course, also in this case, uniqueness is not guaranteed.

We have entirely concentrated on the normalization and optimization of the s-t tgds, while a transformation of the egds has not been considered. Indeed, a normal form of the (source or target) egds is not important for our purposes since we will show in Theorem 11 that the unique (up to isomorphism) canonical universal solution in data exchange only depends on the normalization of the s-t tgds – the equivalence of the source egds and the concrete syntax of the egds are irrelevant.

CHAPTER 5

Application to answering aggregate queries

As an application for schema mapping normalization, in this chapter we discuss the semantics and evaluation of aggregate queries in data exchange, that is, queries of the form SELECT f FROM R, where f stands for one of the aggregate operators $\mathsf{min}(R.A)$, $\mathsf{max}(R.A)$, $\mathsf{count}(R.A)$, $\mathsf{count}(*)$, $\mathsf{sum}(R.A)$, or $\mathsf{avg}(R.A)$, and where R is a target relation symbol or, more generally, a conjunctive query over the target schema and A is an attribute of R. For this purpose, we first recall some basic notions on query answering in data exchange as well as some fundamental results on aggregate queries from [1].

5.1 Certain answers

Though any target database satisfying the schema mapping and local constraints is called a "solution", a random choice of a candidate for materializing a target database is not satisfactory: query answering in data exchange cannot be reduced to evaluating queries against random solutions. The widely accepted approach is based on the notion of *certain answers*. Here, we will generalize the definition given in Section 2.6:

Definition 18. *Let Σ be a schema mapping over the schema $\langle \mathbf{S}, \mathbf{T} \rangle$, and let I be an instance over $\mathbf{S}$. Then, the certain answers for a query q over $\mathbf{T}$ and for the source instance I are*

$$certain(q, I, \mathcal{W}(I)) = \bigcap \{q(J) | J \in \mathcal{W}(I)\},$$

where $\mathcal{W}(I)$ is the set of possible worlds *for I and Σ.*

Several proposals can be found in the literature [22,34,39,40] as to which solutions should be taken as possible worlds $\mathcal{W}(I)$. Typical examples are the set of all solutions, the set of universal solutions, the core of the universal solutions, or the CWA-solutions. For conjunctive queries, all these proposals lead to identical results.

5.2 Aggregate certain answers

Afrati and Kolaitis [1] initiated the study of the semantics of aggregate queries in data exchange. They adopted the notion of *aggregate certain answers* for inconsistent databases of Arenas et al. [4] to data exchange:

Definition 19. *[1] Let q be a query of the form* `SELECT` f `FROM` R*, where R is a target relation symbol or, more generally, a first-order query over the target schema $\mathbf{T}$, and f is one of the following aggregate operators:* $\mathsf{min}(R.A)$*,* $\mathsf{max}(R.A)$*,* $\mathsf{count}(R.A)$*,* $\mathsf{count}(*)$*,* $\mathsf{sum}(R.A)$*, or* $\mathsf{avg}(R.A)$ *for some attribute A of R. For all aggregate operators but* $\mathsf{count}(*)$*, tuples with a null value in attribute $R.A$ are ignored in the computation.*

- *Value r is a* possible answer *of q w.r.t. I and $\mathcal{W}(I)$ if there exists an instance $J \in \mathcal{W}(I)$ for which $f(q)(J) = r$.*
- *$poss(f(q), I, \mathcal{W}(I))$ denotes the set of all possible answers of the aggregate query $f(q)$ w.r.t. I and $\mathcal{W}(I)$.*
- *For the aggregate query $f(q)$, the aggregate certain answer agg-certain$(f, I, \mathcal{W}(I))$ w.r.t. I and $\mathcal{W}(I)$ is the interval*
  $[\mathsf{glb}(poss(f(q), I, \mathcal{W}(I))),\ \mathsf{lub}(poss(f(q), I, \mathcal{W}(I)))]$*, where* glb *and* lub *stand, respectively, for the greatest lower bound and the least upper bound.*

5.3 Semantics of aggregate queries via endomorphic images

A key issue in defining the semantics of queries in data exchange is to define which set of possible worlds should be considered. In [1], Afrati and Kolaitis showed that all previously considered sets of possible worlds yield a trivial semantics of aggregate queries. Therefore, they introduced a new approach via the *endomorphic images of the canonical universal solution*. Let $Endom(I, \mathcal{M})$ denote the endomorphic images of the canonical universal solution $J^* = CanSol(I)$, i.e.: $J \in Endom(I, \mathcal{M})$ if there exists an endomorphism $h\colon J^* \to J^*$, s.t. $J = h(J^*)$. As shown in [1], taking $\mathcal{W}(I) = Endom(I, \mathcal{M})$ leads to an interesting and non-trivial semantics of aggregate queries. However, in general, the semantics definition depends on the concrete syntactic representation of the s-t tgds.

Example 23. *Consider the source schema $\mathbf{S} = \{P\}$, target schema $\mathbf{T} = \{R\}$ and the pair of schema mappings $\mathcal{M}_1 = \langle \mathbf{S}, \mathbf{T}, \Sigma_1 \rangle$ and $\mathcal{M}_2 = \langle \mathbf{S}, \mathbf{T}, \Sigma_2 \rangle$ with the following s-t tgds:*
$\Sigma_1 = \{P(x) \to (\exists y) R(1, x, y)\}$ *and*
$\Sigma_2 = \{P(x) \to (\exists y_1 \dots y_n) R(1, x, y_1) \wedge \dots \wedge R(1, x, y_n)\}$

Clearly, $\mathcal{M}_1$ and $\mathcal{M}_2$ are logically equivalent. However, for the source instance $I = \{P(a)\}$, they yield different canonical universal solutions $J_1 = \{R(1, a, y)\}$ and $J_2 = \{R(1, a, y_1), \dots, R(1, a, y_n)\}$, respectively, where y and $y_1, \dots, y_n$ are distinct labelled nulls. Let A denote the

name of the first attribute of R. Then all of the three aggregate queries $\mathsf{count}(R.A)$, $\mathsf{count}(*)$, *and* $\mathsf{sum}(R.A)$ *have the range semantics* $[1,1]$ *in* $\mathcal{M}_1$ *and* $[1,n]$ *in* $\mathcal{M}_2$, *i.e.:* $\mathcal{M}_1$ *admits only one possible world, in which all three aggregate queries evaluate to 1. In contrast,* $\mathcal{M}_2$ *gives rise to a number of possible worlds with* $\{R(1,a,y_1), \ldots, R(1,a,y_n)\}$ *being the biggest one and* $\{R(1,a,y)\}$ *the smallest. Thus, the aggregate queries may take values between 1 and* n.

In order to eliminate the dependence on the concrete syntactic representation of the s-t tgds, we have defined a new normal form of s-t tgds in Definition 12. Below, we show that we thus get a unique canonical universal solution also in the presence of target egds.

Theorem 11. *Let* $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$, *be a schema mapping consisting of a set* $\Sigma = \Sigma_{st} \cup \Sigma_t$ *of s-t tgds and target egds and let* $\Sigma^* = \Sigma_s \cup \Sigma^*_{st} \cup \Sigma_t$ *be the normal form of* Σ. *Moreover, let* I *be a source instance and* J^* *the canonical universal solution for* I *under* $\mathcal{M}$ *obtained via a chase with* Σ^*_{st} *followed by a chase with* Σ_t *in arbitrary order. Then* J^* *is unique up to isomorphism. We denote* J^* *as* $CanSol^*(I)$.

Proof. By the logical equivalence between Σ and Σ^*, the chase with Σ fails iff the chase with Σ^* fails. We may thus restrict ourselves to the case that both chases succeed. Suppose that the chase of J with Σ (resp. Σ^*) consists of n (resp. n^*) egd-applications and write J_i (resp. J^*_i) to denote the intermediate result after the i-th step with $i \in \{0, \ldots, n\}$ (resp. $i \in \{0, \ldots, n^*\}$). In particular, $J = J_0 = J^*_0$. Clearly, every J_i and J^*_i is a homomorphic image of J.

Egds have the effect that variables may disappear from J. We therefore concentrate on the *positions* in J. To this end, we assume that every atom A in J is equipped with a unique identifier $id(A)$. A position in J is thus uniquely determined by $id(A)$ of an atom A and a position in A (i.e., an index between 1 and the arity of the predicate symbol of A). We assume that duplicate atoms, which may be produced by the chase, are not deleted. Then the positions in J persist in all instances J_i and J^*_i, even though variables from J may disappear in J_i and J^*_i. The application of an egd $\varepsilon\colon \varphi(\overline{x}) \rightarrow z_1 = z_2$ to an instance J_i (resp. J^*_i) means that the variables in ε are bound by some substitution $\sigma\colon \overline{x} \rightarrow Const \cup var(J)$. This substitution σ is determined by assigning each atom in $\varphi(\overline{x})$ to an atom in J_i. Thus, every variable occurrence in ε is assigned to some position in J. We can thus represent σ as a mapping from the variables in ε to tuples of positions in J: For $\overline{x} = \{x_1, \ldots, x_k\}$, σ is of the form $\sigma = \{x_1 \leftarrow (p_{11}, \ldots, p_{1j_1}), \ldots, x_k \leftarrow (p_{k1}, \ldots, p_{kj_k})\}$ with $j_1, \ldots, j_k \geq 1$, where the $p_{\alpha\beta}$'s denote positions in J. If a variable x_α occurs more than once in $\varphi(\overline{x})$ then it is mapped to several positions. Clearly, σ is well-defined for an instance J_i only if for every $\alpha \in \{1, \ldots, k\}$, all positions $p_{\alpha 1}, \ldots, p_{\alpha j_\alpha}$ have identical values in J_i.

In order to describe the instances resulting from the application of egds to J, we introduce the notion of *equality graphs*: The vertices of these equality graphs are the positions in J. We say that an equality graph *corresponds* to an instance J' (which was obtained from J by the application of some egds) if the following equivalence holds: *Two vertices corresponding to positions* p_1 *and* p_2 *in* J *are connected (not necessarily adjacent) if* p_1 *and* p_2 *have the*

same value in J'. Obviously, if two instances J' and J'' obtained from J via egds are such that the corresponding equality graphs have the same connected components, then J' and J'' are isomorphic. Thus, in order to show that $chase(J, \Sigma)$ and $chase(J, \Sigma^*)$ are isomorphic, it suffices to show that the equality graphs corresponding to $chase(J, \Sigma)$ and $chase(J, \Sigma^*)$ have the same connected components.

We construct the equality graph $\mathcal{E}_i$ of J_i (and, analogously the graph $\mathcal{E}_i^*$ corresponding to J_i^*) inductively as follows: The vertices never change, i.e., in every graph $\mathcal{E}_i$, there is one vertex for each position in J. By slight abuse of notation, we thus identify the vertices with the positions. As edges, we introduce in the graph $\mathcal{E}_0$ corresponding to $J = J_0$ an edge between any two (vertices corresponding to) positions p_1 and p_2 if they have the same value in J. Suppose that we have already constructed J_{i-1}. Then J_i is constructed as follows: Suppose that the egd $\varphi(\overline{x}\,) \rightarrow z_1 = z_2$ with $z_1, z_2 \in \overline{x}$ is applied in the i-th chase step. By the above considerations, this means that we apply a substitution $\sigma = \{x_1 \leftarrow (p_{11}, \ldots, p_{1j_1}), \ldots, x_k \leftarrow (p_{k1}, \ldots, p_{kj_k})\}$ to the variables $\overline{x} = \{x_1, \ldots, x_k\}$, i.e., every variable in $\varphi(\overline{x}\,)$ is mapped to one or more positions in J_{i-1} (or, equivalently, positions in J), s.t. for every $\alpha \in \{1, \ldots, k\}$, all positions $p_{\alpha 1}, \ldots, p_{\alpha j_\alpha}$ have identical values in J_{i-1}. Note that z_j with $j \in \{1, 2\}$ is a variable $x_\alpha \in \overline{x}$. We thus choose as vertex v_j in $\mathcal{E}_{i-1}$ some position $p_{\alpha\beta}$ for non-deterministically selected $\beta \in \{1, \ldots, j_\alpha\}$. Then $\mathcal{E}_i$ is obtained from $\mathcal{E}_{i-1}$ by inserting an edge between v_1 and v_2. It can be easily verified that every $\mathcal{E}_i$ is an equality graph corresponding to J_i. By the above considerations, it suffices to show that $\mathcal{E}_n$ and $\mathcal{E}_{n^*}^*$ have the same connected components. We prove by induction on $i \in \{0, \ldots, n^*\}$ that any two vertices v_1, v_2 connected in $\mathcal{E}_i^*$ are also connected in $\mathcal{E}_n$. The proof that any two vertices v_1, v_2 connected in $\mathcal{E}_i$ are also connected in $\mathcal{E}_{n*}^*$ is symmetric.

"$i = 0$". $\mathcal{E}_0'$ is the initial equality graph corresponding to J. By construction, every edge in $\mathcal{E}_0^* = \mathcal{E}_0$ is contained in $\mathcal{E}_n$.

"$(i-1) \rightarrow i$". Suppose that any two vertices connected in $\mathcal{E}_{i-1}^*$ are also connected in $\mathcal{E}_n$. We have to show that then also any two vertices connected in $\mathcal{E}_i^*$ are connected in $\mathcal{E}_n$. By construction, $\mathcal{E}_i^*$ contains at most one additional edge compared with $\mathcal{E}_{i-1}^*$, say (v_1, v_2). We have to show that v_1 and v_2 are also connected (not necessarily adjacent) in $\mathcal{E}_n$. Let $\varepsilon\colon \varphi(\overline{x}\,) \rightarrow z_1 = z_2$ with $z_1, z_2 \in \overline{x}$ denote the egd which was applied in the i-th chase step with Σ^*, i.e., a substitution $\sigma = \{x_1 \leftarrow (p_{11}, \ldots, p_{1j_1}), \ldots, x_k \leftarrow (p_{k1}, \ldots, p_{kj_k})\}$ was applied to the variables $\overline{x} = \{x_1, \ldots, x_k\}$. The remainder of the proof proceeds in three steps:

(1) The substitution σ is also well-defined for $chase(J, \Sigma)$

(2) $chase(J, \Sigma) \models \varphi(\overline{x}\sigma)$

(3) The vertices v_1, v_2 are connected in $\mathcal{E}_n$.

Proof of (1). Let $\alpha \in \{1, \ldots, k\}$. We have to show that the positions $p_{\alpha 1}, \ldots, p_{\alpha j_\alpha}$ have identical values in $chase(J, \Sigma)$. Note that σ is well-defined as a mapping of $\overline{x}$ to J_{i-1}^* since this substitution was applied for the i-th chase step with Σ^*. Hence, the vertices $p_{\alpha 1}, \ldots, p_{\alpha j_\alpha}$ are connected in $\mathcal{E}_{i-1}^*$. But then, by the induction hypothesis, they are also connected in $\mathcal{E}_n$. This

means that the positions $p_{\alpha 1}, \ldots, p_{\alpha j_\alpha}$ indeed have identical values in $chase(J, \Sigma)$.

Proof of (2). Let $A(\overline{x}\,)$ be a conjunct in $\varphi(\overline{x}\,)$. We have to show that $chase(J, \Sigma) \models A(\overline{x}\sigma)$. Clearly, $J^*_{i-1} \models A(\overline{x}\sigma)$ since σ was applied for the i-th chase step with Σ^*, i.e., there exists an atom $B \in J$ with identifier $id(B)$, s.t. the atom B (i.e., precisely speaking, the atom with identifier $id(B)$) in J^*_{i-1} coincides with $A(\overline{x}\sigma)$. We claim that then $A(\overline{x}\sigma)$ also coincides with the atom B in $chase(J, \Sigma)$: By definition, σ as a mapping to $chase(J, \Sigma)$ maps the variables in $\overline{x}$ to the (values at the) same positions like σ as a mapping to J^*_{i-1}. Hence, if a variable occurring in $A(\overline{x})$ is mapped to some position of B in J^*_{i-1} then it is mapped to the same position of B in $chase(J, \Sigma)$. By (1), if some variable occurs in several positions in $A(\overline{x})$, then the corresponding positions of B have identical values in $chase(J, \Sigma)$. It thus remains to show that if some constant c occurs at a position p in A then B in $chase(J, \Sigma)$ also has the value c at this position. Since $A(\overline{x}\sigma)$ coincides with B in J^*_{i-1}, B has the value c at position p in J^*_{i-1}, i.e., there exists a position q, s.t. p and q are connected in $\mathcal{E}^*_{i-1}$ (this also comprises the case that p and q are identical) and the value at position q in J was c. Note, that the (constant) value at position q can never change during the chase. Moreover, by the induction hypothesis, p is connected with q also in $\mathcal{E}_n$. Thus, B in $chase(J, \Sigma)$ also has the value c at the position p.

Proof of (3). Recall the construction of $\mathcal{E}^*_i$. Namely, let ε be an egd $\varphi(\overline{x}\,) \rightarrow z_1 = z_2$ with z_1 being a variable x_α and z_2 a variable x_γ. Then v_1 is some position $p_{\alpha\beta}$ and v_2 is some position $p_{\gamma\delta}$ according to the substitution σ. In other words, the edge (v_1, v_2) was introduced in $\mathcal{E}^*_i$ in order to enforce the equality $z_1\sigma = z_2\sigma$ in J^*_i.

We have to show that v_1, v_2 are connected in $\mathcal{E}_n$, where v_1 is the position $p_{\alpha\beta}$ and v_2 is the position $p_{\gamma\delta}$. By assumption, Σ and Σ^* are equivalent. Hence, since $chase(J, \Sigma) \models \Sigma$, also $chase(J, \Sigma) \models \Sigma^*$ holds. In particular, $chase(J, \Sigma) \models \varepsilon$. By the above considerations, $chase(J, \Sigma) \models \varphi(\overline{x}\sigma)$. Hence, the equality $z_1\sigma = z_2\sigma$ must also be fulfilled in $chase(J, \Sigma)$ and, therefore, the values at the positions $p_{\alpha\beta}$ and $p_{\gamma\delta}$ are identical in $chase(J, \Sigma)$. Thus, the vertices v_1 and v_2 are indeed connected in $\mathcal{E}_n$. □

To obtain a unique range semantics of the aggregate functions min, max, count, count$(*)$, sum, and avg, we therefore propose to follow the approach of [1], with the only difference that we take the unique target instance $CanSol^*(I)$ from Theorem 11.

Part II

Relaxed notions of equivalence

CHAPTER 6

Introduction and background

Two main results in this part of the thesis are (i) decidability of optimizations based on DE-equivalence for restricted classes of schema mappings (with target dependencies), which constitutes the subject of Chapter 7, and (ii) the undecidability of CQ-equivalence testing for Second-Order tgds, considered in Chapter 8.

In the current chapter, we provide a context for these results. Our starting point will be the role of relaxed notions of equivalence for optimizing source-to-target dependencies.

6.1 Optimization of s-t tgds

This chapter extends the results first formulated in [54] for mappings with target egds only.

Let schema mappings include no target dependencies. Then, as shown by Fagin et al., the notions of logical, DE- and CQ-equivalence coincide.

Theorem 12 ([23])**.** *Let $\mathcal{M}_1, \mathcal{M}_2$ be mappings given by sets of s-t tgds. Then, $\mathcal{M}_1 \equiv \mathcal{M}_2$ iff $\mathcal{M}_1 \equiv_{CQ} \mathcal{M}_2$ iff $\mathcal{M}_1 \equiv_{DE} \mathcal{M}_2$.*

But what happens if the mappings $\mathcal{M}_1$ and $\mathcal{M}_2$ have target dependencies? We start by considering the setting similar to that in Chapter 4, namely, when the set of target constraints in the mappings is fixed up to logical equivalence.

Lemma 20. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ and $\Upsilon = \Upsilon_{st} \cup \Upsilon_t$ be schema mappings with $\Sigma \equiv_{CQ} \Upsilon$. Moreover, for an arbitrary conjunction $\varphi(\bar{x})$, let $\Delta^{\Sigma} = \Delta^{\Sigma}_s(\varphi(\bar{x})) \cup \Delta^{\Sigma}_{st}(\varphi(\bar{x}))$ and $\Delta^{\Upsilon} = \Delta^{\Upsilon}_s(\varphi(\bar{x})) \cup \Delta^{\Upsilon}_{st}(\varphi(\bar{x}))$, respectively, denote the output of the* PROPAGATE *procedure. Then $\Delta^{\Sigma} \equiv \Delta^{\Upsilon}$ holds.*

Proof. We even show a slightly stronger result, namely: If the assumption of the lemma holds, then exactly the following two cases are possible:

- $\Delta_{st}^{\Sigma}(\varphi(\bar{x})) = \Delta_{st}^{\Upsilon}(\varphi(\bar{x})) = \emptyset$ and both $\Delta_{s}^{\Sigma}(\varphi(\bar{x}))$ and $\Delta_{s}^{\Upsilon}(\varphi(\bar{x}))$ contain a single egd of the form $\varphi(\bar{x}) \rightarrow c = c'$ for distinct constants c, c' possibly different for $\Delta_{s}^{\Sigma}(\varphi(\bar{x}))$ and $\Delta_{s}^{\Upsilon}(\varphi(\bar{x}))$, or
- (1) $\Delta_{s}^{\Sigma}(\varphi(\bar{x})) = \Delta_{s}^{\Upsilon}(\varphi(\bar{x}))$ and (2) τ and τ' are identical up to variable renaming for $\Delta_{st}^{\Sigma}(\varphi(\bar{x})) = \{\tau\}$ and $\Delta_{st}^{\Upsilon}(\varphi(\bar{x})) = \{\tau'\}$.

For the former case, let any of $\Delta_{st}^{\Sigma}(\varphi(\bar{x}))$, $\Delta_{st}^{\Upsilon}(\varphi(\bar{x}))$ be empty. This can happen if the consistency check at step 1 of the PROPAGATE procedure fails, and the chase of $At(\varphi(\bar{x}))$ with Σ resp. Υ causes a unification of distinct constants. Assume w.l.o.g. that $\Delta_{st}^{\Sigma}(\varphi(\bar{x})) = \emptyset$ that is, chasing $At(\varphi(\bar{x}))$ with Σ enforces the unification of distinct constants c, c' occurring in $\varphi(\bar{x})$ or in Σ. We show that also the chase of $At(\varphi(\bar{x}))$ with Υ causes unification of distinct constants occurring in $\varphi(\bar{x})$ or in Υ. Assume to the contrary, no such constant clash takes place in the chase of $At(\varphi(\bar{x}))$ with Υ. Then, there exists a substitution λ for $\bar{x}$ and a "Skolemization" I' of $At(\varphi(\bar{x}\lambda))$ assigning distinct fresh constants to the variables in $\bar{x}\lambda$, such that the chase of I' with Υ succeeds. However, the same egds that fire in the chase of $At(\varphi(\bar{x}))$ with Σ are still applicable when I' is chased, and thus the unsatisfiable equality $c = c'$ also causes this chase to fail. We obtain a contradiction to the assumption $\Sigma \equiv_{CQ} \Upsilon$. Thus, we have shown that if either of $\Delta_{st}^{\Sigma}(\varphi(\bar{x}))$, $\Delta_{st}^{\Upsilon}(\varphi(\bar{x}))$ is empty, then necessarily both sets are, and $\Delta_{s}^{\Sigma}(\varphi(\bar{x}))$, $\Delta_{s}^{\Upsilon}(\varphi(\bar{x}))$ contain a single egd of the form $\varphi(\bar{x}) \rightarrow c = c'$.

For the latter case, when both $\Delta_{st}^{\Sigma}(\varphi(\bar{x}))$ and $\Delta_{st}^{\Upsilon}(\varphi(\bar{x}))$ contain a single tgd τ resp. τ', we first introduce some useful notation: We write $J^{\Sigma} = J_S^{\Sigma} \cup J_T^{\Sigma}$ and $J^{\Upsilon} = J_S^{\Upsilon} \cup J_T^{\Upsilon}$ to denote the result of chasing $I = At(\varphi(\bar{x}))\}$ with Σ respectively Υ.

(1) $\Delta_{s}^{\Sigma}(\varphi(\bar{x})) = \Delta_{s}^{\Upsilon}(\varphi(\bar{x}))$: Suppose to the contrary that $\Delta_{s}^{\Sigma}(\varphi(\bar{x})) \neq \Delta_{s}^{\Upsilon}(\varphi(\bar{x}))$ holds. W.l.o.g., assume that there exists a source egd $\varphi(x) \rightarrow x_i = x_j$ in $\Delta_{s}^{\Sigma}(\varphi(\bar{x})) \setminus \Delta_{s}^{\Upsilon}(\varphi(\bar{x}))$. By the construction of $\Delta_{s}^{\Sigma}(\varphi(\bar{x}))$ respectively $\Delta_{s}^{\Upsilon}(\varphi(\bar{x}))$ in step 3 of the PROPAGATE procedure, the egd $\varphi(x) \rightarrow x_i = x_j$ is satisfied by J_S^{Σ} but not by J_S^{Υ}.

We construct the source instance I' by "Skolemizing" J_S^{Υ}, i.e., the variables in J_S^{Υ} are replaced by pairwise distinct, fresh constants. By this construction, we know that I' satisfies $\Delta_{s}^{\Upsilon}(\varphi(\bar{x}))$. Moreover, the chase of I with Υ can be replayed on I'. This chase of I' with Υ clearly succeeds since all necessary equalities (i.e., the equalities enforced by $\Delta_{s}^{\Upsilon}(\varphi(\bar{x}))$ indeed hold in I'. In contrast, I' violates $\Delta_{s}^{\Sigma}(\varphi(\bar{x}))$. Hence, by Lemma 12, the chase of I' with Σ fails. This contradicts the assumption $\Sigma \equiv_{CQ} \Upsilon$.

(2) $\Delta_{st}^{\Sigma}(\varphi(\bar{x})) = \{\tau\}$ and $\Delta_{st}^{\Upsilon}(\varphi(\bar{x})) = \{\tau'\}$, the s-t tgds τ and τ' are identical up to variable renaming: By part (1) of the proof, the source egds $\Delta_{s}^{\Sigma}(\varphi(\bar{x}))$ and $\Delta_{s}^{\Upsilon}(\varphi(\bar{x}))$ coincide. Hence, $J_S^{\Sigma} = J_S^{\Upsilon}$. By step 2 of PROPAGATE, the s-t tgds τ and τ' are of the form $\tau = \varphi(\bar{x}\lambda) \rightarrow (\exists \bar{y})\psi(\bar{x}\lambda, \bar{y})$ and $\tau' = \varphi(\bar{x}\lambda) \rightarrow (\exists \bar{z})\chi(\bar{x}\lambda, \bar{z})$, respectively, for some substitution λ, s.t. $At(\varphi(\bar{x}\lambda)) = J_S^{\Sigma} = J_S^{\Upsilon}$.

Let I' denote the source instance obtained by "Skolemizing" J_S^{Σ} (or, equivalently, J_S^{Υ}), i.e., the variables in J_S^{Σ} are replaced by pairwise distinct, fresh constants. Thus, $I' = At(\varphi(\bar{x}\lambda\mu))$, where μ is a substitution that sends each variable x_i present in I' to a fresh constant c_i.

As in part (1) of the proof, the chase of I with Σ (resp. Υ) can be replayed on I'. Since I' satisfies $\Delta_s^{\Sigma}(\varphi(\bar{x}))$ (resp. $\Delta_s^{\Upsilon}(\varphi(\bar{x}))$), this chase of $\langle I', \emptyset \rangle$ with Σ (resp. Υ) succeeds and yields a pair of instances $\langle I', J'_{\Sigma} \rangle$ (resp. $\langle I', J'_{\Upsilon} \rangle$), s.t. J'_{Σ} is obtained from J_S^{Σ} (resp. J'_{Υ} is obtained from J_S^{Υ}) by applying the substitution μ to the variables in $\overline{x}\lambda \cap J_S^{\Sigma}$ (resp. $\overline{x}\lambda \cap J_S^{\Upsilon}$).

Let $C_{\Sigma} = \mathit{core}(J'_{\Sigma})$ and $C_{\Upsilon} = \mathit{core}(J'_{\Upsilon})$. Then we have $C_{\Sigma} = At(\psi(\bar{x}\lambda\mu, \bar{y}))$ and $C_{\Upsilon} = At(\chi(\bar{x}\lambda\mu, \bar{z}))$, respectively. This is due to the fact that J'_{Σ} (resp. J'_{Υ}) is obtained from J_S^{Σ} (resp. J_S^{Υ}) via the "Skolemization" substitution μ and, therefore, the core C_{Σ} of J'_{Σ} (resp. core C_{Υ} of J'_{Υ}) can be obtained from J_S^{Σ} (resp. J_S^{Υ}) by first computing the core (considering the variables x_i as constants) and then applying substitution μ. But, by step 2 of PROPAGATE, $\mathit{core}(J'_{\Sigma}) = At(\psi(\bar{x}\lambda, \bar{y}))$ respectively $\mathit{core}(J'_{\Upsilon}) = At(\chi(\bar{x}\lambda, \bar{z}))$ holds.

C_{Σ} and C_{Υ} are homomorphically equivalent, since we are assuming $\Sigma \equiv_{CQ} \Upsilon$. For cores, homomorphical equivalence and isomorphism coincide. Hence, there exists a variable renaming η from $\overline{y}$ to $\overline{z}$, s.t. $At(\psi(\bar{x}\lambda\mu, \bar{y}\eta)) = At(\chi(\bar{x}\lambda\mu, \bar{z}))$. But then also $\psi(\bar{x}\lambda\mu, \bar{y}\eta) = \chi(\bar{x}\lambda\mu, \bar{z})$, i.e., τ and τ' are identical up to variable renaming η. □

We now want to combine the above result with Lemmas 12 and 13 to show that if two mappings are CQ-equivalent, then their difference can be reduced to the target dependencies (while the s-t tgds can be replaced by common source egds and s-t tgds). Since DE-equivalence implies CQ-equivalence, this clearly holds for DE-equivalent mappings as well. Before we formalize this idea in Theorem 13 below, it is convenient to prove the following lemma. It is a slight generalization of Lemma 12 and tells us that all s-t tgds in a schema mapping may be replaced by the output produced by successive calls of PROPAGATE (provided that PROPAGATE is called with all CQs $\varphi(x)$ that occur as antecedents of the s-t tgds in Σ).

Lemma 21. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ be a schema mapping and let Φ be a set of CQs, s.t. $\{\varphi(x) \mid \varphi(x)$ is the antecedent of some $\sigma \in \Sigma_{st}\} \subseteq \Phi$. Moreover, let $\Sigma_s^* = \bigcup_{\varphi(x)\in\Phi} \Delta_s^{\Sigma}(\varphi(x))$ and $\Sigma_{st}^* = \bigcup_{\varphi(x)\in\Phi} \Delta_{st}^{\Sigma}(\varphi(x))$. Then $\Sigma \equiv \Sigma_s^* \cup \Sigma_{st}^* \cup \Sigma_t$.*

Proof. Let $\Sigma_{st} = \{\sigma_1, \ldots, \sigma_m\}$ and let $(\varphi_1, \ldots, \varphi_m)$, s.t. for each $i \in \{1, \ldots, m\}$, φ_i is the antecedent of σ_m. Then Φ is of the form $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ for some $n \geq m$.

We define Σ_0:=Σ_{st} and, for each $i \in \{1, \ldots, m\}$, we set Σ_i:=$(\Sigma_{i-1} \setminus \{\sigma_i\}) \cup \Delta_s^{\Sigma_{i-1}}(\varphi_i) \cup \Delta_{st}^{\Sigma_{i-1}}(\varphi_i)$. By Lemma 12 and 13, we have the logical equivalence $\Sigma_i \cup \Sigma_t \equiv \Sigma_{i-1} \cup \Sigma_t$ for each $i \in \{1, \ldots, m\}$. Hence, $\Sigma = \Sigma_0 \cup \Sigma_t \equiv \Sigma_m \cup \Sigma_t$.

Note that $\Sigma_m = \bigcup_{i=1}^m \Delta_s^{\Sigma_{i-1}}(\varphi_i) \cup \bigcup_{i=1}^m \Delta_{st}^{\Sigma_{i-1}}(\varphi_i)$. Moreover, the logical equivalence $\Sigma_i \cup \Sigma_t \equiv \Sigma_{st} \cup \Sigma_t$ implies CQ-equivalence $\Sigma_i \cup \Sigma_t \equiv_{CQ} \Sigma_{st} \cup \Sigma_t$. Hence, by Lemma 20, $\Delta_s^{\Sigma_{i-1}}(\varphi_i) \equiv \Delta_s^{\Sigma}(\varphi_i)$ and $\Delta_{st}^{\Sigma_{i-1}}(\varphi_i) \equiv \Delta_s^{\Sigma}(\varphi_i)$ holds. Hence, $\Sigma \equiv \Sigma_m \cup \Sigma_t \equiv \bigcup_{i=1}^m \Delta_s^{\Sigma}(\varphi_i) \cup \bigcup_{i=1}^m \Delta_{st}^{\Sigma}(\varphi_i)$ and, by Lemma 12, $\Sigma \equiv \Sigma_m \cup \Sigma_t \equiv \bigcup_{i=1}^n \Delta_s^{\Sigma}(\varphi_i) \cup \bigcup_{i=1}^n \Delta_{st}^{\Sigma}(\varphi_i)$. □

Theorem 13. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ and $\Upsilon = \Upsilon_{st} \cup \Upsilon_t$ with $\Sigma \equiv_{CQ} \Upsilon$. Then there exist a set Σ_s^* of source egds and a set Σ_{st}^* of s-t tgds, s.t. $\Sigma \equiv \Sigma_s^* \cup \Sigma_{st}^* \cup \Sigma_t$ and $\Upsilon \equiv \Sigma_s^* \cup \Sigma_{st}^* \cup \Upsilon_t$.*

Proof. The idea is to call PROPAGATE with every antecedent occurring in $\Sigma_{st} \cup \Upsilon_{st}$ and to take Σ_s^* (resp. Σ_{st}^*) as the set of all source egds (resp. all s-t tgds) produced by these procedure calls. Formally, we set $\Phi = \{\varphi(x) \mid \varphi(x) \text{ is the antecedent of some } \sigma \in \Sigma_{st}\} \cup \{\varphi(x) \mid \varphi(x) \text{ is the antecedent of some } \sigma \in \Upsilon_{st}\}$ and $\Sigma_s^* = \bigcup_{\varphi(x)\in\Phi} \Delta_s^{\Sigma}(\varphi(x))$ and $\Sigma_{st}^* = \bigcup_{\varphi(x)\in\Phi} \Delta_{st}^{\Sigma}(\varphi(x))$.

Then, by Lemma 21, $\Sigma \equiv \Sigma_s^* \cup \Sigma_{st}^* \cup \Sigma_t$. Likewise, $\Upsilon \equiv \Upsilon_s^* \cup \Upsilon_{st}^* \cup \Upsilon_t$ holds for $\Upsilon_s^* = \bigcup_{\varphi(x)\in\Phi} \Delta_s^{\Upsilon}(\varphi(x))$ and $\Upsilon_{st}^* = \bigcup_{\varphi(x)\in\Phi} \Delta_{st}^{\Upsilon}(\varphi(x))$.

By $\Sigma \equiv_{CQ} \Upsilon$ and Lemma 20, $\Delta_s^{\Sigma}(\varphi(x)) \equiv \Delta_s^{\Upsilon}(\varphi(x))$ and $\Delta_{st}^{\Sigma}(\varphi(x)) \equiv \Delta_{st}^{\Upsilon}(\varphi(x))$ holds for every $\varphi(x) \in \Phi$. Hence, $\Sigma_s^* \equiv \Upsilon_s^*$ and $\Sigma_{st}^* \equiv \Upsilon_{st}^*$. Thus, $\Upsilon \equiv \Sigma_s^* \cup \Sigma_{st}^* \cup \Upsilon_t$. □

With this theorem at hand, we can now settle the question of the decidability of DE- and CQ-equivalence of schema mappings with logically equivalent (and, in particular, with identical) target dependencies. Again, it is convenient first to prove a lemma.

Lemma 22. *Let $\Sigma = \Sigma_{st} \cup \Sigma_t$ and $\Upsilon = \Upsilon_{st} \cup \Upsilon_t$, such that $\Sigma \equiv_{CQ} \Upsilon$ and $\Sigma_t \equiv \Upsilon_t$ holds. Then $\Sigma \equiv \Upsilon$.*

Proof. By Theorem 13, there exist a set of source egds ${\Sigma_t}^*$ and a set of s-t tgds Σ_{st}^* such that $\Sigma \equiv \Sigma_s^* \cup \Sigma_{st}^* \cup \Sigma_t$ and $\Upsilon \equiv \Sigma_s^* \cup \Sigma_{st}^* \cup \Upsilon_t$. The claim follows from $\Sigma_t \equiv \Upsilon_t$. □

Theorem 14. *Suppose that the problems of DE- and CQ-equivalence are restricted to pairs of schema mappings $\Sigma = \Sigma_{st} \cup \Sigma_t$ and $\Upsilon = \Upsilon_{st} \cup \Upsilon_t$, s.t. Σ_t and Υ_t are* logically equivalent. *With this restriction, the DE- and CQ-equivalence problems $\Sigma \equiv_{DE} \Upsilon$ and $\Sigma \equiv_{CQ} \Upsilon$, respectively, are decidable.*

Proof. By the condition $\Sigma_t \equiv \Upsilon_t$ and by Lemma 22, $\Sigma \equiv \Upsilon$ holds iff $\Sigma \equiv_{CQ} \Upsilon$ holds. Hence, also $\Sigma \equiv_{DE} \Upsilon$ coincides with logical equivalence. □

In other words, the relaxed notions of equivalence are decidable on schema mappings if only the s-t tgds are allowed to vary. This naturally raises the question if the decidability result of Theorem 14 allows us to exploit additional possibilities (compared with logical equivalence) of optimizing the s-t tgds in the presence of target dependencies.

We now show for a broad class of optimization problems on s-t tgds that the relaxed notions of equivalence lead to exactly the same notion of optimality as logical equivalence. Negatively, this means that the relaxed notions of equivalence do not give us additional power. Positively, this means that for optimizing the s-t tgds one can interchangeably use algorithms for any of these notions of equivalence. Below, we carry over the notions of optimality to s-t tgds and generalize these notions to arbitrary, real-valued target functions. W.l.o.g., we restrict ourselves to minimization problems. For the sake of a uniform notation, in Definition 20 we denote logical equivalence by $\equiv_{log}$ rather than $\equiv$.

Definition 20. *Let $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$ with $\Sigma = \Sigma_{st} \cup \Sigma_t$ be a mapping, let $x \in \{log, DE, CQ\}$ and let F be a function that assigns a real number to every set of s-t tgds.*

Then the set of s-t tgds Σ_{st} is called F-optimal *w.r.t. x-equivalence, if there does not exist a set of s-t tgds Σ'_{st}, s.t. $F(\Sigma'_{st}) < F(\Sigma_{st})$ and $\Sigma_{st} \cup \Sigma_t \equiv_x \Sigma'_{st} \cup \Sigma_t$.*

Formally, we show below that, for any real-valued function F, the F-optimality w.r.t. DE- or CQ-equivalence coincides with the F-optimality w.r.t. logical equivalence.

Theorem 15. *Let $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$ with $\Sigma = \Sigma_{st} \cup \Sigma_t$ be a schema mapping and let F be a function that assigns a real number to every set of s-t tgds. Then Σ_{st} is F-optimal w.r.t. logical equivalence iff it is F-optimal w.r.t. DE-equivalence iff it is F-optimal w.r.t. CQ-equivalence.*

Proof. Logical equivalence entails DE-equivalence, which entails CQ-equivalence [23]. Hence, it suffices to show that if Σ_{st} is F-optimal w.r.t. logical equivalence then it is also F-optimal w.r.t. CQ-equivalence. Suppose to the contrary that Σ_{st} is F-optimal w.r.t. logical equivalence but not w.r.t. CQ-equivalence. Then there exists Σ'_{st}, s.t. $F(\Sigma'_{st}) < F(\Sigma_{st})$ and $\Sigma_{st} \cup \Sigma_t \equiv_{CQ} \Sigma'_{st} \cup \Sigma_t$. By Lemma 22, then also $\Sigma_{st} \cup \Sigma_t \equiv \Sigma'_{st} \cup \Sigma_t$ holds, which contradicts the assumption that Σ_{st} is F-optimal w.r.t. logical equivalence. □

We have shown that if the target dependencies are fixed, the notions of logical, DE- and CQ-equivalence coincide, in full analogy with the situation when mappings are specified by sets of source-to-target tgds alone. Now, we move on to a situation where the target dependencies are allowed to vary, and consider the relaxed notions of equivalence in this setting.

6.2 Overview of undecidability results

We start with an undecidability result that immediately follows from the undecidability of equivalence of Datalog programs [55]:

Theorem 16 ([23])**.** *CQ-equivalence is undecidable for mappings based on full s-t tgds and full target tgds.*

As shown in [49], this result can be extended to mappings with target egds (and no target tgds):

Theorem 17 ([49])**.** *CQ-equivalence is undecidable for schema mappings based on s-t tgds and target egds.*

Furthermore, Theorems 16 and 17 hold even if restricted to mappings with just a single target dependency each. The proof utilizes the result of Gottlob and Papadimitriou [29], who proved that for every Datalog program, one can find an (essentially) equivalent Datalog program which consists of a single rule (called sirup):

Theorem 18 ([49])**.** *CQ-equivalence is undecidable for schema mappings based on full s-t tgds and a single full target tgd, or s-t tgds and a single target egd.*

Already based on these results, one can receive an impression that CQ-equivalence is in general difficult to test in the presence of target constraints. Yet a closer investigation is necessary to support such a claim. In particular, decidability of specific optimization tasks, as well as decidability of testing CQ-equivalence for restricted classes of target dependencies must be investigated. In [49], most of these questions have been settled in the negative: both testing CQ equivalence and optimizing schema mappings w.r.t. CQ-equivalence is undecidable, even for very restricted target dependencies. In particular, CQ-equivalence becomes undecidable if target key dependencies are allowed:

Theorem 19 ([49]). *CQ-equivalence is undecidable for schema mappings based on s-t tgds and at most one key dependency per target relation.*

Therefore, it is important to clarify the role of DE- equivalence, a notion which is stricter than CQ-equivalence and yet more relaxed than logical equivalence. However, many results in [49] concerned with DE-equivalence are not particularly encouraging: in the presence of unrestricted target egds or full target tgds, DE-equivalence is undecidable as well as CQ-equivalence:

Theorem 20 ([49]). *DE-equivalence is undecidable for schema mappings based on s-t tgds and target egds.*

Furthermore, if target integrity constraints are encoded by (full) tgds, the undecidability holds even if the source-to-target tgds in the mappings are full as well:

Theorem 21 ([49]). *DE-equivalence is undecidable for schema mappings based on full s-t tgds and full target tgds.*

For mappings in which the source-to-target tgds are not necessarily full, the above results can be considerably strengthened:

Proposition 2 ([49]). *DE- and CQ-equivalence remain undecidable for schema mappings based on s-t tgds and either (1) full target tgds or (2) target egds, even if one of the following restrictions applies:*

- *constant symbols are not allowed in dependencies, or*
- *the sets of target dependencies are fixed.*

Note that in the second claim of Proposition 2, we are dealing with the situation where two mappings in question have two a priori known sets Σ_t^* and Σ_t^{**} of target dependencies. Of course, the two sets Σ_t^* and Σ_t^{**} must be *distinct*: Otherwise, both CQ- and DE-equivalence become decidable by Theorem 14.

Similarly, many natural optimization tasks are undecidable under DE-equivalence as well as under CQ-equivalence.

Theorem 22 ([49]). *Let $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$ be a a schema mapping with $\Sigma = \Sigma_{st} \cup \Sigma_t$ based on s-t tgds and target egds, or full s-t tgds and full target tgds. Then, the following optimization problems are undecidable w.r.t. DE- resp. CQ-equivalence.:*

- *Test if a given dependency $\tau \in \Sigma$ is redundant.*
- *Test if Σ is cardinality-minimal.*

Notably, cardinality-minimality remains undecidable even if the set of source-to-target dependencies is fixed. This is in contrast to the opposite situation considered in the previous section: Recall that if the target dependencies are fixed, CQ-, DE- and logical equivalence collapse, see Lemma 22.

Theorem 23 ([49]). *Let $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$ be a schema mapping with $\Sigma = \Sigma_{st} \cup \Sigma_t$ based on s-t tgds and target egds, or full s-t tgds and full target tgds. Then it is undecidable if Σ_t is cardinality-minimal w.r.t. DE- resp. CQ-equivalence.*

Finally, antecedent-minimality of target dependencies is also undecidable under the relaxed notions of equivalence:

Theorem 24 ([49]). *Let $\langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$ be a a mapping with $\Sigma = \Sigma_{st} \cup \Sigma_t$ of s-t tgds and target egds. Then it is undecidable if for all Σ'_t, s.t. $\Sigma_{st} \cup \Sigma_t \equiv_{CQ} \Sigma_{st} \cup \Sigma'_t$ (resp. $\Sigma_{st} \cup \Sigma_t \equiv_{DE} \Sigma_{st} \cup \Sigma'_t$), the total number of atoms in the antecedents in Σ_t is less than or equal to the total number of atoms in the antecedents in Σ'_t. That is, if Σ_t is antecedent-minimal w.r.t. DE- resp. CQ-equivalence.*

However, the situation changes completely when restrictions of target dependencies are considered. For a large class of target constraints DE-equivalence turns out to be decidable, and so are most important optimization tasks. This will be the subject of the next chapter.

CHAPTER 7

DE-equivalence: decidable case

Among the results outlined in Section 6.2 there was the undecidability of testing CQ-equivalence of mappings with target dependencies as weak as sets of key constraints. In this chapter, we show that DE-equivalence is decidable for mappings whose target dependencies belong to a considerably bigger class. The idea of the equivalence test is rather simple: given two mappings, we identify dependencies in one mapping which are not logically implied by the other mapping. For DE-equivalence, it is *sufficient* that no such "differing" dependency ever fires in any possible chase sequence (see Theorem 25). Moreover, we will show that for a certain class of target dependencies, this condition is also *necessary* for DE-equivalence.

Definition 21. *A query φ is said to be satisfiable in a mapping Σ if there exists a source instance I such that the chase of I with Σ terminates and succeeds, and $chase(I, \Sigma) \models \varphi$ holds.*

Definition 22. *Let Σ and Σ' be schema mappings. Dependency $\tau \in \Sigma \cup \Sigma'$ is said to be "differing", if it is not implied by one of the mappings, i.e., either $\Sigma \not\models \tau$ or $\Sigma' \not\models \tau$ holds.*

Definition 23 (AUC)**.** *We say that a pair of mappings Σ, Σ' meets the* antecedent unsatisfiability condition (AUC)*, if no differing dependency τ in any of Σ, Σ' has an antecedent satisfiable in the mapping which contains τ.*

Effectively, AUC makes Σ and Σ' indistinguishable for the chase. As it was shown by Theorem 13, for any two mappings $\Sigma = \Sigma_{st} \cup \Sigma_t$ and $\Sigma' = \Sigma'_{st} \cup \Sigma'_t$, whenever $\Sigma \equiv_{CQ} \Sigma'$ holds, there exist sets Σ^*_s of source egds and Σ^*_{st} of s-t tgds, such that $\Sigma \equiv \Sigma^*_s \cup \Sigma^*_{st} \cup \Sigma_t$ and $\Sigma' \equiv \Sigma^*_s \cup \Sigma^*_{st} \cup \Sigma'_t$. Since DE-equivalence implies CQ-equivalence, in this chapter we always assume that two mappings whose DE-equivalence is under consideration have exactly the same set of s-t tgds and source egds. As a consequence, by "differing dependency" we will always assume a target one. We also notice that the following simple proposition holds:

Proposition 3. *Let $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$ be a schema mapping s.t. $\Sigma = \Sigma_s \cup \Sigma_{st} \cup \Sigma_t$ is a set of source egds, s-t tgds and target tgds and egds. For any dependency τ over $\mathbf{T}$, $\Sigma \models \tau$ iff $\Sigma_t \models \tau$.*

Proof. "If" is by $\Sigma_t \subseteq \Sigma$, and "only if" is due to the fact that $\Sigma \setminus \Sigma_t = \Sigma_{st} \cup \Sigma_s$ is the set of source-to-target tgds and source egds, which, as it is easy to see from Lemma 10, have no impact on the implication test: having antecedents over $\mathbf{S}$, they never fire in the chase of the antecedent database of τ, and thus $\Sigma \models \tau$ only if $\Sigma_t \models \tau$. □

The following theorem puts AUC into effect, as illustrated by the mappings $\mathcal{M}$ and $\mathcal{M}_1$ from Example 8 in Section 1.1.

Theorem 25. *Let $\Sigma = \Sigma_s \cup \Sigma_{st} \cup \Sigma_t$ and $\Sigma' = \Sigma_s \cup \Sigma_{st} \cup \Sigma'_t$ be schema mappings with terminating chase property where the target dependencies consist of egds and tgds. If Σ and Σ' satisfy the AUC, then $\Sigma \equiv_{DE} \Sigma'$ holds.*

This result is not very surprising, if we recall that DE-equivalence is concerned with universal solutions, and that AUC makes two mappings indistinguishable for the chase, a primary tool for actually computing a universal solution. The following lemma helps to make this intuition precise:

Lemma 23. *Let $\Sigma = \Sigma_s \cup \Sigma_{st} \cup \Sigma_t$ and $\Sigma' = \Sigma_s \cup \Sigma_{st} \cup \Sigma'_t$ be two mappings between the schemas $\mathbf{S}$ and $\mathbf{T}$, and I an instance over $\mathbf{S}$, such that Σ' does not cause a chase failure on I. Let n be arbitrary. Assume that in the first n steps of the chase of I with Σ, only those dependencies fire which are logically implied by Σ'. Then, $chase_n(I, \Sigma) \to chase(I, \Sigma')$ holds.*

Proof. Follows from Lemma 3.4 of [22], which makes the following claim: Let K be an instance which satisfies a dependency τ, and let K_1 and K_2 be instances, such that the latter is produced from the former by a non-failing chase step enforcing τ. Then $K_1 \to K$ implies $K_2 \to K$.

By taking $K = chase(I, \Sigma')$, it is easy to see that (i) the chase of $K_0 = chase(I, \Sigma_{st})$ with Σ_t starts with an instance from which there is a homomorphism onto K, and (ii) K satisfies all the dependencies firing in this chase. Hence, $chase_n(I, \Sigma) \to K = chase(I, \Sigma')$ for every n such that only dependencies implied by Σ' are firing up to step n. □

We are now ready to prove Theorem 25:

Proof of Theorem 25. Suppose that the two mappings are not DE-equivalent: w.l.o.g., there is a source instance I and a target instance J such that $J \in UnivSol(I, \Sigma')$ and $J \notin UnivSol(I, \Sigma)$. We show that there must exist at least one differing target dependency violating the preconditions of the theorem.

Since J is a universal solution for Σ', $core(I, \Sigma') \cong core(J)$. Hence, the chase of I with Σ' must succeed. For Σ, we have the following two cases:

1. I is a counter-example to $\Sigma' =_{CQ} \Sigma$: the chase of I with Σ fails, or $core(I, \Sigma) \ncong core(I, \Sigma')$. Consider two parallel chases of I, one with Σ and another with Σ'. If neither sequence contains a firing of a differing dependency, then by Lemma 23, the chase of I with Σ succeeds (since the chase Σ' does) and moreover, $chase(I, \Sigma) \leftrightarrow chase(I, \Sigma')$ holds. Thus, $core(I, \Sigma) \cong core(I, \Sigma')$ which contradicts our assumption.

Hence, it must be the case that at least one differing dependency fires in the course of the chase of I with Σ or with Σ', thereby its antecedent must be satisfiable in the respective mapping.

2. I illustrates $\Sigma' =_{CQ} \Sigma$: $core(I, \Sigma) \cong core(I, \Sigma') \cong core(J)$. Then, $J \notin Sol(I, \Sigma)$ must be the case, as otherwise J would be a universal solution for Σ, which contradicts the assumption that J is a witness for $\Sigma \not\equiv_{DE} \Sigma'$.

Then, there must be a dependency $\tau \in \Sigma$ which is violated only by J but not by $core(J)$. Since $J \models \Sigma'$, it must be the case that $\Sigma' \not\models \tau$. It remains to show that $ant(\tau)$ is satisfiable in Σ. It is very easy to see, though: $J \models ant(\tau)$ (otherwise, J would not violate τ) and $J \leftrightarrow chase(I, \Sigma)$, since the two instances have isomorphic cores. Thus, $chase(I, \Sigma) \models ant(\tau)$. $\square$

It turns out that for DE-equivalent mappings, the AUC can be reformulated in terms of unsatisfiability in either mapping:

Theorem 26. *Let $\Sigma = \Sigma_s \cup \Sigma_{st} \cup \Sigma_t$ and $\Sigma' = \Sigma_s \cup \Sigma_{st} \cup \Sigma'_t$ be two DE-equivalent mappings. For any differing dependency $\tau \in \Sigma \cup \Sigma'$, let $\Sigma_\tau \in \{\Sigma, \Sigma'\}$ denote the mapping containing τ, and let $\Sigma_{\neg\tau}$ be the other mapping, $\Sigma_{\neg\tau} \not\models \tau$. Then, the following two statements are equivalent:*

A: For any differing dependency $\tau \in \Sigma \cup \Sigma'$, the antecedent of τ is unsatisfiable in $\Sigma_{\neg\tau}$;

B: (AUC) For any differing dependency $\tau \in \Sigma \cup \Sigma'$, the antecedent of τ is unsatisfiable in Σ_τ.

Proof. For convenience, we expand the formulation of the claim of the theorem. Specifically, (A) is equivalent to the following pair of statements:

A1: For any $\sigma \in \Sigma$, $\Sigma' \not\models \sigma$, and for any source I_σ, $chase(I_\sigma, \Sigma') \not\models ant(\sigma)$.
A2: For any $\sigma' \in \Sigma'$, $\Sigma \not\models \sigma'$, and for any source $I_{\sigma'}$, $chase(I_{\sigma'}, \Sigma) \not\models ant(\sigma')$.

Likewise, (B) is equivalent to $(B1 \wedge B2)$:

B1: For any $\tau \in \Sigma$, $\Sigma' \not\models \tau$, and for any source I_τ, $chase(I_\tau, \Sigma) \not\models ant(\tau)$.
B2: For any $\delta \in \Sigma'$, $\Sigma \not\models \delta$, and for any source I_δ, $chase(I_\delta, \Sigma') \not\models ant(\delta)$.

(A1 $\wedge$ A2) $\Rightarrow$ (B1 $\wedge$ B2). We only show that *(A1)* implies *(B1)*; *(A2) $\Rightarrow$ (B2)* follows immediately by symmetry.

Indeed, assume *(A1)* and suppose that *(B1)* does not hold: there exists such a $\tau \in \Sigma$, $\Sigma' \not\models \tau$ and a source instance I_τ, that $chase(I_\tau, \Sigma) \models ant(\tau)$.

Firstly, by $\Sigma \equiv_{DE} \Sigma'$, the chase of I_τ with Σ must succeed, as otherwise I_τ would be a counter-example to DE-equivalence of Σ and Σ'. We show by induction on n, that for no dependency $\delta \in \Sigma$ which is not implied by Σ', $chase_n(I_\tau, \Sigma) \models ant(\delta)$ can hold.

$n = 1$: $\nexists \delta \in \Sigma$ such that $\Sigma' \not\models \delta$ and $chase(I_\tau, \Sigma_{st}) \models ant(\delta)$. Otherwise, as source-to-target dependencies in Σ' and Σ coincide, and the chase of I_τ with Σ' succeeds, also $chase(I_\tau, \Sigma') \models ant(\delta)$ holds, which contradicts *(A1)*.

$n \Rightarrow n+1$: By induction hypothesis, up to step n of the chase of I_τ with Σ, no dependency that is not implied by Σ' fires. Then, by Lemma 23, it holds that $chase_n(I_\tau, \Sigma) \rightarrow chase(I_\tau, \Sigma')$.

But then also at step $n+1$ of the chase with Σ no dependency δ can fire such that $\Sigma' \not\models \delta$, as then $chase(I_\tau, \Sigma') \models ant(\delta)$ would be the case, which contradicts *(A1)*.

(B1 ∧ B2) ⇒ (A1 ∧ A2). We only prove *(A1)* assuming *(B1)* and *(B2)*, since *(B1 ∧ B2) ⇒ (A2)* is a symmetric case.

Suppose that *(A1)* does not hold, and there exists some differing dependency $\sigma \in \Sigma$, whose antecedent $ant(\sigma)$ is satisfiable in Σ': That is, $\Sigma' \not\models \sigma$ and for some source instance I_σ, $chase(I_\sigma, \Sigma') \models ant(\sigma)$.

Then, there must be some integer n, such that $chase_n(I_\sigma, \Sigma') \models ant(\sigma)$. At the same time, by *(B1)*, $chase(I_\sigma, \Sigma) \not\models ant(\sigma)$.

It must be the case that the chase with Σ succeeds on I_σ, as otherwise the DE-equivalence of Σ and Σ' would be immediately violated.

Now, by Lemma 23, we have that if in the course of the chase of I_σ with Σ' no dependency δ fires s.t. $\Sigma \not\models \delta$, then $chase(I_\sigma, \Sigma') \to chase(I_\sigma, \Sigma)$ holds, and thus also $chase(I_\sigma, \Sigma) \models ant(\sigma)$ must be the case, which contradicts *(B1)*. Hence, there must be some $\delta \in \Sigma'$ s.t. $\Sigma \not\models \delta$ and δ applies in the course of the chase of I_σ. Clearly, this implies that $ant(\delta)$ is satisfiable in Σ', and we have derived a contradiction to *(B2)*. □

Theorem 26 will allow us to show, that for some useful class of mappings, the antecedent unsatisfiability condition is also necessary for DE-equivalence. Namely, this is the case for mappings in which target dependencies are *constant-free and connected*:

Definition 24. *A conjunctive query is said to be connected if so is its dual graph (that is, the graph whose vertices correspond to the query atoms and an edge is drawn between two vertices whenever the respective atoms share a variable).*

An egd is connected if its antecedent is. Finally, a tgd is connected if the conjunction of its antecedent and conclusion is: both the antecedent and the conclusion are connected and share at least one variable.

We also extend the notion of connectedness to databases:

Definition 25. *Let I be an instance. The* dual graph *$\mathcal{G}_I$ of I has vertices associated with the atoms of I. An edge between $A(\bar{x})$ and $B(\bar{y})$ is drawn whenever $\bar{x}$ and $\bar{y}$ have a term in common (be it a constant or a labelled null). We say that I is connected if $\mathcal{G}_I$ is.*

The following lemma will be of help subsequently:

Lemma 24. *Consider a connected Boolean conjunctive query φ, and an instance I. If there exists a homomorphism h mapping each atom of φ onto a fact in I (if $I \models \varphi$, that is), the subinstance $h(\varphi)$ of I is connected.*

Proof. We apply induction on the size of φ. For an atomic φ, the claim is trivial. Assume now that all images of any connected Boolean CQ φ_n of size n are connected. Then, consider a query

φ_{n+1} obtained by augmenting φ_n with an additional atom A that shares at least one variable with some atom B in φ_n. Let h be a homomorphism mapping φ_{n+1} onto I. By the induction hypothesis, $h(B)$ belongs to a connected component of $h(\varphi_n) \subseteq I$. Clearly, by construction of φ_{n+1}, $h(A)$ must share a term with $h(B)$, and thus $h(\varphi_{n+1})$ is connected, too. □

Now, everything is set up to demonstrate AUC as a necessary condition for DE-equivalence:

Theorem 27. *Consider the mappings $\Sigma = \Sigma_s \cup \Sigma_{st} \cup \Sigma_t$ and $\Sigma' = \Sigma_s \cup \Sigma_{st} \cup \Sigma'_t$ in which both Σ_t and Σ'_t are constant-free and connected. Then, $\Sigma \equiv_{DE} \Sigma'$ implies that the two mappings satisfy the antecedent unsatisfiability condition.*

Proof. Assume that $\Sigma \equiv_{DE} \Sigma'$ holds, but the antecedent unsatisfiability condition is violated: that is, there exists a differing dependency δ in either of the mappings, with the antecedent satisfiable in the mapping which contains δ. Then, by Theorem 26, also the following statement holds: W.l.o.g. there exists $\tau \in \Sigma_t$ such that $\Sigma' \not\models \tau$ and the antecedent $\varphi(\bar{x})$ of τ is satisfiable in Σ'. Now if τ and all dependencies in Σ'_t are constant-free and connected, we derive a contradiction by showing $\Sigma \not\equiv_{DE} \Sigma'$:

Indeed, assume there exists a source instance I such that $\varphi(\bar{x})$ is satisfied in $J = chase(I, \Sigma')$. Let $[\varphi]$ be a database isomorphic to $\varphi(\bar{x})$: each atom of $\varphi(\bar{x})$ is turned into a fact in $[\varphi]$ by instantiating the variables with distinct labelled nulls not occurring in J.

Based on $[\varphi]$, we now build a counter-example to $\Sigma \equiv_{DE} \Sigma'$. We first note that the chase with both Σ and Σ' succeeds on $[\varphi]$, as the target dependencies in both mappings are constant-free, and $dom([\varphi])$ consists solely of labelled nulls.

Consider the instance $J_\varphi = chase([\varphi], \Sigma')$. We prove three properties, justifying that the instance $J' = J \cup J_\varphi$ is exactly a counter-example to DE-equivalence of Σ and Σ'.

a. $J' \not\models \Sigma_t$ and hence, $J' \notin Sol(I, \Sigma)$.

b. $J' \models \Sigma'$. Since $J \in Sol(I, \Sigma')$, we have $J' \in Sol(I, \Sigma')$.

c. There exists a homomorphism from J' onto any other solution for I under Σ'. Hence, $J' \in UnivSol(I, \Sigma')$.

(a) First note that $J_\varphi \not\models \tau$, since otherwise $\Sigma' \models \tau$ would be the case, by the dependency implication test from Lemma 10. We show now that also $J' \not\models \tau$ holds by arguing that the facts or equalities missing in J_φ cannot be provided by J. This is immediate if τ is an egd: Missing equalities in J_φ will violate τ irrespectively of any facts added to J_φ.

Let τ be a tgd. $J_\varphi \not\models \tau$ means that there exists an assignment λ for the variables of $\varphi(\bar{x})$, such that no assignment μ exists for which $At(\psi(\bar{x}\lambda, \bar{y}\mu)) \subseteq J_\varphi$ holds. We show that for this very assignment λ, there is also no assignment χ, such that $At(\psi(\bar{x}\lambda, \bar{y}\chi)) \subseteq J'$.

Indeed, by the connectedness condition on target dependencies, $\psi(\bar{x}, \bar{y})$ is connected. By Lemma 24, every image of $\psi(\bar{x}, \bar{y})$ must be connected, too. Recall that λ sends the atoms of $\varphi(\bar{x})$ onto J_φ, and that $dom(J_\varphi) \cap dom(J) = \emptyset$. It is thus not possible that $At(\psi(\bar{x}\lambda, \bar{y}\chi)) \subseteq J$ holds.

Moreover, since any instance combining the atoms from J_φ and J is necessarily disconnected, there is no χ s.t. $At(\psi(\bar{x}\lambda, \bar{y}\chi)) \subseteq J_\varphi \cup J$ holds.

(b) To show $J' \in Sol(I, \Sigma')$, we first note that both J and J_φ satisfy Σ'. It remains to show that no further dependencies from Σ' fire on their union J'. This follows from Lemma 24 and the fact that all target dependencies in Σ' are constant-free and have connected antecedents. Since $dom(J_\varphi) \cap dom(J) = \emptyset$, the antecedent of no target dependency in Σ' can be surjectively mapped onto any subinstance of J' which mixes facts from J and J_φ.

Finally, (c) is shown by proving $J' \to J$, as J is a canonical universal solution for I under Σ'. Recall that, by the initial assumption made in this proof, $J \models \varphi(\bar{x})$ and thus there is a homomorphism $[\varphi] \to J$, by construction of $[\varphi]$. It takes an easy inductive proof to show that after each chase step n of $[\varphi]$ with Σ', the homomorphism $chase_n([\varphi], \Sigma') \to J$ holds, using the Lemma 3.4 of [22] (cited in the proof of Lemma 23 above). □

Theorem 27 allows one to reduce the data exchange equivalence of mappings with terminating chase property, containing constant-free and connected target dependencies, to testing the satisfiability of conjunctive queries according to Definition 21. This is good news, since satisfiability can be efficiently decided for a broad class of mappings. In particular, we bring together the results of this chapter and Lemma 10 to establish the DE-equivalence as a useful tool for optimizing practically relevant schema mappings, given by *functional dependencies* (FDs) and *weakly acyclic sets of inclusion dependencies* (IDs).

Weak acyclicity (see [16, 22]) is a sufficient condition, ensuring that a chase of arbitrary source instance terminates in polynomial time (data complexity). This property is formalized by setting up the *dependency graph* G corresponding to a set Σ of tgds. The vertices of G are the positions R^i of every relation R in the schema. Let a variable x occur at position R^i in the antecedent of some tgd τ. Then there is an edge $\left(R^i, S^j\right)$ in G if either

1. x also occurs at position S^j in the conclusion of τ or
2. x occurs in in the conclusion of τ and there is an existentially quantified variable y at position S^j in the conclusion of τ.

Edges resulting from rule (2) are called *special*. A set of tgds is *weakly acyclic* if there is no cycle containing a special edge.

Theorem 28. *DE-equivalence of schema mappings with target dependencies consisting of FDs and IDs, and possessing the terminating chase property is decidable.*

Furthermore, suppose that the sets of IDs in these schema mappings are weakly-acyclic and that the number of terms in the antecedent resp. conclusion of s-t tgds in the mappings is bounded by some constant b. Then DE-equivalence of such mappings can be decided in polynomial time.

Proof. First note that FDs and IDs are always connected and constant-free. Then, given the mappings $\Sigma = \Sigma_{st} \cup \Sigma_t$ and $\Sigma' = \Sigma'_{st} \cup \Sigma'_t$ where both Σ_t and Σ'_t are restricted to FDs and IDs and possess the terminating chase property, the following procedure decides $\Sigma \equiv_{DE} \Sigma'$:

1. Transform Σ and Σ' into the form $\Sigma_s \cup \hat{\Sigma}_{st} \cup \Sigma_t$ resp. $\Sigma'_s \cup \hat{\Sigma}'_{st} \cup \Sigma'_t$ using the PROPAGATE procedure. By Theorem 13, if $\Sigma_s \not\equiv \Sigma'_s$ or $\hat{\Sigma}_{st} \not\equiv \hat{\Sigma}'_{st}$, conclude $\Sigma \not\equiv_{DE} \Sigma'$.
2. Conclude $\Sigma \equiv_{DE} \Sigma'$ if the pair (Σ, Σ') meets the antecedent unsatisfiability condition and $\Sigma \not\equiv_{DE} \Sigma'$ otherwise.

The complexity estimation rests upon the fact that both steps of the decision procedure take polynomial time if the sets of tgds in the mappings are weakly acyclic and the dependency size is bounded. For the s-t dependencies this bound is b, and for target inclusion and target dependencies it is implicit in the size of the schema, which we consider fixed. Let d_0 be the maximum of the two bounds. We do then one further adaptation: as a final dependency size bound d, we take $max(d_0, c)$, where c is the maximal number of antecedent resp. conclusion atoms in the dependencies in Σ resp. Σ'. (Note, that as b limits a number of antecedent resp. conclusion terms, and a schema is fixed, there is only a fixed number of different atoms which can be constructed, and thus c can be considered fixed as well).

Let D be the maximal number of dependencies in Σ resp. Σ'. For correctness of the complexity estimation, the following two claims are essential:

Claim 1: *Let weakly-acyclic mapping Σ be defined as above and consist of at most D dependencies, and let a source instance I have size n. Then, the size of $chase(I, \Sigma)$ is polynomial both in D and in n.*
This is shown implicitly by the proof of Theorem 3.9 [22], which claims that the number of steps in a chase sequence is polynomial in the size of the source instance.

Claim 2: *Let T be an instance of size polynomial in D. Then for each $\tau \in \Sigma \cup \Sigma'$, deciding $T \models \tau$ takes $poly(D)$ time, i.e., the required time is a polynomial in D.*
Indeed, there are at most $|dom(T)|^d$ ways in which the antecedent of τ can be mapped onto T. For each such application, we have to check that also the conclusion of τ is satisfied, which amounts to evaluating a CQ of bounded size.

We now analyze the running time of the two steps of the above decision procedure.

Step 1 of the decision procedure. Applying PROPAGATE to an instance with at most d facts (the antecedent of some s-t tgd) comes down to chasing it with D dependencies. Let J_T denote the chase result. Since d is constant, by Claim 1 the size of J_T is $poly(D)$. Step 2 uses J_T to construct the new s-t tgd τ. This is preceded by computing the core of J_T, which is an expensive operation.[1] However, for weakly-acyclic sets of tgds of bounded size, core computation is feasible in polynomial time. First note that *depth* p of the chase, i.e., the maximal

[1] In principle, core computation could be omitted: Every result presented so far would also hold if J_T is directly converted into the conclusion of τ.

number of existential edges in any path in the dependency graph of Σ is bounded by the number of vertices in this graph (every path containing a special edge is acyclic), that is, depends on the schema $\mathbf{T}$ only and thus can be considered fixed. The core computation algorithm of [51] performs iterations consisting of evaluating a query of size $O(d^p) = O(1)$ against J_T. These iterations then replay the chase of J_T, thereby obtaining a homomorphism from J_T in its proper subset. Finally, the homomorphism must be transformed into an idempotent one, which takes $O(|dom(J_T)|^2)$ time. After less than $|dom(J_T)|^2$ iterations, the core is found. Thus, the second step of PROPAGATE is also feasible in $poly(D)$ time.

Assume that unifications of the source variables performed by the chase are recorded at step 1 of the PROPAGATE. Then, also the 3rd step, producing the source egds is feasible in constant time, since all egds use the same antecedent and there is only a fixed number of variables to equate. In total, each call of the PROPAGATE procedure takes $poly(D)$ time, and there is no more than $2D$ calls, for each antecedent of the s-t tgd in Σ resp. Σ'. As a result, the sets $\hat{\Sigma}_{st}$ and $\hat{\Sigma}'_{st}$ have at most $2D$ dependencies, and the size of Σ_s resp. Σ'_s is $O(D)$.

Checking $\Sigma_s = \Sigma'_s$, by Lemma 10, amounts to chasing at most $2D$ antecedents of source egds (of bounded size) with $O(D)$ source egds, which takes time $O(D^2)$.

In contrast with source egds, the dependencies in $\hat{\Sigma}_{st}$ resp. $\hat{\Sigma}'_{st}$ can have conclusions of size $poly(D)$, by Claim 1. Let $\tau \in \hat{\Sigma}_{st}$ be such an s-t tgd, and it has to be found if $\hat{\Sigma}'_{st} \models \tau$ holds. For that, we chase the instance $A_\tau = At(ant(\tau))$ with $\hat{\Sigma}'_{st}$ which by Claim 1 results in the target instance J of polynomial size. Then, by Lemma 10, it remains to test $\langle A_\tau, J\rangle \models \tau$.

Recall that the conclusion of τ has been produced by the PROPAGATE procedure by chasing A_τ with Σ. Hence, instead of testing $\langle A_\tau, J\rangle \models \tau$, we can as well check $\langle A_\tau, J\rangle \models \Sigma$. If the latter condition fails, a counter-example to DE-equivalence has been found; otherwise, we can conclude $\langle A_\tau, J\rangle \models \tau$. Since the size of the combined database $\langle A_\tau, J\rangle$ is polynomial in D, by Claim 2 we know that $\langle A_\tau, J\rangle \models \Sigma$ can be decided in polynomial time, too.

Step 2 of the decision procedure. Testing the antecedent unsatisfiability condition amounts to (a) finding the differing dependencies in Σ_t, Σ'_t and (b) testing the satisfiability of the antecedents of the identified dependencies.

(a) Consider a dependency $\tau \in \Sigma_t$. By Lemma 10, the implication test consists of two phases: (i) chasing of the antecedent database A_τ of τ with Σ'; and (ii) checking $chase(A_\tau, \Sigma') \models \tau$. By Claim 1, we know that the size $chase(A_\tau, \Sigma')$ is polynomial in D. Then, by Claim 2, also the second phase takes polynomial time.

(b) The core of the antecedent of any ID or FD consists of a single atom with pairwise distinct variables. If target dependencies in Σ are restricted to IDs and FDs, satisfiability of the antecedent $R(\overline{z})$ of any of them can be tested by simply checking if an R atom occurs in the conclusion of some s-t tgd in $\hat{\Sigma}_{st}$. For the soundness of this test method, assume that such tgd σ exists. Then the source instance A_σ obtained by instantiating the variables of $ant(\sigma)$ with distinct constants is a witness that $R(\overline{z})$ is satisfiable in Σ. Indeed, both the antecedent (and thus, A_σ) and the conclusion of σ result from the same chase with Σ at step 1 of the PROPAGATE

procedure.

In the opposite direction, assume that there exists a source database I such that $chase(I, \Sigma) \models \exists \overline{z} R(\overline{z})$. Let $\sigma_0, \ldots, \sigma_k$ be a smallest sequence of tgds that need to fire in order to add an R fact to $chase(I, \Sigma)$: σ_0 is a s-t tgd in Σ, and σ_{i+1} is a target ID which is not applicable unless σ_i has fired. Let σ be the tgd in $\hat{\Sigma}_{st}$ produced by the PROPAGATE procedure applied to $ant(\sigma_0)$ and Σ. We know that σ exists, that is, the chase of $At(ant(\sigma_0))$ with Σ does not fail due to an attempt to unify distinct constants from $ant(\sigma_0)$ resp. Σ: otherwise, it would also fail on I. The conclusion of σ includes all target relational symbols occurring in σ_0 and hence any chase in which σ fires, also contains chase steps with $\sigma_1, \ldots, \sigma_k$. In particular, this holds for the chase sequence that produced the conclusion of σ, and thus at least one R atom must occur there. We have shown the completeness of our satisfiability check. It is feasible in time $O(D)$. □

The previous result extends to mapping optimization with respect to redundancy elimination and subset-minimality. Namely, the following theorem can be easily shown:

Theorem 29. *Let $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$ be a schema mapping with target FDs and weakly acyclic set of IDs. Moreover, assume that the number of terms in the s-t tgds of Σ is limited by some constant b. Then, the following problems can be decided in polynomial time:*

- *checking if a dependency $\tau \in \Sigma$ is redundant w.r.t. DE-equivalence, and therefore,*
- *checking if Σ is subset-minimal.*

Thus, in terms of complexity, DE-equivalence, while offering strictly higher optimization potential, is no worse than logical equivalence (studied in [31]) in the setting of Theorem 28. Note also, that the Theorems 28 and 29 can be adapted for a class of mappings with more expressible target dependencies than FDs and IDs: namely, constant-free and connected target tgds and egds. In this case we assume that also s-t tgds are free of constants. Then, the following satisfiability test using a notion called *critical instance* in [43] can be used:

Definition 26. *A* critical instance $I^1_{\mathbf{S}}$ *for schema* $\mathbf{S}$ *contains a single atom $R(c, \ldots c)$ for each relation R in* $\mathbf{S}$*, where c is some constant: $dom(I^1_{\mathbf{S}}) = \{c\}$.*

Theorem 30. *Let $\mathcal{M} = \langle \mathbf{S}, \mathbf{T}, \Sigma \rangle$ be schema mapping where Σ consists of constant-free dependencies: source egds, s-t tgds, and target egds and tgds, possessing the terminating chase property. A Boolean conjunctive query q is satisfiable in Σ iff $chase(I^1_{\mathbf{S}}, \Sigma) \models q$.*

Proof. There is a homomorphism h (not preserving constants) from any source database I onto $I^1_{\mathbf{S}}$. Thus, any dependency with constant-free antecedent which fires in the chase of I is also applicable when $I^1_{\mathbf{S}}$ is chased, and h can be extended to h': $chase(I, \Sigma) \rightarrow chase(I^1_{\mathbf{S}}, \Sigma)$ by extending the domain of h to the nulls introduced by the chase of I with Σ. Hence, for every I, $chase(I, \Sigma) \models q$ only if $chase(I^1_{\mathbf{S}}, \Sigma) \models q$. The "if" direction is trivial. □

Theorem 28 can now be reformulated as follows:

Theorem 31. *DE-equivalence of schema mappings with constant-free and connected dependencies, possessing the terminating chase property, is decidable.*

Furthermore, suppose that the number of terms in each dependency in the mapping is bounded by some constant b, and the set of target tgds is weakly-acyclic. Then DE-equivalence of such mappings can be decided in polynomial time.

Proof. The same as in Theorem 28, except for the running time estimation being based on the bound b also in case of target dependencies, and the satisfiability test (needed to check the AUC in Step 2 of the decision procedure) based on Theorem 30. Claim 1 in the proof of Theorem 28 and polynomial data complexity of CQs justify that also in this case the satisfiability test is feasible in $poly(|\Sigma|)$ time. □

For mappings with target dependencies of arbitrary size, further useful minimization tasks may be investigated: for example, testing if any atom can be eliminated from a given dependency. The decidability of such tasks, concerned with redundancy and subset-minimality, can be seen as a direct consequence of Theorem 31. There is no immediate solution for cardinality-minimality, however, since this problem does not offer a bounded search space.

CHAPTER 8

CQ-equivalence of SO-tgds

8.1 Background on logical equivalence of SO-tgds

In this chapter we are dealing with a more expressive language than that of s-t tgds, namely with the language of SO-tgds. Recall that the two main features distinguish SO-tgds from s-t tgds: (1) arbitrary Skolem functions instead of existential variables and (2) equalities between terms in the antecedents. These features are crucial for the complexity of equivalence testing: logical equivalence, decidable by a simple syntactic characterization based on Lemma 1 in the case of s-t tgds, becomes undecidable for SO-tgds. It follows as a corollary of the undecidability result by Arenas et al. in the context of inverse schema mappings [6].

Theorem 32. *Let τ and τ' be two SO tgds. It is undecidable whether $\tau \equiv \tau'$.*

Proof (Sketch).[1] In [20], a schema mapping $\mathcal{M}_2$ is defined to be an inverse of another mapping $\mathcal{M}_1$ if their composition $\mathcal{M}_1 \circ \mathcal{M}_2$ is logically equivalent to the identity mapping, i.e., a set of s-t tgds of the form $(\forall \overline{x})\big(P(\overline{x}) \rightarrow \widehat{P}(\overline{x})\big)$ for every source relation P. In [6, Corollary 9.5] the authors show that the following problem is undecidable: Given mappings $\mathcal{M}_1$ and $\mathcal{M}_2$ specified by s-t tgds, check whether $\mathcal{M}_2$ is an inverse of $\mathcal{M}_1$.

In other words, checking if the composition $\mathcal{M}_1 \circ \mathcal{M}_2$ is equivalent to the identity mapping is undecidable. Clearly, a set of s-t tgds can be represented as an SO tgd. Let τ denote the SO tgd corresponding to the set of s-t tgds of the identity mapping. Likewise, $\mathcal{M}_1 \circ \mathcal{M}_2$ can be represented as an SO tgd, say τ'. Then $\tau \equiv \tau'$ holds iff $\mathcal{M}_2$ is an inverse of $\mathcal{M}_1$. Hence, the logical equivalence of two SO tgds is undecidable. □

In [27], also a significantly stronger result has been shown using a reduction from the Halting Problem for Turing Machines:

[1] We thank the anonymous referee of AMW 2011 who pointed out this proof.

Theorem 33 ([27]). *Let τ and τ' be two SO tgds. It is undecidable whether $\tau \equiv \tau'$, even when $\tau \equiv_{CQ} \tau'$ is known to hold.*

That is, even if the two SO-tgds are known to be CQ-equivalent, it is undecidable if they are logically equivalent. But is it possible to find out if two SO-tgds are CQ-equivalent? We will make a first step towards answering this question in the next section.

8.2 CQ-Equivalence

We now show the undecidability of CQ-equivalence of schema mappings based on SO-tgds *and source key dependencies*, by a reduction from the Domino Problem (DP), which was first shown undecidable by Berger [8].

Definition 27. *A* domino system *(DS) is given by a set $\mathcal{D}$ of* domino types *defined as follows. Let $\mathcal{C}$ be a set of colours. Then, each domino type is a quadruple $\langle l, t, r, b\rangle$ of values from $\mathcal{C}$, corresponding respectively to the left, top, right and bottom colour of a square domino piece. Let $d.side$ (with $side \in \{left, right, bottom, top\}$) denote an accessor function $\mathcal{D} \to \mathcal{C}$ returning the colour of d, corresponding to $side$. Then, the* domino problem *(DP) given by $\mathcal{D}$ asks if for each $m, n \in \mathbb{N}$ a* consistent tiling *of an $m \times n$ grid with the domino types from $\mathcal{D}$ exists. That is, if there is a function $t_{m,n} \colon \{1 \dots m\} \times \{1 \dots n\} \to \mathcal{D}$, such that the equality $t_{m,n}(i,j).right = t_{m,n}(i+1,j).left$ holds for $1 \leq i < m$ and $1 \leq j \leq n$ and $t_{m,n}(k,l).bottom = t_{m,n}(k,l+1).top$ for $1 \leq k \leq m$ and $1 \leq l < n$.*

Note that the conventional formulation of the Domino Problem [8] asks if there exists a consistent tiling of an infinite plane, while Definition 27 seeks for tilings of arbitrarily big rectangles. There is no contradiction: in fact, the proof of undecidability specifies an encoding of a the blank-tape Turing Machine $\mathcal{M}$ as a DS $\mathcal{D}_\mathcal{M}$, such that $\mathcal{M}$ does not halt iff there is a tiling of an arbitrarily big square grid with $\mathcal{D}_\mathcal{M}$ ([2, p.414]).

Chains, proper instances, and coordinates. We model a grid using the source schema $\mathbf{S}$ consisting of two binary relations C_h and C_v intended to define a horizontal and vertical successor relation: For an instance I of $\mathbf{S}$, C_h^I and C_v^I denote the relations of I. Often in this chapter, our statements will apply to both relations C_h and C_v alike. To avoid repetitions, we will then use the symbol $C_{h|v}$ resp. $C_{h|v}^I$ as a placeholder for any of C_h, C_v resp. C_h^I, C_v^I. The intended form of $C_{h|v}^I$ is $\{(a_0, a_1), (a_1, a_2) \dots (a_{n-1}, a_n)\}$, with $a_i \neq a_j$, for every $0 \leq i < j \leq n$. We define $a < b$ w.r.t. an order $C_{h|v}^I$, if $(a,b) \in C_{h|v}^I$ or $\exists x, C_{h|v}(x,b) \wedge a < x$. Taking the tuples of $C_{h|v}^I$ as edges of a directed graph (which we call the *dual graph*), the order defined above corresponds to an *acyclic chain*. If each relation of an instance I over $\mathbf{S}$ contains a single acyclic chain, I is called *proper*. The size (m,n) of such a proper instance I is defined as $(|C_h|, |C_v|)$.

Domino tilings are modelled by the target relation D of arity 4. Each of its four arguments is populated by a functional term $c^s(x,y)$ where c corresponds to the colour, s to the side of the domino piece, and the pair (x,y) defines the position in the grid. To save notation, we will

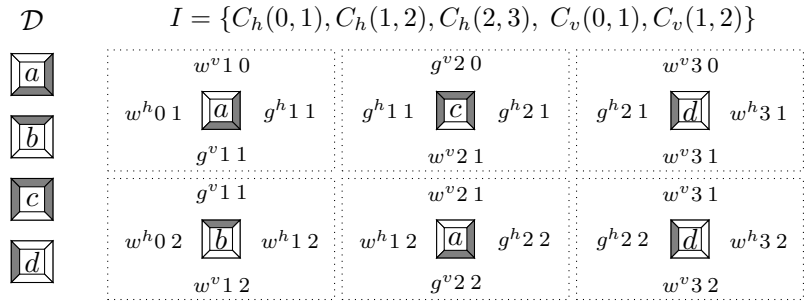

Figure 8.1: Dotted rectangles are facts in $core(I, \Sigma_{\mathcal{D}})$, corresponding to a consistent tiling

omit parentheses when writing the terms in the relation D (e.g. the above term will be written as $c^s xy$). Given a fact $d \in D$, we define the *accessor function* $d[i]$, returning the term in the i-th place of d with $1 \leq i \leq 4$. Recall that the sequence of sides in the D-tuple is as follows: $\langle \mathit{left}, \mathit{top}, \mathit{right}, \mathit{bottom} \rangle$.

Definition 28. *Let $\mathcal{D}$ be a DS with types $d_1 \dots d_k$. A simulation mapping $\Sigma_{\mathcal{D}}$ for $\mathcal{D}$ over the source schema $\mathbf{S} = \{C_h(\cdot,\cdot), C_v(\cdot,\cdot)\}$ and the target schema $\mathbf{T} = \{D(\cdot,\cdot,\cdot,\cdot)\}$ consists of an SO tgd $\tau_{\mathcal{D}}$ and two source key dependencies:*

$$\epsilon_1 \colon \quad C_h(x, x_1) \wedge C_h(x, x_2) \to x_1 = x_2,$$

$$\epsilon_2 \colon \quad C_v(x, x_1) \wedge C_v(x, x_2) \to x_1 = x_2.$$

The SO tgd $\tau_{\mathcal{D}}$ has the form $\exists \mathsf{count}\, \exists p^h \exists p^v \exists q^h \exists q^v (\tau_{d_1} \wedge \ldots \wedge \tau_{d_k} \wedge \gamma_h \wedge \gamma_v)$, where $\tau_{d_1} \dots \tau_{d_k}$ encode the corresponding domino types:

$$\tau_{d_i} \colon \quad C_h(x_1, x_2) \wedge C_v(y_1, y_2) \to D(l^h\, x_1\, y_2, t^v\, x_2\, y_1, r^h\, x_2\, y_2, b^v\, x_2\, y_2),$$

l, t, r, b denoting respectively the left, top, right and bottom colours of d_i. Superscripts distinguish the horizontal and vertical dimensions, so that there are two distinct functions in count *for each colour in $\mathcal{D}$. Moreover,* count *contains neither $p^{h|v}$ nor $q^{h|v}$.*

The conjuncts γ_h and γ_v are called guards *and have a form similar to τ_{d_i}:*

$$\gamma_h \colon \quad C_h(x_0, x_1) \wedge C_h(x_1, x_2) \wedge C_v(y_1, y_2) \to D(p^h\, x_1\, y_2, p^v\, x_2\, y_1, p^h\, x_2\, y_2, p^v\, x_2\, y_2)$$

$$\gamma_v \colon \quad C_h(x_1, x_2) \wedge C_v(y_0, y_1) \wedge C_v(y_1, y_2) \to D(q^h\, x_1\, y_2, q^v\, x_2\, y_1, q^h\, x_2\, y_2, q^v\, x_2\, y_2)$$

Informally, given a proper source instance of size (m, n), the conjuncts $\tau_{d_1} \dots \tau_{d_{|\mathcal{D}|}}$ create a stack of $|\mathcal{D}|$ domino tiles at each position in the $m \times n$ grid. Additionally, the guards γ_h and γ_v create two more tiles with unique colours p and q on each position (i, j) with $i, j > 1$, or a single tile if $i = 1$ or $j = 1$. The guards tackle source instances with cycles instead of the linear order, as will be explained later.

For a source instance I, a D-fact d enforced by the SO tgd in a simulation mapping (that is, $d \in chase(I, \Sigma_{\mathcal{D}})$) has a form $D(l^h x_p\, y, t^v xy_p, r^h xy, b^v xy)$, s.t. $(x_p, x) \in C_h^I$, $(y_p, y) \in C_v^I$.

Definition 29. *Let $\mathcal{D}$ be a DS and I a proper instance of size (m, n). A fact $d = D(l^h x_p y, t^v x y_p, r^h x y, b^v x y) \in chase(I, \Sigma_\mathcal{D})$ is associated with the domino type $\langle l, t, r, b\rangle$, and with the pair (x, y) of I-values, occurring as arguments in its two last terms. Moreover, let i be the ordinal of x w.r.t. the order defined by C_h^I and j be the ordinal of y w.r.t. C_v^I. We call the pair (i, j) the* coordinates *of d defining its* grid position.

Proposition 4. *Let I be a proper source instance, and $\Sigma_\mathcal{D}$ the simulation mapping of a DS $\mathcal{D}$. Moreover, let d_1 and d_2 be two facts in $chase(I, \tau_\mathcal{D})$, d_1 associated with the pair of I-values (x_1, y_1), and d_2 with (x_2, y_2). Then, the following claims hold true:*

1. $d_1[3] = d_2[1] \Rightarrow (x_1, x_2) \in C_h^I \wedge y_1 = y_2 \wedge \forall i \neq 3\ \forall j \neq 1\ d_1[i] \neq d_2[j]$.
2. $d_1[2] = d_2[4] \Rightarrow (y_1, y_2) \in C_v^I \wedge x_1 = x_2 \wedge \forall i \neq 4\ \forall j \neq 2\ d_1[i] \neq d_2[j]$.
3. $\exists i \in \{1...4\}\ d_1[i] = d_2[i] \Rightarrow x_1 = x_2 \wedge y_1 = y_2$.
4. $d_1[1] \neq d_2[2] \wedge d_1[3] \neq d_2[4] \wedge d_1[1] \neq d_2[4]$.

Claim 1 stipulates, that whenever a term is shared between the 3^{rd} place of d_1 and the 1^{st} place of d_2, the facts correspond to horizontally adjacent grid positions (i.e., have first coordinates i and $i + 1$ and the same second coordinate) and can have no further terms in common. Claim 2 is an analogue of Claim 1 for the equality $d_1[4] = d_2[2]$ and vertical adjacency: d_1 and d_2 must have coordinates (i, j) and $(i, j + 1)$. Claim 3 states that terms at respective positions can be only shared by facts with the same coordinates. Finally, Claim 4 restricts term equalities to the above cases.

Proof (Sketch) of Proposition 4. All four claims follow directly from Definition 28 and the fact that the source instance I is proper. Note that the domain of $chase(I, \tau_\mathcal{D})$ consists of labelled nulls associated with binary Skolem functional symbols. The equality $d_1[i] = d_2[j]$ means that the leading symbols and corresponding arguments of the Skolem terms coincide.
Claim 1. Suppose that $d_1[3] = d_2[1]$ holds. The facts d_1, d_2 have been instantiated by the chase, and therefore, by Definition 28, they must have the form $D(c^h x_0 y_1, {}^v_2 x_1 y_0, c_3^h x_1 y_1, c_4^v x_1 y_1)$ and $D(c_3^h x_1 y_1, c_5^v x_2 y_0', c_6^h x_2 y_1, c_7^v x_2 y_1)$, respectively. Moreover, the following inclusions hold: $(x_0, x_1), (x_1, x_2) \in C_h^I$ and $(y_0, y_1), (y_0', y_1), (y_1, y_2) \in C_v^I$. Since I is proper, $y_0' = y_0$ must be the case, otherwise C_v^I is not a chain. By the same reason, the inequalities $x_0 \neq x_1$, $x_1 \neq x_2$, $x_0 \neq x_2$, and $y_0 \neq y_1$ hold, which proves the claim.

Claim 2 can be shown analogously to Claim 1. *Claim 3* follows from the observation that the arguments of each Skolem term from $dom(chase(I, \Sigma_\mathcal{D}))$ uniquely define the coordinates of the corresponding domino piece. Finally, *Claim 4* is due to the superscripts ^{h} and ^{v} appended to the leading symbols of Skolem terms. □

Example 24. *Consider a DS $\mathcal{D}$ containing the four types a, b, c, d in Figure 8.1, using only two colours white and grey denoted as w and g, respectively. In the simulation mapping $\Sigma_\mathcal{D}$, the*

first type is modelled by the following implicational conjunct in the SO tgd τ_a: $C_h(x_1, x_2) \wedge C_v(y_1, y_2) \rightarrow D(w^h\, x_1\, y_2, w^v\, x_2\, y_1, g^h\, x_2\, y_2, g^v\, x_2\, y_2)$.

The 3×2 *grid is represented by a proper source instance* I *(see top of Figure 8.1). The six dotted rectangles represent a consistent tiling of this grid with* $\mathcal{D}$*. Each rectangle bears the four terms of the corresponding* D*-fact in* $chase(I, \Sigma_\mathcal{D})$*: e.g., the topmost leftmost tile has the position (1,1) and is encoded by the fact* $D(w^h01, w^v10, g^h11, g^v11)$.

Problem reduction. Let $\mathcal{D}$ be a DS and $\Sigma_\mathcal{D}$ its simulation mapping. Furthermore, consider a singleton colour set $\mathcal{C}_b = \{b\}$ (where b stands for "black") and a corresponding domino set $\mathcal{B} = \{\langle b, b, b, b\rangle\}$ with the simulation mapping $\Sigma_\mathcal{B}$. We claim that $\mathcal{D}$ has a solution iff $\Sigma_\mathcal{D} \equiv_{CQ} \Sigma_\mathcal{B}$ holds. The SO tgd in $\Sigma_\mathcal{B}$ has the following form:

$$\tau_\mathcal{B}\colon\ \exists b^h \exists b^v \exists p^h \exists p^v \exists q^h \exists q^v \\ \big(\big(C_h(x_1, x_2) \wedge C_v(y_1, y_2) \rightarrow D(b^h\, x_1\, y_2, b^v\, x_2\, y_1, b^h\, x_2\, y_2, b^v\, x_2\, y_2)\big) \wedge \gamma_h \wedge \gamma_v\big).$$

It is easy to see that the conjuncts γ_h and γ_v are redundant in $\tau_\mathcal{B}$. Hence, in the following we assume $\tau_\mathcal{B}$ to contain a single implicational conjunct and no guards. Moreover, the following relationship holds between the chase result of $\tau_\mathcal{B}$ and $\tau_\mathcal{D}$:

Proposition 5. *Let* $\mathcal{D}$ *be a DS with the simulation mapping* $\Sigma_\mathcal{D}$*, let* (i_1, j_1) *and* (i_2, j_2) *be two distinct grid positions, and let* $d_1, d_2 \in chase(I, \Sigma_\mathcal{D})$ *be facts at positions* (i_1, j_1) *resp.* (i_2, j_2)*. Then also* $chase(I, \Sigma_\mathcal{B})$ *contains two facts* d'_1, d'_2 *at these grid positions. Moreover, for every* $1 \leq k, l \leq 4$*, if* $d_1[k] = d_2[l]$ *then* $d'_1[k] = d'_2[l]$*, i.e., if two terms in* d_1 *and* d_2 *are equal, then the terms at these places in* d'_1 *and* d'_2 *are equal as well.*

Proof (Sketch). This proposition uses the observation that every implicational conjunct τ_{d_i} encoding a domino type d_i in the SO tgd of a simulation mapping produces exactly one D-fact for each grid position, and that $\tau_\mathcal{B} \in \Sigma_\mathcal{B}$ uses the minimal set of functional symbols. □

For DPs that have a solution (of which $\mathcal{B}$ is obviously an example) the following lemma, which is crucial for our construction, holds, as will be shown shortly:

Lemma 25. *Let* I *be a proper instance of size* (m, n) *and let* $\mathcal{D}$ *be a DS. Then, there is a consistent tiling of an* $m \times n$ *grid with* $\mathcal{D}$ *iff* $core(I, \tau_\mathcal{D}) \cong core(I, \Sigma_\mathcal{B})$ *holds.*

Example 25. *It is easy to check that the six* D*-facts in Figure 8.1 indeed form* $core(I, \Sigma_\mathcal{D})$*. For* I*,* $\Sigma_\mathcal{B}$ *yields a solution with the isomorphic core: each fact in* $core(I, \Sigma_\mathcal{B})$ *can be obtained from some fact of* $core(I, \Sigma_\mathcal{D})$ *by replacing leading symbols* w *and* g *with* b*.*

For the proof of Lemma 25 and subsequent results, we recall the notion of connectedness of an instance, see Definition 25.

Proof of Lemma 25. First we recall our convention, that $\Sigma_\mathcal{B}$ contains an SO tgd $\tau_\mathcal{B}$ with a single implicational conjunct

$$C_h(x_1, x_2) \wedge C_v(y_1, y_2) \rightarrow D(b^h\, x_1\, y_2, b^v\, x_2\, y_1, b^h\, x_2\, y_2, b^v\, x_2\, y_2).$$

Hence, for a proper source instance I, $core(I, \Sigma_{\mathcal{B}}) = chase(I, \Sigma_{\mathcal{B}})$.

[$\Rightarrow$] Assume that $\mathcal{D}$ consistently tiles an $m \times n$ grid. We first show that some subinstance $J_{\mathcal{B}}$ of $J = chase(I, \Sigma_{\mathcal{D}})$ is isomorphic to $chase(I, \Sigma_{\mathcal{B}})$, and then that $J \to J_{\mathcal{B}}$ holds.

Consider the SO tgd $\tau_{\mathcal{D}} \in \Sigma_{\mathcal{D}}$. For each domino type $\delta = \langle l, t, r, b\rangle$ in $\mathcal{D}$, there is an implicational conjunct τ_δ of $\tau_{\mathcal{D}}$ of the form

$$C_h(x_1, x_2) \wedge C_v(y_1, y_2) \to D(l^h x_1 y_2, t^v x_2 y_1, r^h x_2 y_2, b^v x_2 y_2).$$

That is, τ_δ coincides with $\tau_{\mathcal{B}}$ up to names of functional symbols in the conclusion. Hence, both $chase(I, \Sigma_{\mathcal{D}})$ and $chase(I, \Sigma_{\mathcal{B}})$ can be partitioned in $m \cdot n$ non-empty subinstances associated with the pairs $(x, y) \in dom(C_h^I) \times dom(C_v^I)$. In $chase(I, \Sigma_{\mathcal{B}})$, there is exactly one D-fact per each grid position, whereas in $chase(I, \Sigma_{\mathcal{D}})$ there are at least $|\mathcal{D}|$ D-facts (plus one or two D-facts produced by the guards γ_h and γ_v, depending on the coordinates of x and y).

We want to show that, if there is a consistent tiling of an $m \times n$ grid with $\mathcal{D}$, a subinstance $J_{\mathcal{B}}$ isomorphic to $chase(I, \Sigma_{\mathcal{B}})$ can be built by unifying, for each $(x, y) \in dom(C_h^I) \times dom(C_v^I)$, the fact $chase(I, \Sigma_{\mathcal{B}})$ associated with (x, y) with some fact of J, associated with the same pair of I-values (x, y).

In the rest of the proof, we use the following notation: for an I-value x_i, the subscript denotes the coordinate of x. The fact associated with a pair of I-values (x_i, y_j) has thus the grid position (i, j). We will add a subscript $_{i,j}$ to variables denoting facts to specify the fact's grid positions.

Let $d_{i,j} = D(b^h x_{i-1} y_j, b^v x_i y_{j-1}, b^h x_i y_j, b^v x_i y_j)$ be a fact in $chase(I, \Sigma_{\mathcal{B}})$ associated with the grid position (i, j). It has a single term in common with at most four other atoms in $chase(I, \Sigma_{\mathcal{B}})$:

1. If $i > 0$, there is a fact $d_{i-1,j} \in chase(I, \Sigma_{\mathcal{B}})$ such that $d_{i-1,j}[3] = d_{i,j}[1]$ holds. The shared term is $b^h x_{i-1} y_j$.
2. If $j > 0$, there is a fact $d_{i,j-1} \in chase(I, \Sigma_{\mathcal{B}})$ such that $d_{i,j-1}[4] = d_{i,j}[2]$ holds. The shared term is $b^v x_i y_{j-1}$.
3. If $i < m$, there is a fact $d_{i+1,j} \in chase(I, \Sigma_{\mathcal{B}})$ such that $d_{i,j}[3] = d_{i+1,j}[1]$ holds. The shared term is $b^h x_i y_j$.
4. If $j < n$, there is a fact $d_{i,j+1} \in chase(I, \Sigma_{\mathcal{B}})$ such that $d_{i,j}[4] = d_{i,j+1}[2]$ holds. The shared term is $b^v x_i y_j$.

Now, let $t_{m,n}$ be a function $\{1 \dots m\} \times \{1 \dots n\} \to \mathcal{D}$ corresponding to a consistent tiling of an $m \times n$ grid with $\mathcal{D}$. We define a function $f\colon \{1 \dots m\} \times \{1 \dots n\} \to J$, s.t. whenever $t_{m,n}(i, j) = \langle c_l, c_t, c_r, c_b\rangle$, where $\langle c_l, c_t, c_r, c_b\rangle \in \mathcal{D}$, we set

$f(i, j) = D(c_l^h x_{i-1} y_j, c_t^v x_i y_{j-1}, c_r^h x_i y_j, c_b^v x_i y_j) \in J$,

where x_k denotes k^{th} value from the order C_h^I, and y_l denotes the l^{th} value from C_v^I. Such a function f is well-defined, since, by construction of $\Sigma_{\mathcal{D}}$, for each position (i, j) of an $m \times n$

grid, there is a pair (x_i, y_j) of I-values in $dom(C_h^I) \times dom(C_v^I)$, such that $f(i,j) \in J$ holds. Moreover, since $t_{m,n}$ corresponds to a consistent tiling, the following term equalities hold, in full analogy to $chase(I, \Sigma_{\mathcal{B}})$:

1. $f(i-1,j)[3] = f(i,j)[1]$, for all $i > 0$
2. $f(i,j-1)[4] = f(i,j)[2]$, for all $j > 0$
3. $f(i,j)[3] = f(i+1,j)[1]$, for all $i < m$
4. $f(i,j)[4] = f(i,j+1)[2]$, for all $j < n$

Indeed, the arguments of the corresponding terms are the same in $chase(I, \Sigma_{\mathcal{D}})$ and $chase(I, \Sigma_{\mathcal{B}})$ and the leading symbols of functional terms coincide by the tiling consistency conditions. Hence, there is an isomorphism between the instances $J_{\mathcal{B}} = \{f(i,j) \mid 1 \leq i \leq m, 1 \leq j \leq n\} \subset J$ and $chase(I, \Sigma_{\mathcal{B}})$.

It remains to show that an endomorphism $J \to J_{\mathcal{B}}$ exists. $J_{\mathcal{B}}$ contains exactly a single fact for every grid position, corresponding to the domino type from the grid tiling. In J, there are more D-facts with the same grid positions: namely, those corresponding to other domino types and those produced by the guards γ_h and γ_v. By Proposition 4 the only equalities between facts at distinct positions in J are those listed under the items (1)–(4) above. We have already shown that these equalities are present in $J_{\mathcal{B}}$. Therefore, we can conclude, that a mapping sending any fact $\hat{d}_{i,j} \in J$ onto $d_{i,j} \in J_{\mathcal{B}}$ is a homomorphism. Hence, $J_{\mathcal{B}}$ is indeed a core of J.

[$\Leftarrow$] Assume there exists an isomorphism e between the $core(I, \Sigma_{\mathcal{B}}) = chase(I, \Sigma_{\mathcal{B}})$ and $core(I, \Sigma_{\mathcal{D}})$. We show that there exists a consistent tiling of an $m \times n$ grid. To do so, we prove two claims:

1. Every isomorphism e' between $core(I, \Sigma_{\mathcal{D}})$ and any subinstance of $chase(I, \Sigma_{\mathcal{B}})$ must relate the facts with the same coordinates. In other words, if the fact $D(\alpha_l, \alpha_t, \alpha_r, \alpha_b)$ is in $core(I, \Sigma_{\mathcal{D}})$ and the fact $D(e'(\alpha_l), e'(\alpha_t), e'(\alpha_r), e'(\alpha_b))$ in $chase(I, \Sigma_{\mathcal{B}})$, then the Skolem terms denoted by α_i and $e'(\alpha_i)$ must have the same pairs of arguments, for each $i \in \{l, t, r, b\}$.
2. There is no isomorphism $h\colon chase(I, \Sigma_{\mathcal{B}}) \to chase(I, \Sigma_{\mathcal{D}})$ such that a fact d with functional terms $p^{h|v}xy$ or $q^{h|v}xy$ (that is, enforced by the guards γ_h or γ_v) is a member of $h(chase(I, \Sigma_{\mathcal{B}}))$.

From these two claims, it follows that the tiling of a grid can be "read off" from $core(I, \Sigma_{\mathcal{D}})$, since in $chase(I, \Sigma_{\mathcal{B}})$ and, hence, in $core(I, \Sigma_{\mathcal{D}})$ there is exactly a single D-fact for every grid position.

Proof of Claim 1. Assume, e relates a fact $d \in core(I, \Sigma_{\mathcal{D}})$ to some fact $d' \in core(I, \Sigma_{\mathcal{B}})$, so that the coordinates (i,j) of d and (i',j') of d' are distinct. W.l.o.g., assume that $i' > i$ holds, and let k denote the difference $i' - i$. Let $d_{0,j} \in core(\Sigma_{\mathcal{D}})$ be the fact with the coordinates $(0,j)$.

That is, the last two Skolem terms in $d_{0,j}$ have the arguments (x_0, y), such that x_0 is the minimal element w.r.t. the linear order defined by C_h^I. Since e is an isomorphism and $i' - i = k$, the image of $d_{0,j}$ in $core(I, \Sigma_\mathcal{B})$ is the fact $d'_{k,j'}$. By Claim (1) of Proposition 4, there is no fact $d_{-1,j}$ in $chase(I, \Sigma_\mathcal{D})$, such that the equality $d_{-1,j}[3] = d_{0,j}[1]$ holds: $(x, x_0) \notin C_h^I$ holds for all x. At the same time, there is an atom $d_{k-1,j'} \in core(I, \tau_\mathcal{B})$ such that equality $d_{k-1,j'}[3] = d_{k,j'}[1]$ holds, which contradicts the assumption that e is an isomorphism.

Proof of Claim 2. Note that applied to a proper instance of size (m, n), the maximal horizontal coordinate of a D-fact produced by γ_h is $m - 1$, and the maximal vertical coordinate of a D-fact produced by γ_v is $n - 1$. Hence, by exactly the same reasoning as in the proof of Claim 1, we can show that no isomorphism can map $chase(I, \Sigma_\mathcal{B})$ on instances produced by the guards. Moreover, $chase(I, \Sigma_\mathcal{B})$ is connected, while the guards produce D-facts which cannot be connected to any fact encoding a domino piece (conjuncts $\tau_1, \ldots \tau_{|\mathcal{D}|}$ and $\gamma_{h|v}$ in Definition 28 use distinct functional symbols). Hence, by a trivial generalization of the Lemma 24 we can show that no homomorphism can map part of the nulls of $chase(I, \Sigma_\mathcal{B})$ to the nulls produced by the conjuncts $\tau_1, \ldots \tau_{|\mathcal{D}|}$ and the other part to the nulls produced by the guards $\gamma_{h|v}$. □

To lift the equivalence between solvability of $\mathcal{D}$ and existence of an isomorphism $core(I, \Sigma_\mathcal{D}) \cong core(I, \Sigma_\mathcal{B})$ to unrestricted source instances I, we will first show, that whenever C_h^I or C_v^I contains a single simple cycle (that is, its dual graph is connected, each vertex having a single incoming and a single outgoing edge), then any simulation mapping yields an instance with the same core as $chase(I, \Sigma_\mathcal{B})$.

Lemma 26. *Let I be of such a form that either C_h^I or C_v^I consists of a single cycle. Then, for arbitrary DS $\mathcal{D}$, $core(I, \Sigma_\mathcal{D}) \cong core(I, \Sigma_\mathcal{B})$ holds.*

Proof (Sketch). W.l.o.g. assume that the single cycle is contained in C_h. For such I, the guard γ_h in the SO tgd $\tau_\mathcal{D} \in \Sigma_\mathcal{D}$ is indistinguishable from $\tau_\mathcal{B}$ in $\Sigma_\mathcal{B}$. Indeed, the conclusions of these implications coincide up to renaming of functional terms. The antecedent φ_h of γ_h has an extra atom $C(x_0, x_1)$, that prevents it from being satisfied by a C_h-fact at the beginning of the chain. However, if C_h^I is a cycle, the relations $q_{1|2}$ defined as $q_1(x_1, x_2) \leftrightarrow C_h(x_1, x_2)$ and $q_2(x_1, x_2) \leftrightarrow \exists x_0\ C_h(x_0, x_1) \wedge C_h(x_1, x_2)$, coincide, and so do $\tau_\mathcal{B}$ and γ_h. Hence, $J = chase(I, \Sigma_\mathcal{D})$ contains a subinstance $J_\mathcal{B}$ isomorphic to $chase(I, \Sigma_\mathcal{B})$. Using Proposition 5, $J \to J_\mathcal{B}$ can be shown. □

In the next lemma, we consider homomorphisms that do not preserve constants, and thus can map a source instance on another one. We call such homomorphisms *constants-agnostic*. It is immediate to see, that conjunctive queries which do not contain constants are closed under such homomorphisms: That is, let I be an instance and h a constants-agnostic homomorphism for I. If $q(\overline{x})$ is a CQ without constants, and a tuple $\bar{a}$ is such that $I \models q(\bar{a})$ then also $h(I) \models q(h(a))$. Note that according to Definition 28, all antecedents of implications in a simulation mapping are constants-free CQs. Therefore, we can show the following lemma.

Lemma 27. *Let I be an instance of* $\mathbf{S}$ *and $I' \subseteq I$ be an image of a constants-agnostic homomorphism h on I (that is, h does not necessarily preserve constants). Then, for the simulation mapping $\Sigma_\mathcal{D}$ of any DP $\mathcal{D}$, $core(I, \Sigma_\mathcal{D}) \cong core(I', \Sigma_\mathcal{D})$.*

Proof. Note that a simulation mapping does not require any values from the domain of a source instance to be transferred into the target instance. Let $D(l^h x_1 y_2, t^v x_2 y_1, r^h x_2 y_2, b^v x_2 y_2)$ be an arbitrary fact in $core(I, \Sigma_\mathcal{D})$. Since the antecedents of the implicational conjuncts in the SO tgd $\tau_\mathcal{D}$ of $\Sigma_\mathcal{D}$ are closed under constants-agnostic homomorphisms, we know that the fact $D(l^h \hat{x}_1 y_2, t^v \hat{x}_2 y_1, r^h \hat{x}_2 y_2, b^v \hat{x}_2 y_2)$ is in $chase(I', \Sigma_\mathcal{D}) \subseteq chase(I, \Sigma_\mathcal{D})$, where $\hat{x}$ abbreviates $h(x)$. Thus, we can easily construct a homomorphism from $core(I, \Sigma_\mathcal{D})$ to $chase(I', \Sigma_\mathcal{D})$, by mapping each labelled null αxy in the domain of $core(I, \Sigma_\mathcal{D})$ to $\alpha \hat{x} y \in dom(chase(I', \Sigma_\mathcal{D}))$. Composing this homomorphism with a homomorphism from $chase(I', \Sigma_\mathcal{D})$ to $core(I', \Sigma_\mathcal{D})$, we obtain a homomorphism $core(I, \Sigma_\mathcal{D}) \to core(I', \Sigma_\mathcal{D})$. Recall the property of cores mentioned in Chapter 2: $J_1 \leftrightarrow J_2$ iff $core(J_1) \cong core(J_2)$, for arbitrary instances J_1, J_2. The claim of the lemma then follows from an easy observation that a homomorphism $core(I', \Sigma_\mathcal{D}) \to core(I, \Sigma_\mathcal{D})$ can be obtained by restricting the homomorphism $chase(I, \Sigma_\mathcal{D}) \to core(I, \Sigma_\mathcal{D})$ to the domain of $core(I', \Sigma_\mathcal{D})$. □

Lemma 27 will be crucial for dealing with source instances that are neither acyclic chains nor simple cycles. The source key dependencies only deprecate the "fork" pattern in the $C_{h|v}$ relations, where distinct elements have a common predecessor. The "join" pattern — a common successor of the distinct elements — is allowed. We will next lift our simulation technique to instances containing the join pattern.

Definition 30. *An instance of the schema* $\mathbf{S}$ admissible *if it satisfies the key dependencies ϵ_1, ϵ_2 of Definition 28.*

The following lemma drastically restricts the diversity of admissible instances that should be considered. In particular, it implies that the join pattern is harmless for our simulation. Recall our convention of writing $C_{h|v}$ where the reasoning applies to the relations C_h and C_v alike.

Lemma 28. *Let I be an admissible instance. Then $core(I, \Sigma_\mathcal{D}) \cong core(I', \Sigma_\mathcal{D})$ holds, where instance $I' \subseteq I$ is obtained as follows: if $C^I_{h|v}$ contains a cycle, $C^{I'}_{h|v}$ consists of all simple cycles of $C^I_{h|v}$. Otherwise, $C^{I'}_{h|v}$ consists of one of the longest acyclic chains in $C^I_{h|v}$.*

Proof. The key dependencies $\epsilon_{1,2}$ enforce that each connected component in $C^I_{h|v}$ can contain at most one cycle. Indeed, let K be a connected component in $C^I_{h|v}$ and $K_c \subseteq K$ be a simple cycle in K. Let $f = C_{h|v}(z_1, z_2)$ be a fact in K. By l_f we denote a *distance* from f to K_c: a number of facts in a shortest chain in the dual graph of K, which contains any of z_1, z_2 together with some $y \in dom(K_c)$. Note that the chain must have the form $\big(C_{h|v}(z_1, z_2) \dots C_{h|v}(z_{l_f+1}, y)\big)$. A chain in the other direction, that is, $\big(C_{h|v}(y, z_{2-l_f}) \dots C_{h|v}(z_1, z_2)\big)$, is only possible if $l_f = 0$: Otherwise K will contain the distinct facts $C_{h|v}(y, z_{2-l_f})$ and $C_{h|v}(y, y') \in K_c$ which violates

the key dependencies, thus rendering K not admissible. By induction on l_f we show that f cannot belong to a simple cycle other than K_c.

Let $l_f = 0$ and assume that f belongs to two distinct simple cycles in K. Consider a chain in K_c starting in f and belonging also to the other cycle. There must be some distance $k < |K_c|$ from f, where the two cycles separate: the facts $C_{h|v}(z_{k+1}, z_{k+2}) \in K_c$ and $C_{h|v}(z_{k+1}, z'_{k+2}) \in K \setminus K_c$ must exist, otherwise, the cycles coincide. But then, K violates the source key dependencies and cannot be admissible. As induction hypothesis, assume that up to a certain distance n, no fact can belong to a cycle other than K_c. We consider a fact $f = C_{h|v}(z_1, z_2)$ with $l_f = n + 1$. Assume that f belongs to a simple cycle $K'_c = C_{h|v}(z_1, z_2), C_{h|v}(z_2, z_3), \ldots, C_{h|v}(z_m, z_1)$. Note that $K'_c \neq K_c$ since $l_f > 0$. By the induction hypothesis, no fact with $l_f \leq n$ can belong to K'_c and, since $l_f = n+1$, there must be a fact $f' = C(z_2, y) \in K$ with $l_{f'} = n$. But then K is not admissible, which is a contradiction. We have shown that every connected component in an admissible source instance has at most one cycle.

Next, we show that if a connected component K contains a simple cycle K_c, then there is a homomorphism (not preserving constants) $h : K \to K_c$ (That is, K_c can be seen as a core of K w.r.t. constants-agnostic homomorphisms). We start defining h by setting it equal to the identity on $dom(K_c)$. Next, we proceed recursively as follows:

1. Set K':=K_c.
2. For each fact $C_{h|v}(z_1, z_2) \in K$ such that $z_2 \in dom(K')$, pick a fact $f' = C_{h|v}(z'_1, z_2)$ from K'.
3. Extend h with the mapping $z_1 \to h(z'1)$, and set K':=$K' \cup \{C_{h|v}(z_1, z_2)\}$.
4. Repeat from step 1 until $K' = K$.

We now show the completeness and soundness of the above procedure. For completeness, note that all facts in K are visited, since K is connected and the procedure systematically explores all paths in K leading to K_c. There are no outbound paths starting in K_c (Otherwise, K is not admissible). The fact f' always exists at step 2, since K_c is a cycle. For the soundness, we have to show that the extension of h at step 3 always results in a homomorphism of K. To this end, we will show two loop invariants of the above procedure:

(i) *If the fact $C_{h|v}(z_1, z_2)$ at step 2 has $z_1 \notin dom(K')$, then for any predecessor z_1^p of z_1 w.r.t. the relation $C^I_{h|v}$, also $z_1^p \notin dom(K')$ holds.* After step 1 of the procedure, this loop invariant holds, since otherwise, there were more than a single cycle in K, which we have shown impossible. Suppose it also holds before the n^{th} iteration. Now suppose that, at step 2 entered at the n^{th} iteration of the loop, a term z_1 is selected whose predecessor z_1^p is in K'. W.l.o.g. we can assume that z_1^p is the closest such term to z_1: that is, there is no other term $z' \in K'$ such that a path from z_1^p to z' and from z' to z_1 exist. By construction of K', there is a path p from z_1^p to K_c via the terms from $dom(K')$. Since z_1 was picked at step 2, we know that there is another distinct path from z_1^p to K_c via z_1. Since z_1^p is the closest to z_1 term from $dom(K')$, there must

be the facts $C_{h|v}(z_1^p, z')$ and $C_{h|v}(z_1^p, z'')$ in K, such that $z' \in dom(K')$ and $z'' \notin dom(K')$ and thus, $z' \neq z''$. Therefore, I cannot be admissible.

(ii) *For each fact* $C_{h|v}(x_1, x_2) \in K'$, $C_{h|v}(h(x_1), h(x_2)) \in K_c$. This is certainly true before the first iteration of the loop: $K' = K_c$ after step 1, and h is the identity on $dom(K_c)$. Assume the invariant holds before the n^{th} iteration of the loop, we show that it is preserved after step 3. Extending the homomorphism h with the mapping $z_1 \to h(z_1')$, we send the fact $C_{h|v}(z_1, z_2)$ onto $f'' = C_{h|v}(h(z_1'), h(z_2))$. The fact f'' is an image of $f' = C_{h|v}(z_1', z_2)$ and $f' \in K'$ held when the loop entered its n^{th} iteration. Since we assumed that the loop invariant holds after the $(n-1)^{th}$ iteration, we have $C_{h|v}(h(z_1'), h(z_2)) \in K_c$, and then at step 3, K' is extended with a fact whose image under h is in K_c, as desired.

The two loop invariants allow us to show the third one, namely (iii) *h is a homomorphism* $K' \to K_c$. Indeed, this claim holds before the first iteration of the loop. Due to the invariant (i), we know that h covers exactly K' and that at least the first argument of every fact in $K \setminus K'$ does not belong to $dom(h)$. By the invariant (ii) we know that images for the values of $dom(K \setminus K')$ picked at step 2 always can be found, and the way these images are chosen at step 3 ensures that h is a homomorphism on K'. This concludes the proof of the soundness of our procedure.

Next, we argue that if a connected component K contains no cycles, then there is a homomorphism (not preserving constants) $h : K \to K_l$ where K_l is a chain of maximal length in K. Let's fix any such K_l in K and set h to be identity on $dom(K_l)$. We can be sure that the resulting procedure remains complete, since again, all paths leading to K_l are explored and no other paths exist (Otherwise, K_l would not be maximal). For the same reason, the fact f' at step 2 must exist. The loop invariants, including the last one "h is a homomorphism $K' \to K_c$", which proves the soundness of the procedure, can be shown as in the above case. □

Finally, we address the source instances I containing multiple connected components in $C^I_{h|v}$. The following property can be shown by induction on the distance (e.g., length of the shortest path, as in the proof of Lemma 28) between two D-facts:

Lemma 29. *Let $\Sigma_{\mathcal{D}}$ be a simulation mapping, I a source instance over* **S***, and let d_1, d_2 be facts in $J = chase(I, \Sigma_{\mathcal{D}})$ containing respectively terms $\alpha(x_1, y_1)$ and $\beta(x_2, y_2)$. Assume that x_1 is a term in a fact $f'_h \in C^I_h$, and x_2 is a term in $f''_h \in C^I_h$. Likewise, let y_1 and y_2 occur, respectively, in the facts $f'_v, f''_v \in C^I_v$. Then, d_1 and d_2 are connected in J only if the facts f'_h and f''_h are connected in C^I_h, and so are f'_v and f''_v in C^I_v.*

Proof. We proceed by induction on the distance l between d_1 and d_2. The following notation is used: given an instance I, x_{p_h} denotes the direct predecessor of x according to the relation C^I_h, and y_{p_v} denotes the direct predecessor of y in C^I_v.
$[l = 0]$, which corresponds to the case $d_1 = d_2$. For any fact of $chase(I, \Sigma_{\mathcal{D}})$ of the form $D(\alpha^h_1 x_{p_h} y, \alpha^v_2 x y_{p_v}, \alpha^h_3 x y, \alpha^v_4 x y)$, the terms x_{p_h}, x stem from some single fact in C^I_h, and the terms y_{p_v}, y from a fact in C^I_v.

$[l \rightarrow l+1]$ Let the facts $d_1, d_2 \in chase(I, \Sigma_{\mathcal{D}})$ be directly connected. We distinguish three cases, corresponding to Claims (1), (2) and (3) of Proposition 4. It follows from Proposition 4 that there are no other cases to be considered.

1. The equality $d_1[3] = d_2[1]$ holds. Let (x', y') and (x'', y'') be the pairs of I-values associated with d_1 resp. d_2 (e.g., pairs of arguments of terms at 3^{rd} and 4^{th} positions in d_1 resp. d_2). By Claim (1) of Proposition 4, $(x', x'') \in C_v^I$, $y' = y''$.

   Let $\alpha(x_1, y_1)$ be any term in d_1 and $\beta(x_2, y_2)$ be a term in d_2. For d_1, there are three possibilities: $(x_1, y_1) = (x', y')$, $(x_1, y_1) = (x'_{p_h}, y')$ or $(x_1, y_1) = (x', b'_{p_v})$. In all cases, we have that x_1, x' belong to the domain of the same connected component in C_h^I, and y_1, y' belong to the domain of the same connected component in C_v^I. Analogously, we show that x_2, x' belong to the domain of the same component of C_h^I, and y_2, y' to the domain of the same component of C_v^I. Hence, also x_1, x_2 belong to the domain of the same connected component in C_h^I, while y_1, y_2 belong to the domain of some component in C_v^I. The desired property follows.

2. The equality $d_1[4] = d_2[2]$ holds. The same reasoning as in the previous case applies, but using Claim (2) of Proposition 4 in place of Claim (1).

3. Respective terms in d_1 and d_2 have the same pairs of attributes: (x_{p_h}, y) for $d_{1|2}[1]$, (x, y_{p_v}) for $d_{1|2}[2]$, and (x, y) for the 3^{rd} and 4^{th} places of $d_{1|2}$. Clearly, $(x_{p_h}, x) \in C_h^I$, $(y_{p_v}, y) \in C_v^I$.

□

The above lemma implies, that if the source I contains l_h and l_v connected components in C_h^I and C_v^I respectively, then for any simulation mapping Σ, $chase(I, \Sigma)$ can be decomposed into $l_h \cdot l_v$ pairwise disconnected instances. By Lemma 28 we can consider every connected component in $C_{h|v}^I$ as an acyclic chain or a simple cycle. Then by applying one of the Lemmas 25, 26 in each of the $l_h \cdot l_v$ cases, the problem reduction can be lifted to the case when relations $C_{h|v}^I$ contain multiple connected components. We are thus able to prove the main result of this chapter.

Theorem 34. *The CQ-equivalence of mappings consisting of SO tgds and source key dependencies is undecidable.*

Proof. We show that for an arbitrary DS $\mathcal{D}$, the corresponding DP has a solution iff $\Sigma_{\mathcal{D}} \equiv_{CQ} \Sigma_{\mathcal{B}}$ holds. By the undecidability of the Domino Problem, the claim of the theorem will follow.

$[\Rightarrow]$. Assume that the domino problem associated with $\mathcal{D}$ has a solution. We show that for any source instance I, either the chase fails both under $\Sigma_{\mathcal{D}}$ and $\Sigma_{\mathcal{B}}$, or $core(I, \Sigma_{\mathcal{D}}) \cong core(I, \Sigma_{\mathcal{B}})$ holds.

Chase failure can be only due to the source key dependencies $\varepsilon_1, \varepsilon_2$ from Definition 28. Since $\Sigma_{\mathcal{D}}$ and $\Sigma_{\mathcal{B}}$ have the same KDs, the respective chases fail on exactly the same set of not admissible instances, namely on those containing a pattern $\{C(x, y_1), C(x, y_2)\}$, with $y_1 \neq y_2$.

Given an arbitrary admissible source instance I, we prove two claims:

1. *A subinstance $J_{\mathcal{B}}$ of $J = chase(I, \Sigma_{\mathcal{D}})$, isomorphic to $chase(I, \Sigma_{\mathcal{B}})$, exists.*

2. *An endomorphism $J \to J_{\mathcal{B}}$ holds.*

Proof of Claim 1. Let $I_{k,l} \subseteq I$ be an instance in which $C_h^{I_{k,l}}$ contains the k^{th} connected component of C_h^I, and $C_v^{I_{k,l}}$ contains the l^{th} connected component of C_v^I. Let $J_{k,l}$ denote an instance produced by chasing $I_{k,l}$ with $\Sigma_{\mathcal{B}}$. Since the antecedent of the simulation mapping $\Sigma_{\mathcal{B}}$ has only 2 atoms, the equality $chase(I, \Sigma_{\mathcal{B}}) = \bigcup_{i,j} J_{i,j}$ holds, where i, j range from 1 to the number of disconnected components in C_h^I and C_v^I, respectively. Moreover, by Lemma 29, subinstances $J_{k,l}$ define a partition of J, in which any two elements are disconnected. Therefore, we can prove the claim independently for each pair k, l and the corresponding source instance $I_{k,l}$ with connected $C_{h|v}^{I_{k,l}}$. We distinguish two cases:

(i) $I_{k,l}$ contains a cycle in one of its relations. By Lemma 28, we can assume that this relation contains exactly a single simple cycle. By Lemma 26, $chase(I_{k,l}, \Sigma_{\mathcal{D}})$ contains a subinstance isomorphic to $chase(I_{k,l}, \Sigma_{\mathcal{B}})$ in this case.

(ii) Both relations of $I_{k,l}$ are acyclic. By Lemma 28, we can assume that $I_{k,l}$ is a proper instance. That is, that both $C_h^{I_{k,l}}$ and $C_v^{I_{k,l}}$ contain a single acyclic chain of size l_h resp. l_v. By Lemma 25, we know that also in this case $\Sigma_{\mathcal{D}}$ contains a subinstance isomorphic to $chase(I_{k,l}, \Sigma_{\mathcal{B}})$. Hence, there exists an isomorphism h between $chase(I, \Sigma_{\mathcal{B}})$ and some subinstance of $chase(I, \Sigma_{\mathcal{D}})$.

Proof of Claim 2. Note that h was constructed in such a way preserving the arguments of the functional terms in D facts. Hence, $chase(I, \Sigma_{\mathcal{D}})$ contains a D fact associated with the pair (x, y) of the values from $dom(I)$ iff there is a fact associated with the same pair (x, y) in $chase(I, \Sigma_{\mathcal{B}})$.

Then, we construct a mapping $e\colon chase(I, \Sigma_{\mathcal{D}}) \to h(chase(I, \Sigma_{\mathcal{B}}))$ unifying, for each pair (x, y) of elements from $dom(I) \times dom(I)$, every fact from $chase(I, \Sigma_{\mathcal{D}})$ with the fact of $h(chase(I, \Sigma_{\mathcal{B}}))$ associated with this pair. By Properties 4 and 5, we conclude that e is indeed an endomorphism.

[$\Leftarrow$] Assuming $\Sigma_{\mathcal{D}} \equiv_{CQ} \Sigma_{\mathcal{B}}$ holds, we must show that a DP given by $\mathcal{D}$ has a solution. Our assumption implies, that for every $m, n \in \mathbb{N}$, and for every proper instance I of size (m, n), $core(I, \Sigma_{\mathcal{D}}) \cong core(I, \Sigma_{\mathcal{B}})$ holds. By Lemma 25, a consistent tiling $\mathcal{D}$ of an $m \times n$ grid exists. Hence, the domino problem corresponding to $\mathcal{D}$ has a solution. □

We finish this chapter by extending the above undecidability result to logical equivalence for mappings comprised of plain SO tgds (i.e., SO tgds without equalities in the antecedents of implications) and source key dependencies. To do this, we will need a result from [23].

Definition 31. *A schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is said to be* preserved under homomorphisms *if, for every source instance I and all target instances J and J' such that $(I, J) \in \Sigma$ and $J \to J'$, we have that $(I, J') \models \Sigma$.*

Fagin et al. showed that for mappings which are preserved under homomorphisms and possess the property that, for each source instance I, existence of solutions for I implies existence of a universal solution for I, CQ-equivalence implies logical equivalence [23, Proposition 3.14]. As argued in Chapter 2, the latter property holds for SO tgds in general. Preservation under homomorphisms can be easily seen to hold for plain SO tgds. Moreover, the same properties hold for mappings that additionally have source key dependencies. Hence, we obtain the following corollary, and, as an immediate consequence, the desired undecidability result for mappings based on plain SO tgds:

Corollary 2. *Let $\Sigma_{\mathcal{D}}$ be a simulation mapping for a DP $\mathcal{D}$. Then, $\Sigma_{\mathcal{D}} \equiv_{CQ} \Sigma_{\mathcal{B}}$ iff $\Sigma_{\mathcal{D}} \equiv \Sigma_{\mathcal{B}}$.*

Theorem 35. *The logical equivalence of mappings consisting of plain SO tgds and source key dependencies is undecidable.*

CHAPTER 9

Conclusion and future work

9.1 Summary

In this thesis we have extended the theory of schema mapping optimization in several directions: we have studied the optimization and normalization of schema mappings under logical equivalence, and then provided two results related to the alternative notions of equivalence.

In the first part of the thesis, we have presented several natural optimality criteria for schema mappings and devised a rewrite rule system for transforming any set of s-t tgds into a logically equivalent normal form, guaranteed to satisfy these criteria. Moreover, we have shown that the normal form of s-t tgds is unique up to renaming of variables.

In order to extend our rewrite rule system to schema mappings including target egds, the most important ingredients of our transformation (namely splitting and simplification of tgds) had to be defined very carefully so as not to destroy the uniqueness of the normal form. We have investigated several forms of splitting and of optimization and we have identified a rewrite rule system which indeed guarantees to produce a normal form that is again unique up to variable renaming. Finally, we have applied the normalization of schema mappings containing target egds to aggregate queries in data exchange. An implementation of the presented algorithms is freely available from `http://www.dbai.tuwien.ac.at/proj/sm`.

In the second part of the thesis, we have focused on the relaxed notions of equivalence for schema mappings. First of all, we have shown that neither DE- nor CQ-equivalence bring additional optimization power compared to logical equivalence, if the set of target dependencies is fixed. This is in a sharp contrast with the situation when the target dependencies can vary: a series of undecidability results from [49] renders equivalence testing and optimization w.r.t. DE- and to CQ-equivalence infeasible in the general case. However, we have shown that for a practical class of mappings with target constraints based on inclusion and functional dependencies, DE-equivalence is decidable and brings greater optimization power than logical equivalence, whereas CQ-equivalence remains undecidable. Yet more expressive target dependencies are al-

lowed if the s-t tgds in the schema mapping are free of constants. To the best of our knowledge, this has been the first evidence of a greater usefulness of DE-equivalence w.r.t. CQ-equivalence.

We have also studied the problem of equivalence testing for schema mappings based on SO-tgds. This problem has been completely unexplored until 2011, when Feinerer et al. [27] showed that logical equivalence is undecidable even for SO-tgds which are known to be CQ-equivalent. In this thesis we have shown that CQ-equivalence of schema mappings based on SO-tgds is undecidable, under the assumption that the source schema contains key dependencies. Without this provision, the decidability of CQ-equivalence for SO-tgds remains open. In their recent paper [21], Fagin and Kolaitis have shown that CQ-equivalence of two mappings, of which one is based on an SO-tgd and another on a set of s-t tgds (GLAV mapping, that is), is decidable.

9.2 Future work

The theory of schema mapping optimization is yet in its infancy. The following are examples of open problems in this theory that are immediately related to the work presented in this thesis. Many further practically relevant questions can be easily defined, taking into account a wide range of applications of schema mappings in information integration.

- The search for classes of mappings for which the relaxed notions of equivalence are decidable should be continued. In particular, the decidability of DE-equivalence for a larger class of mappings than that described in Chapter 7 seems plausible.
- CQ-equivalence of mappings based solely on SO-tgds (without integrity constraints on the source resp. target schema) remains open. Moreover, DE-equivalence of SO-tgds has not been studied. The recent results [21] indicate a potential for practically relevant decidable fragments of mappings based on SO-tgds.
- Equivalence of schema mappings defined over incomplete source instances (instances with labelled nulls, that is) should be studied.

9.3 Publications

This thesis summarizes the results published in conference proceedings and in journals. In particular, the results of Chapters 3 and 4 have been in part presented at the VLDB conference [30] and then appeared in the special issue "Best papers of VLDB 2009" of *The VLDB Journal* [31]. Chapters 6 and 7 are based on the work presented at ICDT 2011 [50], which has also been invited for the "Best Papers of ICDT 2011" issue of *Theory of Computing Systems* (currently available as online publication [49]). Chapter 8 contains an enhancement of a result presented at AMW 2011 [27].

Bibliography

[1] Foto N. Afrati and Phokion G. Kolaitis. Answering aggregate queries in data exchange. In *Proc. PODS'08*, pages 129–138, 2008.

[2] Cyril Allauzen and Bruno Durand. *The classical decision problem, by E. Börger, E. Grädel and Y. Gurevich*, chapter "Appendix A: Tiling Problems", pages 407–420. Springer, 2001.

[3] Marcelo Arenas, Pablo Barceló, Ronald Fagin, and Leonid Libkin. Locally consistent transformations and query answering in data exchange. In *Proc. PODS'04*, pages 229–240. ACM, 2004.

[4] Marcelo Arenas, Leopoldo E. Bertossi, Jan Chomicki, Xin He, Vijay Raghavan, and Jeremy Spinrad. Scalar aggregation in inconsistent databases. *Theor. Comput. Sci.*, 3(296):405–434, 2003.

[5] Marcelo Arenas, Jorge Pérez, Juan L. Reutter, and Cristian Riveros. Composition and inversion of schema mappings. *SIGMOD Record*, 38(3):17–28, 2009.

[6] Marcelo Arenas, Jorge Pérez, and Cristian Riveros. The recovery of a schema mapping: Bringing exchanged data back. *ACM Trans. Database Syst.*, 34(4), 2009.

[7] Catriel Beeri and Moshe Y. Vardi. A proof procedure for data dependencies. *J. ACM*, 31(4):718–741, 1984.

[8] R. Berger. *The undecidability of the domino problem*. Amer Mathematical Society, 1966.

[9] Philip A. Bernstein. Applying model management to classical meta data problems. In *Proc. CIDR'03*, 2003.

[10] Philip A. Bernstein, Todd J. Green, Sergey Melnik, and Alan Nash. Implementing mapping composition. *VLDB J.*, 17(2):333–353, 2008.

[11] Philip A. Bernstein and Sergey Melnik. Model management 2.0: manipulating richer mappings. In *Proc. SIGMOD'07*, pages 1–12. ACM, 2007.

[12] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. Datalog$^{\pm}$: a unified approach to ontologies and integrity constraints. In *Proceedings of the 12th International Conference on Database Theory*, ICDT '09, pages 14–30, New York, NY, USA, 2009. ACM.

[13] Andrea Calì, Georg Gottlob, Thomas Lukasiewicz, and Andreas Pieris. A logical toolbox for ontological reasoning. *SIGMOD Rec.*, 40(3):5–14, November 2011.

[14] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proc. STOC'77*, pages 77–90. ACM Press, 1977.

[15] Alin Deutsch, Alan Nash, and Jeffrey B. Remmel. The chase revisited. In *Proc. PODS'08*, pages 149–158. ACM, 2008.

[16] Alin Deutsch and Val Tannen. Xml queries and constraints, containment and reformulation. *Theor. Comput. Sci.*, 336(1):57–87, May 2005.

[17] Thomas Eiter and Michael Fink. Uniform equivalence of logic programs under the stable model semantics. In *ICLP*, pages 224–238, 2003.

[18] Thomas Eiter, Michael Fink, Hans Tompits, and Stefan Woltran. Simplifying logic programs under uniform and strong equivalence. In *LPNMR*, pages 87–99, 2004.

[19] Ronald Fagin. Horn clauses and database dependencies. *J. ACM*, 29(4):952–985, 1982.

[20] Ronald Fagin. Inverting schema mappings. In *Proc. PODS'06*, pages 50–59. ACM, 2006.

[21] Ronald Fagin and Phokion G. Kolaitis. Local transformations and conjunctive-query equivalence. In *PODS*, pages 179–190, 2012.

[22] 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.

[23] Ronald Fagin, Phokion G. Kolaitis, Alan Nash, and Lucian Popa. Towards a theory of schema-mapping optimization. In *Proc. PODS'08*, pages 33–42. ACM, 2008.

[24] Ronald Fagin, Phokion G. Kolaitis, and Lucian Popa. Data exchange: getting to the core. *ACM TODS*, 30(1):174–210, 2005.

[25] 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.

[26] Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang Chiew Tan. Reverse data exchange: Coping with nulls. *ACM Trans. Database Syst.*, 36(2):11, 2011.

[27] Ingo Feinerer, Reinhard Pichler, Emanuel Sallinger, and Vadim Savenkov. On the undecidability of the equivalence of second-order tuple generating dependencies. In *Proc. AMW'11*, volume 749 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.

[28] Haim Gaifman, Harry G. Mairson, Yehoshua Sagiv, and Moshe Y. Vardi. Undecidable optimization problems for database logic programs. In *LICS*, pages 106–115, 1987.

[29] Georg Gottlob and Christos H. Papadimitriou. On the complexity of single-rule datalog queries. *Inf. Comput.*, 183(1):104–122, 2003.

[30] Georg Gottlob, Reinhard Pichler, and Vadim Savenkov. Normalization and optimization of schema mappings. *PVLDB*, 2(1):1102–1113, 2009.

[31] Georg Gottlob, Reinhard Pichler, and Vadim Savenkov. Normalization and optimization of schema mappings. *VLDB J.*, 20(2):277–302, 2011.

[32] Alon Y. Halevy, Anand Rajaraman, and Joann J. Ordille. Data integration: The teenage years. In *Proc. VLDB'06*, pages 9–16. ACM, 2006.

[33] Pavol Hell and Jaroslav Nešetřil. The core of a graph. *Discrete Mathematics*, 109(1-3):117–126, 1992.

[34] André Hernich and Nicole Schweikardt. Cwa-solutions for data exchange settings with target dependencies. In *Proc. PODS'07*, pages 113–122. ACM, 2007.

[35] Tomasz Imielinski and Witold Lipski Jr. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.

[36] David S. Johnson and Anthony C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. Comput. Syst. Sci.*, 28(1):167–189, 1984.

[37] Phokion G. Kolaitis. Schema mappings, data exchange, and metadata management. In *Proc. PODS'05*, pages 61–75, 2005.

[38] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proc. PODS'02*, pages 233–246. ACM, 2002.

[39] Leonid Libkin. Data exchange and incomplete information. In *Proc. PODS'06*, pages 60–69. ACM Press, 2006.

[40] Leonid Libkin and Cristina Sirangelo. Data exchange and schema mappings in open and closed worlds. In *Proc. PODS'08*, pages 139–148. ACM, 2008.

[41] Jayant Madhavan and Alon Y. Halevy. Composing mappings among data sources. In *Proc. VLDB'03*, pages 572–583, 2003.

[42] M. Maher. Equivalences of logic programs. In Ehud Shapiro, editor, *Third International Conference on Logic Programming*, volume 225 of *Lecture Notes in Computer Science*, pages 410–424. Springer Berlin / Heidelberg, 1986.

[43] Bruno Marnette. Generalized schema-mappings: from termination to tractability. In *Proc. PODS'09*, pages 13–22, 2009.

[44] Bruno Marnette, Giansalvatore Mecca, and Paolo Papotti. Scalable data exchange with functional dependencies. *PVLDB*, 3(1):105–116, 2010.

[45] Giansalvatore Mecca, Paolo Papotti, and Salvatore Raunich. Core schema mappings: Scalable core computations in data exchange. *Inf. Syst.*, 37(7):677–711, 2012.

[46] Michael Meier, Michael Schmidt, and Georg Lausen. On chase termination beyond stratification. *PVLDB*, 2(1):970–981, 2009.

[47] Sergey Melnik. *Generic Model Management: Concepts and Algorithms*, volume 2967 of *LNCS*. Springer, 2004.

[48] Alan Nash, Philip A. Bernstein, and Sergey Melnik. Composition of mappings given by embedded dependencies. *ACM Trans. Database Syst.*, 32(1):4, 2007.

[49] Reinhard Pichler, Emanuel Sallinger, and Vadim Savenkov. Relaxed notions of schema mapping equivalence revisited. *Theory of Computing Systems*, pages 1–59. 10.1007/s00224-012-9397-0.

[50] Reinhard Pichler, Emanuel Sallinger, and Vadim Savenkov. Relaxed notions of schema mapping equivalence revisited. In *Proc. ICDT'11*, pages 90–101. ACM, 2011.

[51] Reinhard Pichler and Vadim Savenkov. Towards practical feasibility of core computation in data exchange. *Theor. Comput. Sci.*, 411(7-9):935–957, 2010.

[52] Y. Sagiv. Optimizing datalog programs. In *Proceedings of the sixth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '87, pages 349–362, New York, NY, USA, 1987. ACM.

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

[54] Emanuel Sallinger. Optimizing schema mappings with relaxed notions of equivalence. Master's thesis, Technische Unversität Wien, Austria, 2010.

[55] Oded Shmueli. Equivalence of datalog queries is undecidable. *J. Log. Program.*, 15(3):231–241, 1993.

[56] Francesca Spezzano and Sergio Greco. Chase termination: A constraints rewriting approach. *PVLDB*, 3(1):93–104, 2010.

[57] Balder ten Cate, Laura Chiticariu, Phokion G. Kolaitis, and Wang Chiew Tan. Laconic schema mappings: Computing the core with sql queries. *PVLDB*, 2(1):1006–1017, 2009.

APPENDIX A

Curriculum Vitae

Vadim A. Savenkov

Degrees	MSc., Russian "Engineer's Diploma"	
Citizenship	Russian Federation	
Current Position	Research Assistant	
Contact Information	DBAI Group Institute of Information Systems Vienna University of Technology Favoritenstraße 9–11 A–1040 Vienna, Austria	*E-mail:* savenkov@dbai.tuwien.ac.at *Phone (office):* +43-1-58801-18445 *Mobile:* +43-680-211-19-97
Homepage	www.dbai.tuwien.ac.at/staff/savenkov	
Professional interests	Theory and practice of data and knowledge management, Information integration	

Education

Vienna University of Technology, Austria

Ph.D., Computer Science (expected graduation: Fall 2012)

- Thesis Topic: Normalization and Optimization of Schema mappings
- Advisor: Prof. Dr. Reinhard Pichler
- Areas of Research: Information Integration, Database Theory

European Master Program in Computational Logic:
Vienna University of Technology, Austria
Dresden University of Technology, Germany

M.Sc., Computational Logic (October 2007)

- Thesis Topic: Implementing Core Computation in Data Exchange
- Advisor: Prof. Dr. Reinhard Pichler
- Areas of Study: Computational Logic, Database Theory

Bauman Moscow State Technical University

Engineer, Information Technologies, 2007
Russian national diploma, awarded after 5 years and 10 months of study, or 2 years on top of the Bachelor's course

- Thesis Topic: Automatic Classification of Speech Signals
- Advisor: Prof. Dr. Vladimir Deviatkov
- Areas of Study: Computer and Microprocessor Systems Engineering, Digital Signal Processing, Artificial Intelligence

B.S., Information Technologies, 2004

- *With Honors*, in Information Technology
- Thesis title: Universal Communication Agent
- Areas of Study: Basic Engineering Disciplines (Mathematics, Physics, etc); Elecronics, Control Theory, Software and Database Design

Professional Record

Vienna University of Technology, Vienna, Austria **2008 to present date**

Research assistant in the Database and Artificial Intelligence Group

Research in the area of Information Integration and Data Management. Participation in projects:

- "SODI – Service-Oriented Data Integration", supported by Viennese Science and Technology Fund (WWTF)
- "METAMORPH – A special-purpose semantic meta-search generator for integrating deep web data", supported by Austrian Society for Development of Research (FFG).

BELL Integrator LLC, Moscow, Russia

Software Development Group Lead 11/2004 to 09/2005

Senior Software Developer 2003 to 11/2004

Bauman Moscow State Technical University
Institute for Manufacturing Automation

Software developer 2001 to 2003

Awards and scholarships

- European Erasmus Mundus External Cooperation Window Scholarship for PhD students, 2009–2011
- Best Thesis Award of the Computer Science Department, TU Vienna, Summer Term 2008
- European Erasmus Mundus Scholarship for Master's students, 2005–2007
- *Ministery of Education, Russian Federation.* Best student work award, 2003

Scientific Activities

Referee Tasks

- Conferences: EDBT, ICDT, LPNMR, ICALP, FOIKS, DEXA.
- Journals: WWW, Information Systems, Acta Informatica.

Participation in Summer Schools

- GI Dagstuhl Seminar "Data Exchange, Integration, and Streams", Schloss Dagstuhl, Germany, November 2010
- GII Doctoral School on Advances in Databases, Rende – Cetraro, Italy, September 2009,
- Bertinoro Workshop on Information Integration, Bertinoro, Italy, September 2007.

Publications

Journal Publications

- Reinhard Pichler, Emanuel Sallinger, and Vadim Savenkov. Relaxed notions of schema mapping equivalence revisited. *Theory of Computing Systems*, pages 1–59. 10.1007/s00224-012-9397-0
- Georg Gottlob, Reinhard Pichler, and Vadim Savenkov. Normalization and optimization of schema mappings. *VLDB Journal*, 20(2):277–302, 2011
- Reinhard Pichler and Vadim Savenkov. Towards practical feasibility of core computation in data exchange. *Theoretical Computer Science*, 411(7-9):935–957, 2010

Peer-Reviewed Conferences and Workshops

- Jorge Pérez, Reinhard Pichler, Emanuel Sallinger, and Vadim Savenkov. Union and intersection of schema mappings. In *Proceedings of 6th Alberto Mendelzon International Workshop on Foundations of Data Management, AMW 2012, Ouro Preto, Brazil, June 27-30, 2012*. CEUR-WS.org
- Ingo Feinerer, Reinhard Pichler, Emanuel Sallinger, and Vadim Savenkov. On the undecidability of the equivalence of second-order tuple generating dependencies. In *Proceedings of the 5th Alberto Mendelzon International Workshop on Foundations of Data Management, Santiago, Chile, May 9-12, 2011*, volume 749 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011
- Reinhard Pichler, Emanuel Sallinger, and Vadim Savenkov. Relaxed notions of schema mapping equivalence revisited. In *Database Theory - ICDT 2011, 14th International Conference, Uppsala, Sweden, March 21-24, 2011, Proceedings*, pages 90–101. ACM, 2011

- Reinhard Pichler, Vadim Savenkov, Sebastian Skritek, and Hong Linh Truong. Uncertain databases in collaborative data management. In *Proceedings of the Fourth International VLDB workshop on Management of Uncertain Data (MUD 2010) in conjunction with VLDB 2010, Singapore, September 13, 2010*, volume WP10-04 of *CTIT Workshop Proceedings Series*, pages 129–143. Centre for Telematics and Information Technology (CTIT), University of Twente, The Netherlands, 2010
- Georg Gottlob, Reinhard Pichler, and Vadim Savenkov. Normalization and optimization of schema mappings. *PVLDB*, 2(1):1102–1113, 2009
- Reinhard Pichler and Vadim Savenkov. Demo: Data exchange modeling tool. *PVLDB*, 2(2):1606–1609, 2009
- Reinhard Pichler and Vadim Savenkov. Towards practical feasibility of core computation in data exchange. In *Proc. LPAR 2008: Doha, Quatar*, pages 62–78, 2008

Surveys

- Schahram Dustdar, Reinhard Pichler, Vadim Savenkov, and Hong Linh Truong. Quality-aware service-oriented data integration: requirements, state of the art and open challenges. *SIGMOD Record*, 41(1):11–19, 2012
- Vadim Savenkov. Algorithms for core computation in data exchange. In *Dagstuhl Follow-Up Series, GI Dagstuhl Seminar "Data Exchange, Integration, and Streams" 2010*. To appear

CERTIFICATES

- German Course, Level: 7^{th} (of 8 possible), University of Vienna, 2007
- Microsoft Certified Software Developer, 2005
- TOEFL (270 out of 300), 2004

PERSONAL INFORMATION

Birth date: 7 July, 1981

Spoken languages: Russian (native), English (fluent in speaking and writing), German (intermediate)

Marital status: Married, two children