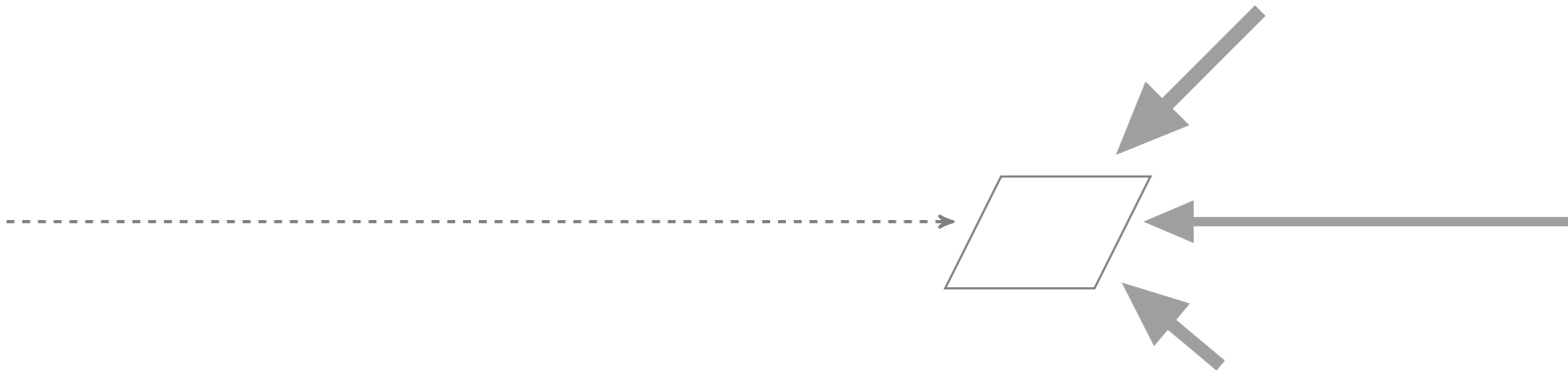
Foundations of Query Answering in Relational Data Exchange



André Hernich

Foundations of Query Answering in Relational Data Exchange

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften

vorgelegt beim Fachbereich Informatik und Mathematik
der Johann Wolfgang Goethe-Universität
in Frankfurt am Main

von

André Hernich

aus Prenzlau

Frankfurt 2010
(D 30)

vom Fachbereich Informatik und Mathematik der
Johann Wolfgang Goethe-Universität als Dissertation angenommen.

Dekan:	Prof. Dr.-Ing. Detlef Krömker
Gutachter:	Prof. Dr. Nicole Schweikardt Prof. Dr. Phokion G. Kolaitis
Datum der Disputation:	18. November 2010

Abstract

Data exchange deals with translating data structured in some format into data structured in some other format, according to a specification of the relationship between the source data and the target data. Such data translation tasks are very common in practice. They arise as one of the many tasks in data integration, for example, in data restructuring, in ETL (Extract-Transform-Load) processes used for updating data warehouses, or in data exchange between different, possibly independently created, applications. While systems for data exchange have been implemented over the past decades, research on the theoretical foundations of data exchange started only recently with the influential article by Fagin, Kolaitis, Miller and Popa. This thesis deals with *relational data exchange*, where the source data and the target data are *relational*.

The basic setting in relational data exchange is the following. We are given a *schema mapping* M that consists of a *source schema* (the format of the source data) and a *target schema* (the format of the target data), and is defined by a finite set Σ of logical formulas which describes the relationship between the source data and the target data. For a source database S, the task is then to find a *solution for S under M*, that is, a target database so that all formulas in Σ are satisfied. Such a solution should reflect S as accurately as possible. Usually, Σ is a set of *tuple generating dependencies (tgds)* and *equality generating dependencies (egds)*. Here, tgds are first-order formulas of the form

$$\forall \bar{x} \forall \bar{y} (\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z})),$$

where φ and ψ are conjunctions of relational atomic formulas $R(\bar{u})$, and $\bar{x}, \bar{y}$ and $\bar{x}, \bar{z}$ are tuples of variables that contain precisely all variables in φ and ψ, respectively. One distinguishes between *source-to-target tgds (st-tgds)*, where φ "speaks" about the source schema and ψ speaks about the target schema, and *target tgds (t-tgds)*, where both φ and ψ speak about the target schema. An *egd* is a first-order formula of the form

$$\forall \bar{x} (\varphi(\bar{x}) \rightarrow x_i = x_j),$$

where φ is a conjunction of relational atomic formulas that speak about the target schema only, $\bar{x}$ is a tuple of variables that contains precisely all variables in φ, and x_i, x_j occur in $\bar{x}$.

One of the major issues in relational data exchange is how to answer queries that are posed against the target schema (i.e., queries that are posed against the result of a data translation). The problem is that schema mappings are

in general underspecified. In particular, there is often more than one possible solution for a given source database, so that it is not a priori clear what the answer to a query should be. A popular approach is to return *the certain answers* to a query. That is, the set of answers to a query q on a schema mapping M and a source database S consists of all tuples $\bar{t}$ such that for all solutions T for S under M we have: $\bar{t}$ belongs to the set of answers to q on T (written $\bar{t} \in q(T)$). For a large class of queries, including *unions of conjunctive queries* which are fundamental in database theory, the issue of how to compute the certain answers to such queries has been investigated quite well. Here, an indispensable tool are the *universal solutions*, introduced by Fagin, Kolaitis, Miller and Popa. Informally, universal solutions are "most general" solutions. In particular, it was shown that for many queries q, including unions of conjunctive queries, computing the certain answers to q on M and S eventually boils down to evaluating q on an arbitrary universal solution for S under M.

For *monotonic queries* (queries, for which the set of answers does not decrease when adding tuples to the database), which also comprise the abovementioned unions of conjunctive queries, the certain answers intuitively correspond to the set of answers a user would expect. However, it has been observed that for some *non-monotonic* queries, the certain answers lead to answers that intuitively do not seem to be accurate. The reason is that schema mappings are often interpreted with additional *implicit* information – information that is not mentioned explicitly by the schema mapping, but, due to the point of view on the schema mapping, are nevertheless assumed to be implicitly represented in the schema mapping. Since there are many possible ways of formally capturing the intuitive notion of "implicit information", several semantics for query answering taking into account implicit information have been proposed. Those semantics are based on the *closed world assumption (CWA)*, and their definition is based on the following idea. Given a schema mapping M and a source database S for M,

1. identify a subset $\mathcal{S}$ of all solutions for S under M that is intended to be the set of all possible outcomes of translating S to the target if implicit information—in the formalized sense—is taken into account, and

2. answer queries q on M and S using the set $\mathcal{S}$, typically by taking the certain answers to q on $\mathcal{S}$ (i.e., the set of all tuples $\bar{t}$ such that $\bar{t} \in q(T)$ for all $T \in \mathcal{S}$).

Depending on the particular application and one's point of view, one or the other of these semantics may be useful.

The contributions of this thesis can be subdivided into three parts: 1. undecidability results concerning computation of universal solutions and the so-called

chase procedure, 2. query answering semantics that take into account implicit information, and 3. the complexity of evaluating queries with respect to the semantics considered in 2. In the following, these parts are described in more detail.

1. Undecidability results concerning computation of universal solutions and the so-called chase procedure.

A schema mapping M defined by tgds only is constructed such that the following problem is undecidable: given a source database S for M, does S have a universal solution under M? This in particular strengthens a result of Deutsch, Nash, and Remmel (2008).

Furthermore, the proof of this result has several consequences concerning termination of the *chase procedure*, which is essential in database theory and is employed for computing universal solutions. More precisely, the chase is a procedure that takes a database I and a set Σ of tgds and egds as input, and iteratively tries to modify I so that the resulting database satisfies all tgds and egds in Σ. Unfortunately, the chase does not always terminate. To this end, various conditions on Σ that ensure chase termination have been proposed in the literature. All of these conditions are *sufficient*, but not necessary for chase termination. In fact, it follows from the proof of the above-mentioned result that there is no *decidable* condition on Σ that is both sufficient and necessary for chase termination: chase termination is undecidable, even with respect to some fixed set Σ of tgds. This also strengthens a result of Deutsch, Nash, and Remmel (2008).

2. Query answering semantics that take into account implicit information.

This thesis gives an overview of query answering semantics in relational data exchange that take into account implicit information. In the following, a more detailed description of the semantics contributed by this thesis is given.

The first query answering semantics that take into account implicit information were introduced by Libkin. These semantics are based on *CWA-solutions*, which were tailored by Libkin to schema mappings defined by st-tgds. CWA-solutions are based on the CWA in the following sense:

1. Every tuple must be justified in some sense by the schema mapping and the source database.

2. Each justification is used at most once.

3. A CWA-solution contains only "facts" that follow from the schema mapping and the source database.

This thesis extends the definition of CWA-solutions to the more general case of schema mappings defined by tgds and egds. The main difficulty is to formalize the first two requirements. We do this in two ways: First, we use a derivation-based approach using a suitably controlled version of the chase. Second, we obtain an equivalent definition in terms of a game. We then show the following:

- CWA-solutions are universal solutions that can be derived as mentioned above.
- A source database has a CWA-solution if and only if it has a universal solution.
- The *core of the universal solutions* introduced by Fagin, Kolaitis and Popa (the "smallest" universal solution) is the "smallest" CWA-solution.

Furthermore, the structure of the set of all CWA-solutions and the complexity of computing CWA-solutions is explored. Finally, this thesis addresses the complexity of the query evaluation problem with respect to the CWA-solution-based semantics. Details to the latter topic are given in 3. below.

The CWA-solution-based semantics reflect an operational point of view on tgds and egds. That is, tgds are considered as rules for deriving tuples, and egds are considered as rules for identifying values. One of the consequences is that these semantics do not take into account logical equivalence of schema mappings (i.e., answers to queries may differ on schema mappings defined by logically equivalent sets of formulas), and query answers do not necessarily reflect the standard semantics of first-order quantifiers (e.g., existential quantifiers express that there are one, two, three or more elements that satisfy the given property, but this is not necessarily reflected by query answers).

For this reason, a second semantics, the *GCWA*-semantics*, is developed that takes implicit information into account and additionally respects logical equivalence of schema mappings and reflects the standard semantics of first-order quantifiers. First, translations of query answering semantics from the area of deductive databases are studied in the context of relational data exchange. Inspired by these semantics, the GCWA*-semantics is developed. Under the GCWA*-semantics, queries are answered by the certain answers on *GCWA*-solutions*. In contrast to the preceding semantics and solution concepts, the GCWA*-semantics and GCWA*-solutions are defined for *all* schema mappings (rather than for schema mappings defined by tgds and egds). For schema mappings defined by st-tgds and egds, GCWA*-solutions are simply solutions without null values that are unions of inclusion-minimal solutions.

3. The complexity of evaluating queries with respect to the semantics in 2.

This thesis first addresses the complexity of evaluating queries with respect to the CWA-solution-based query answering semantics. More precisely, the *data*

complexity is considered, that is, the complexity with respect to *fixed* schema mappings and *fixed* queries. It turns out that for a large number of monotonic queries, including unions of conjunctive queries, two of the CWA-solution-based semantics yield precisely the above-mentioned certain answers, so that all results on computing the certain answers to such queries carry over to these semantics. For properly restricted schema mappings, the query evaluation problem for first-order queries with respect to the CWA-solution-based semantics is in co-NP or NP, depending on the semantics. Furthermore, there are simple schema mappings and conjunctive queries with just one additional inequality ($\neq$) such that this problem is complete for the corresponding class. In contrast, it is known that the certain answers to such queries can be computed efficiently with respect to a large number of schema mappings.

A larger part of this thesis deals with the complexity of evaluating queries with respect to the GCWA*-semantics. As above, the data complexity is considered. It is shown that on monotonic queries, the GCWA*-semantics yields precisely the above-mentioned certain answers, so that all results obtained for computing certain answers on monotonic queries directly apply to the GCWA*-semantics. However, there are simple schema mappings M defined by st-tgds and simple *existential queries* q (queries of the form $\exists \bar{x}\, \varphi$, where φ is quantifier-free) such that the query evaluation problem

$\textsc{Eval}(M, q)$	
Input:	a source database S for M and a tuple $\bar{t}$
Question:	Is $\bar{t}$ an answer to q with respect to the GCWA*-semantics?

is co-NP-hard. Permitting only one additional universal quantifier can make this problem undecidable. It seems then surprising that $\textsc{Eval}(M, q)$ is in PTIME for *universal queries* q (queries of the form $\forall \bar{x}\, \varphi$, where φ is quantifier-free) and suitable restrictions on M.

This result is explained in more detail below. Precisely it states that for every schema mapping M defined by certain st-tgds, called *packed st-tgds*, and each universal query q, there is a polynomial time algorithm that takes the core of the universal solutions for some source database S as input and computes the set of all answers to q on M and S with respect to the GCWA*-semantics. Standard results on computing the core of the universal solutions (e.g., by Fagin, Kolaitis and Popa) in particular imply that $\textsc{Eval}(M, q)$ belongs to PTIME.

For proving the main result, it suffices to develop a polynomial time algorithm for the following problem: given the core of the universal solutions for some source database S and a tuple $\bar{t}$, does $\bar{t}$ belong to the set of answers to q on M and S with respect to the GCWA*-semantics? To this end, the problem

is reduced to the problem of checking whether there is a union of one or more inclusion-minimal databases in $poss(T_0)$ that satisfies $\neg q(\bar{t})$. Here, $poss(T_0)$ is the set of all databases obtained from the core T_0 of the universal solutions for S by replacing null values with constants. By transforming $\neg q$ into a kind of disjunctive normal form, one can then focus, without loss of generality, on the case that $\neg q$ is logically equivalent to a formula $\bar{q} = \exists \bar{x}\, \varphi$, where φ is a conjunction of atomic formulas and negations of atomic formulas. If φ consists of one atomic formula and $\bar{x}$ is the empty tuple, the problem can be solved as follows. First, the infinite set of all inclusion-minimal databases in $poss(T_0)$ is reduced to a set $\mathcal{S}$ of possibly exponential size. The technically difficult part is then to identify a particular subset of $\mathcal{S}$ of polynomial size from which all tuples that occur in inclusion-minimal databases of $poss(T_0)$ can be reconstructed. Finally, for solving the general problem, partial solutions are combined in a suitable way to solutions to the whole problem, where the results proved for the case of one atomic formula help to prove correctness of this construction.

Keywords: data exchange, certain answers, closed world assumption (CWA), deductive database

Zusammenfassung

Beim Datenaustausch geht es darum, Daten von einem Format in ein anderes Format gemäß einer vorgegebenen Spezifikation zu transformieren. Solche Datentransformationen finden sich in vielen Anwendungsbereichen wieder. Sie kommen als eine der vielen Aufgaben in der Datenintegration vor, zum Beispiel bei der Datenrestrukturierung, bei der Aktualisierung von Datenwarenhäusern oder beim Datenaustausch zwischen verschiedenen, möglicherweise unabhängig voneinander erstellten Anwendungen. Obwohl Systeme für den Datenaustausch bereits seit einiger Zeit implementiert werden, wurde erst mit der einflussreichen Arbeit von Fagin, Kolaitis, Miller und Popa (2005) damit begonnen, die theoretischen Grundlagen des Datenaustauschs zu erforschen. Diese Dissertation beschäftigt sich mit *relationalem* Datenaustausch, bei dem die Quell- und die Zieldaten *relational* sind.

Die grundsätzliche Problemstellung im relationalen Datenaustausch ist die Folgende: Gegeben ist ein so genanntes *Schema-Mapping* M, das aus einem *Quellschema* (dem Format der Quelldatenbank) und einem *Zielschema* (dem Format der Zieldatenbank) besteht und durch eine endliche Menge Σ von logischen Formeln definiert wird, die die Beziehung zwischen Quell- und Zieldaten beschreiben. Für eine Quelldatenbank S soll dann eine *Lösung für S unter M* gefunden werden, d.h. eine Zieldatenbank, so dass alle Formeln aus Σ erfüllt sind. Diese Lösung sollte S so genau wie möglich wiederspiegeln. Gewöhnlich ist Σ dabei eine Menge von so genannten *tgds* („tuple generating dependencies“) und *egds* („equality generating dependencies“). Hierbei sind tgds Formeln der Logik erster Stufe der Form

$$\forall \bar{x} \forall \bar{y} (\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z})),$$

wobei φ und ψ Konjunktionen von Relationsatomen $R(\bar{u})$ und $\bar{x}, \bar{y}$ bzw. $\bar{x}, \bar{z}$ Variablentupel sind, die genau die in φ bzw. ψ frei vorkommenden Variablen enthalten. Man unterscheidet zwischen *st-tgds* („source-to-target tgds“, bei denen φ nur über das Quellschema und ψ nur über das Zielschema „spricht“) und *t-tgds* („target tgds“, bei denen φ und ψ beide nur über das Zielschema sprechen). Ein *egd* ist eine Formel der Form

$$\forall \bar{x} (\varphi(\bar{x}) \rightarrow x_i = x_j),$$

wobei φ eine Konjunktion von Relationsatomen ist, die nur über das Zielschema sprechen, $\bar{x}$ ein Variablentupel mit genau den in φ frei vorkommenden Variablen ist und x_i, x_j Variablen aus $\bar{x}$ sind.

Eine wichtige Frage im relationalen Datenaustausch ist, wie Anfragen über dem Zielschema (d.h. Anfragen an das Resultat des Datenaustauschs) beantwortet werden sollen. Das Problem ist, dass Schema-Mappings im Allgemeinen unterspezifiziert sind. Insbesondere gibt es oft mehrere mögliche Lösungen zu einer Quelldatenbank, so dass nicht a priori klar ist, was die Antwort zu einer Anfrage sein soll. Ein Ansatz ist, nur die *sicheren Antworten* zu einer Anfrage zurückzuliefern. D.h. die Antwortmenge zu einer Anfrage q bzgl. eines Schema-Mappings M und einer Quelldatenbank S besteht aus allen Tupeln $\bar{t}$, so dass für jede Lösung T für S unter M gilt: $\bar{t}$ liegt in der Antwortmenge zu q auf T (kurz: $\bar{t} \in q(T)$). Für eine große Klasse von Anfragen, die insbesondere die in der Datenbanktheorie wichtige Klasse der *konjunktiven Anfragen* enthält, wurde bereits gut erforscht, wie die sicheren Antworten für solche Anfragen berechnet werden können. Ein wichtiges Hilfmittel dabei sind die von Fagin, Kolaitis, Miller und Popa eingeführten *universellen Lösungen*, die intuitiv "allgemeinste Lösungen" sind. Insbesondere wurde gezeigt, dass für viele Anfragen q, darunter auch konjunktive Anfragen, die Berechnung der sicheren Antworten zu q bzgl. M und S im Prinzip nichts Anderes ist, als die Auswertung von q auf einer beliebigen universellen Lösung für S unter M.

Für so genannte *monotone Anfragen* (Anfragen, für die die Antwortmenge nicht kleiner wird, wenn Tupel zur Datenbank hinzugefügt werden), zu denen auch die oben erwähnten konjunktiven Anfragen gehören, sind die sicheren Antworten intuitiv genau die Antworten, die man als Benutzer erwarten würde. Es wurde aber beobachtet, dass für einige *nicht-monotone* Anfragen die sicheren Antworten nicht dem entsprechen, was man intuitiv erwarten würde. Der Grund dafür ist, dass Schema-Mappings oft mit zusätzlichen *impliziten* Informationen interpretiert werden – Informationen, die im Schema-Mapping nicht explizit erwähnt, jedoch durch die Sichtweise auf das Schema-Mapping oft als implizit gegeben angesehen werden. Da es viele Möglichkeiten gibt, den hochgradig intuitiven Begriff „implizite Informationen“ formal einzufangen, existieren verschiedene Anfragesemantiken, mit denen Anfragen unter Berücksichtigung solcher impliziten Informationen beantwortet werden können. Diese Semantiken basieren auf Varianten der *Closed World Assumption (CWA)* und werden basierend auf der folgenden Idee definiert. Für ein gegebenes Schema-Mapping M und eine Quelldatenbank S

1. identifiziert man die Menge $\mathcal{S}$ aller Lösungen für S unter M, die mögliche Resultate der Transformation von S darstellen, wenn implizite Informationen – im jeweils formalisierten Sinn – berücksichtigt werden, und

2. beantwortet Anfragen q mit Hilfe der Menge $\mathcal{S}$, typischerweise durch die sicheren Antworten zu q bzgl. $\mathcal{S}$ (d.h. die Menge der Tupel $\bar{t}$, so dass $\bar{t} \in q(T)$ für alle $T \in \mathcal{S}$).

Abhängig von der konkreten Anwendung und der eigenen Sichtweise kann die eine oder die andere Semantik sinnvoll sein.

Die Beiträge dieser Dissertation können grob in drei Gruppen eingeteilt werden: 1. Unentscheidbarkeitsresultate hinsichtlich der Berechnung universeller Lösungen und der so genannten Chase-Prozedur, 2. Anfragesemantiken zur Beantwortung von Anfragen unter Berücksichtigung impliziter Informationen und 3. Bestimmung der Komplexität der Anfrageverarbeitung bezüglich der in 2. betrachteten Anfragesemantiken. Im Folgenden werden diese drei Gruppen etwas näher beschrieben.

1. *Unentscheidbarkeitsresultate hinsichtlich der Berechnung universeller Lösungen und der so genannten Chase-Prozedur.*

Es wird ein nur durch tgds definiertes Schema-Mapping M konstruiert, so dass folgendes Problem unentscheidbar ist: Gegeben eine Quelldatenbank S für M, besitzt S eine universelle Lösung unter M? Dieses Resultat verstärkt insbesondere ein Ergebnis von Deutsch, Nash und Remmel (2008).

Weiterhin hat der Beweis dieses Resultats einige Konsequenzen in Bezug auf das Problem, ob die in der Datenbanktheorie essentielle und zur Berechnung von universellen Lösungen eingesetzte *Chase-Prozedur* terminiert. Die Chase-Prozedur bekommt als Eingabe eine Datenbank I und eine Menge Σ von tgds und egds. Sie versucht dann, I iterativ mittels der tgds und egds in Σ so zu modifizieren, dass die resultierende Datenbank die tgds und egds in Σ erfüllt. Unglücklicherweise terminiert die Chase-Prozedur nicht immer. Jedoch wurde eine Reihe von Kriterien an Σ vorgeschlagen, die sicherstellen, dass die Chase-Prozedur terminiert. All diese Kriterien sind hinreichend, aber nicht notwendig für die Terminierung der Chase-Prozedur. Tatsächlich folgt aus dem Beweis des oben genannten Resultats, dass es kein *entscheidbares* hinreichendes und notwendiges Kriterium an Σ gibt, das die Terminierung der Chase-Prozedur sicherstellt. Es gibt nämlich eine Menge Σ, die nur tgds enthält, so dass das folgende Problem unentscheidbar ist: Gegeben eine Datenbank I, terminiert die Chase-Prozedur für I und Σ? Dies verstärkt ebenfalls ein Ergebnis von Deutsch, Nash und Remmel (2008).

2. *Anfragesemantiken zur Beantwortung von Anfragen unter Berücksichtigung impliziter Informationen.*

Die Dissertation gibt einen Überblick über Anfragesemantiken, die implizite Informationen berücksichtigen, und steuert selbst entsprechende Semantiken bei. Im Folgenden werden die Hauptbeiträge dieser Dissertation zu diesem Thema zusammengefasst.

Die ersten Anfragesemantiken, die implizite Informationen berücksichtigen, wurden von Libkin eingeführt. Diese Semantiken basieren auf *CWA-Lösungen*,

die von Libkin speziell auf durch st-tgds definierte Schema-Mappings zugeschnitten wurden. CWA-Lösungen basieren im folgenden Sinn auf der CWA:

1. Alle Tupel müssen auf eine bestimmte Art durch das Schema-Mapping und die Quelldatenbank gerechtfertigt sein.

2. Jede mögliche Rechtfertigung wird nur einmal eingesetzt.

3. Die CWA-Lösung enthält nur „Fakten“, die aus dem Schema-Mapping und der Quelldatenbank folgen.

In dieser Dissertation wird die Definition von CWA-Lösungen auf durch tgds und egds definierte Schema-Mappings erweitert. Die Hauptschwierigkeit besteht in der Formalisierung der Anforderungen 1 und 2, die hier zum Einen durch einen ableitungsbasierten Ansatz mittels einer passend kontrollierten Variante der Chase-Prozedur und zum Anderen spielbasiert charakterisiert werden. Es wird dann folgendes gezeigt:

- CWA-Lösungen sind universelle Lösungen, die in dem oben genannten Sinn ableitbar sind.

- Eine Quelldatenbank besitzt genau dann eine CWA-Lösung, wenn sie eine universelle Lösung besitzt.

- Der von Fagin, Kolaitis und Popa eingeführte Kern der universellen Lösungen (die „kleinste“ universelle Lösung) ist die „kleinste“ CWA-Lösung.

Weiterhin wird die Struktur der Menge der CWA-Lösungen und die Komplexität der Berechnung von CWA-Lösungen untersucht. Schließlich wendet sich die Dissertation der Komplexität des Auswertungsproblems für die CWA-Lösungsbasierten Anfragesemantiken zu. Dies wird unter 3. näher beschrieben.

Die den CWA-Lösungs-basierten Semantiken zugrunde liegenden Annahmen spiegeln die operationale Sichtweise auf tgds und egds wieder. Infolgedessen berücksichtigen diese Semantiken jedoch nicht logische Äquivalenz zwischen Schema-Mappings (d.h. auf Schema-Mappings, die durch logisch äquivalente Mengen von Formeln definiert sind, können Anfragen verschieden beantwortet werden), und auch die Standard-Semantik von Quantoren der Logik erster Stufe (z.B. Existenzquantoren sagen aus, dass ein oder zwei oder drei usw. Elemente mit der entsprechenden Eigenschaft existieren) wird nicht immer in den Anfrageergebnissen wiedergespiegelt.

Aus diesem Grund wird eine zweite Semantik entwickelt, die *GCWA*-Semantik*, die implizite Informationen und zudem auch logische Äquivalenz von Schema-Mappings und die Standard-Semantik von Quantoren der Logik erster

Stufe berücksichtigt. Zunächst werden Anfragesemantiken aus dem Bereich der deduktiven Datenbanken in den Kontext relationalen Datenaustauschs übertragen und in diesem Kontext untersucht. Inspiriert von diesen Semantiken wird dann die GCWA*-Semantik entwickelt. Unter der GCWA*-Semantik werden Anfragen durch die sicheren Antworten auf *GCWA*-Lösungen* beantwortet. Im Unterschied zu den vorhergehenden Semantiken und Lösungskonzepten sind die GCWA*-Semantik und GCWA*-Lösungen für *alle* Schema-Mappings definiert (nicht nur für solche, die durch tgds und egds spezifiziert werden). Für durch st-tgds und egds definierte Schema-Mappings sind GCWA*-Lösungen einfach Lösungen ohne Null-Werte, die Vereinigungen von inklusionsminimalen Lösungen sind.

3. *Bestimmung der Komplexität der Anfrageverarbeitung bezüglich der in 2. betrachteten Anfragesemantiken.*

Die Dissertation wendet sich in diesem Teil zuerst der Komplexität des Auswertungsproblems für die CWA-Lösungs-basierten Anfragesemantiken zu. Genauer wird die *Datenkomplexität* betrachtet, d.h. die Komplexität bzgl. eines *festen* Schema-Mappings und einer *festen* Anfrage. Es stellt sich heraus, dass zwei dieser Semantiken für konjunktive Anfragen und einige Erweiterungen davon genau die anfangs erwähnten sicheren Antworten liefern, so dass sich alle für die sicheren Antworten erhaltenen Resultate auf diese Semantiken übertragen. Bei ausreichender Einschränkung von Schema-Mappings liegt das Auswertungsproblem für Anfragen in der Logik erster Stufe – abhängig von der jeweiligen Semantik – in co-NP bzw. in NP. Außerdem existieren einfache Schema-Mappings und konjunktive Anfragen mit nur einer Ungleichung ($\neq$), bei denen dieses Problem vollständig für die jeweilige Klasse ist. Im Gegensatz dazu ist bekannt, dass die sicheren Antworten für solche Anfragen bzgl. einer Vielzahl von Schema-Mappings effizient berechnet werden können.

Ein größerer Teil der Dissertation beschäftigt sich mit der Komplexität des Auswertungsproblems bezüglich der GCWA*-Semantik. Wie oben wird auch hier die Datenkomplexität betrachtet. Es wird gezeigt, dass die GCWA*-Semantik auf monotonen Anfragen genau die anfangs erwähnten sicheren Antworten liefert, so dass sich alle für die sicheren Antworten erhaltenen Resultate für monotone Anfragen auf diese Semantiken übertragen. Allerdings gibt es einfache, durch st-tgds definierte Schema-Mappings M und einfache *existentielle* Anfragen q (Anfragen der Form $\exists \bar{x}\, \varphi$, wobei φ quantorenfrei ist), so dass das Auswertungsproblem

EVAL(M, q)	
Eingabe:	eine Quelldatenbank S für M und ein Tupel $\bar{t}$
Frage:	Ist $\bar{t}$ eine Antwort zu q bzgl. der GCWA*-Semantik?

co-NP-hart ist. Lässt man neben Existenzquantoren auch Allquantoren zu, so kann dieses Problem unentscheidbar werden. Umso überraschender scheint es, dass $\textsc{Eval}(M, q)$ für *universelle* Anfragen q (Anfragen der Form $\forall \bar{x}\, \varphi$, wobei φ quantorenfrei ist) bei passender Einschränkung von M in PTIME liegt.

Dieses Resultat soll zum Abschluss dieser Zusammenfassung noch etwas genauer beschrieben werden. Präziser besagt es, dass für jedes Schema-Mapping M, das durch spezielle st-tgds, so genannte *gepackte st-tgds*, beschrieben wird, und für jede universelle Anfrage q ein Polynomialzeitalgorithmus existiert, der bei Eingabe des Kerns der universellen Lösungen einer Quelldatenbank S die Antwortmenge zu q bzgl. M und S unter der GCWA*-Semantik ausgibt. Aus Resultaten zur Berechnung des Kerns der universellen Lösungen (z.B. von Fagin, Kolaitis und Popa) folgt insbesondere, dass $\textsc{Eval}(M, q)$ in PTIME liegt.

Um das Hauptresultat zu beweisen, reicht es aus, einen Polynomialzeitalgorithmus für das folgende Problem anzugeben: Gegeben der Kern der universellen Lösungen einer Quelldatenbank S und ein Tupel $\bar{t}$, liegt $\bar{t}$ in der Antwortmenge zu q bzgl. M und S unter der GCWA*-Semantik? Dazu wird das Problem zuerst auf den Test reduziert, ob eine Vereinigung von ein oder mehr inklusions-minimalen Datenbanken aus $\mathit{poss}(K)$ existiert, die $\neg q(\bar{t})$ erfüllt. Hierbei ist $\mathit{poss}(K)$ die Menge der Datenbanken, die sich aus dem Kern K der universellen Lösungen für S durch Ersetzen von Null-Werten durch Konstanten ergeben. Durch Transformation von $\neg q$ in eine Art disjunktive Normalform kann man sich dann o.B.d.A. auf den Fall beschränken, dass $\neg q$ äquivalent zu einer Formel $\bar{q} = \exists \bar{x}\, \varphi$ ist, wobei φ eine Konjunktion von atomaren und negierten atomaren Formeln ist. Besteht φ aus einer atomaren Formel und ist $\bar{x}$ das leere Tupel, so lässt sich das Problem wie folgt lösen. Zuerst wird die unendliche Menge der inklusions-minimalen Datenbanken in $\mathit{poss}(K)$ auf eine endliche Menge $\mathcal{S}$ möglicherweise exponentieller Größe reduziert. Der technisch aufwändigste Teil besteht dann darin, eine Teilmenge von $\mathcal{S}$ polynomieller Größe ausfindig zu machen, aus denen alle in inklusions-minimalen Lösungen vorkommenden Tupel rekonstruiert werden können. Um schließlich das allgemeine Problem zu lösen, setzt man dann Teillösungen in geeigneter Weise zu einer Gesamtlösung zusammen, wobei die für den einfachen Fall bewiesenen Resultate helfen, die Korrektheit dieser Konstruktion nachzuweisen.

Schlagwörter: Data Exchange, Sichere Antworten, Closed World Assumption (CWA), Deduktive Datenbank

Preface

This thesis presents most of the work I did on *foundations of query answering in relational data exchange* during the past four years. Parts of the results have been or are about to get published in journals, or proceedings of international conferences. In particular, parts of the material in Chapters 2 and 3 are based on Hernich and Schweikardt [2007, 2010]. Furthermore, Chapters 4 and 5 are based on Hernich [2010], which was presented at the *13th International Conference on Database Theory (ICDT 2010)* and received one of the two ICDT 2010 Best Student Paper Awards.

I assume basic familiarity with database theory, as it can be obtained, for example, from the textbook by Abiteboul et al. [1995]. Wherever possible, definitions are given directly preceding their first use.

This thesis would not have been possible without the support of many people. First of all, I owe a particular debt to my advisor, Nicole Schweikardt. I greatly benefited from many inspiring discussions with her, her patience and her encouragement throughout the last five years of my research and the writing of this thesis. The friendly and fruitful atmosphere in her working group led not only to results on this thesis' topic, but also to results on the topic of *read/write streams*, published in Grohe et al. [2009] and Hernich and Schweikardt [2008]. Second, I thank Phokion G. Kolaitis very much for serving as a reviewer for this thesis. When attending the *Logic and Databases* workshop at the Isaac Newton Institute in Cambridge, UK, in 2006, I had the opportunity to listen to interesting talks by Phokion G. Kolaitis and Georg Gottlob, which greatly influenced the direction of my research. I thank the organizers, Anuj Dawar and Martin Grohe, of this workshop for inviting me, and the Isaac Newton Institute for their financial support enabling me to attend the workshop. Finally, I am grateful to André Böhm, Dominik Freydenberger, Frederik Harwath and Lucas Heimberg for reading parts of this thesis and their helpful comments.

André Hernich
Frankfurt am Main, March 2010

Für meine Eltern.

Contents

1 Introduction

Data exchange deals with translating data structured in some format into data structured in some other format, according to a specification of the relationship between the source data and the target data. Such data translation tasks are very common in practice. They arise, for example, in data restructuring, in ETL (Extract-Transform-Load) processes which are used for updating data warehouses, or in data exchange between different, possibly independently created, applications (see, e.g., Haas et al. [2005], Fagin et al. [2005a]). Over the past decades, various systems that support data exchange have been implemented [Shu et al., 1977, Haas et al., 2005].

Research on the theoretical foundations of data exchange started only recently with the seminal articles of Fagin et al. [2005a,b]. These articles focused on *relational data exchange*, that is, data exchange for the case that the source data and the target data are relational (i.e., stored in relational databases). A large body of work in the data exchange literature is devoted to relational data exchange. However, data exchange based on other data models has been considered, too. The case of XML data, for example, has been studied by Arenas and Libkin [2008] and Amano et al. [2009]. Also, extensions of the basic setting of relational data exchange, where data is exchanged between multiple "parties", have been considered [Fuxman et al., 2006, De Giacomo et al., 2007]. Nevertheless, an important reason for studying relational data exchange is the fact that some of the interesting fundamental issues of data exchange—like issues related to query answering—arise already in the context of relational data exchange. To understand these issues, it seems to be a good idea to first study them in the basic setting of relational data exchange. For more background on data exchange, I refer the interested reader to Fagin et al. [2005a], Haas et al. [2005], or to the survey articles by Kolaitis [2005] and Barceló [2009].

One of the major issues in data exchange is *query answering*, which refers to the problem of answering queries that are posed against target data (see, e.g., Fagin et al. [2005a], Kolaitis [2005], Barceló [2009]). The main difficulty is that data translations are usually underspecified. In particular, there is usually more than one possibility to translate source data to the target, so that it is not a priori clear what the answer to a query should be. One of the fundamental goals in data exchange is to find appropriate semantics for query answering. This is one of the goals pursued in this thesis.

The remaining part of this introductory chapter is organized as follows. In Section 1.1, we review the central concepts of relational data exchange: *schema*

mappings and *solutions*. The presentation of these concepts is inspired by Section 2 in Hernich and Schweikardt [2010]. A survey of research topics in relational data exchange is given in Section 1.2. Section 1.3 explains the basics of query answering in relational data exchange. First, it motivates and defines the certain answers semantics, which seems to be the right semantics to answer a large number of queries that are frequently encountered in practice. Second, it explains why the certain answers semantics does not seem to be sufficient to answer more general queries. Section 1.4 summarizes the main contributions of this thesis, and Section 1.5 describes the structure of this thesis.

1.1 The Central Concepts of Relational Data Exchange

We begin by recalling some standard notions from database theory. A comprehensive introduction to database theory is the textbook by Abiteboul et al. [1995]. Our notation slightly deviates from the notation used in that book.

A *schema* is a finite set of relation symbols. Each relation symbol R has an associated positive integer $\mathrm{ar}(R)$, called its *arity*. Given a schema σ, an *instance* I over σ assigns to each relation symbol $R \in \sigma$ a finite $\mathrm{ar}(R)$-ary relation R^I. The *active domain* of I, denoted by $\mathrm{dom}(I)$, is the set of all elements that occur in the tuples of the relations of I, that is, $\mathrm{dom}(I)$ consists of all elements u for which there is a relation symbol $R \in \sigma$ and a tuple $(t_1, \ldots, t_{\mathrm{ar}(R)}) \in R^I$ with $u \in \{t_1, \ldots, t_{\mathrm{ar}(R)}\}$. We assume a fixed infinite set *Dom*, whose elements are called *values*, such that $\mathrm{dom}(I) \subseteq \mathit{Dom}$ for all instances I. The set of all instances over σ is denoted by $\mathrm{inst}(\sigma)$.

Let us also fix the following notations that are used over and over again throughout this thesis. Let $f\colon X \to Y$ be a mapping, where X and Y are arbitrary sets. Given a tuple $\bar{t} = (t_1, \ldots, t_n) \in X^n$, we extend f to $\bar{t}$ by applying f pointwise to $\bar{t}$, that is, $f(\bar{t}) := (f(t_1), \ldots, f(t_n))$. Given a relation $R \subseteq X^n$, we extend f to R by $f(R) := \{f(\bar{t}) \mid \bar{t} \in R\}$. In particular, $f(X)$ is the *range* of f (we often consider sets as unary relations). Finally, if I is an instance over a schema σ with $\mathrm{dom}(I) \subseteq X$, and if $Y \subseteq \mathit{Dom}$, we let $f(I)$ be the instance with $R^{f(I)} := f(R^I)$ for each $R \in \sigma$.

Given two schemas σ_{s} and σ_{t}, a specification of the relationship between instances over σ_{s} and instances over σ_{t}, and an instance S over σ_{s}, the goal in relational data exchange is to translate S (the *source instance*) into an instance T over σ_{t} (a *target instance*) that satisfies the given specification [Fagin et al., 2005a]. The following example describes one specific setting, and is used as a running example throughout the remainder of this section.

1.1 Example (Restructuring a library database)
Consider the scenario of restructuring a library database. Let us assume that

S is a source instance over the schema $\sigma_s = \{Books, Authors\}$, where $Books^S$ stores the books contained in the library as tuples of the form $(isbn, title)$, and $Authors^S$ stores the relation between books and authors as tuples of the form $(isbn, author\_name)$. The goal is to restructure S into a target instance T over the schema $\sigma_t = \{BookInfo, AuthorList, WrittenBy\}$, where $BookInfo^T$ stores the books in the library as tuples of the form $(isbn, title, genre)$, $AuthorList^T$ stores information on the authors as tuples of the form $(author\_id, name)$, and $WrittenBy^T$ stores the relation between books and authors as tuples of the form $(isbn, author\_id)$.

An additional requirement is that the attribute *author_id* of *AuthorList* is a key (i.e., each value for *author_id* implies a unique value for the attribute *name*), and that for every tuple $(isbn, author\_id)$ in $WrittenBy^T$, the relations $BookInfo^T$ and $AuthorList^T$ contain entries for the book with ISBN number *isbn*, and the author with author ID *author_id*, respectively. □

A formal framework for specifying requirements such as those described in Example 1.1 has been developed by Fagin et al. [2005a]. The central concepts of this framework are *schema mappings* and *solutions*.

1.1.1 Schema Mappings and Solutions

This section reviews the central concepts for specifying requirements such as those described in Example 1.1: *schema mappings* and *solutions*. These concepts were developed by Fagin et al. [2005a].

A schema mapping is a binary relation between instances over a source schema and instances over a target schema:

1.2 Definition (Schema mapping)
A *schema mapping* M is a subset of $\text{inst}(\sigma_s) \times \text{inst}(\sigma_t)$, where σ_s and σ_t are schemas. We refer to σ_s as the *source schema* of M, and call an instance over σ_s a *source instance* for M. Analogously, we refer to σ_t as the *target schema* of M, and call an instance over σ_t a *target instance* for M.

This definition is essentially the definition of schema mappings used by Arenas et al. [2009b]. Schema mappings as defined by Fagin et al. [2005a] correspond to *logically defined* schema mappings, which are the topic of Section 1.1.2.

The target instances to which a source instance S is associated by a schema mapping M are called *solutions* (for S under M), and—recalling the goal stated at the beginning of this section—solutions correspond to the target instances to which S can be translated, given M:

1.3 Definition (Solution)
Let M be a schema mapping, and let S be a source instance for M. A *solution*

for S under M is a target instance T for M with $(S, T) \in M$. The set of all solutions for S under M is denoted by $sol(M, S)$.

Note that, despite of their name, schema mappings are not mappings in the mathematical sense. That is, given a schema mapping M, it is in general not the case that for every source instance S for M there is precisely one solution for S under M.

1.4 Example (Library database restructuring, continued)
The requirements from Example 1.1 can be represented by a schema mapping M that consists of all pairs $(S, T) \in \text{inst}(\sigma_\text{s}) \times \text{inst}(\sigma_\text{t})$, where σ_s and σ_t are the schemas from Example 1.1, and:

1. For every tuple $(isbn, title) \in Books^S$, there is some value *genre* such that $(isbn, title, genre) \in BookInfo^T$.

2. For every tuple $(isbn, name) \in Authors^S$, there is some value *author_id* such that $(author\_id, name)$ belongs to $AuthorList^T$ and $(isbn, author\_id)$ belongs to $WrittenBy^T$.

3. For every $(author\_id, name) \in AuthorList^T$ and every $(isbn, author\_id) \in WrittenBy^T$, we have $(isbn, name) \in Authors^S$.

4. If $(author\_id, name)$ and $(author\_id, name')$ belong to $AuthorList^T$, then $name = name'$.

5. For every $(isbn, author\_id) \in WrittenBy^T$, there are values *title*, *genre* and *name* such that $(isbn, title, genre) \in BookInfo^T$ and $(author\_id, name) \in AuthorList^T$.

For example, let S^* be the source instance with

$$\begin{aligned} Books^{S^*} &= \{(\text{“0-201-53771-0”}, \text{“Foundations of Databases”})\}, \\ Authors^{S^*} &= \{(\text{“0-201-53771-0”}, \text{“Serge Abiteboul”}), \\ &\quad (\text{“0-201-53771-0”}, \text{“Richard Hull”}), \\ &\quad (\text{“0-201-53771-0”}, \text{“Victor Vianu”}), \\ &\quad (\text{“0-201-53082-1”}, \text{“Christos H. Papadimitriou”})\}. \end{aligned}$$

Then the target instance T for M with

$$\begin{aligned} BookInfo^T &= \{(\text{“0-201-53771-0”}, \text{“Foundations of Databases”}, \text{“DB”}), \\ &\quad (\text{“0-201-53082-1”}, \text{“Computational Complexity”}, \text{“CC”})\}, \end{aligned}$$

$$\begin{aligned} \mathit{AuthorList}^T &= \{(1,\ \text{“Serge Abiteboul”}),\ (2,\ \text{“Richard Hull”}),\\ &\qquad (3,\ \text{“Victor Vianu”}),\ (4,\ \text{“Christos H. Papadimitriou”})\},\\ \mathit{WrittenBy}^T &= \{(\text{“0-201-53771-0”},\ 1),\ (\text{“0-201-53771-0”},\ 2),\\ &\qquad (\text{“0-201-53771-0”},\ 3),\ (\text{“0-201-53082-1”},\ 4)\} \end{aligned}$$

is a solution for S^* under M. Furthermore, every target instance T' that is obtained from T by replacing the values “Computational Complexity”, “DB”, “CC”, 1, 2, 3 and 4 by other values leads to a solution for S^* under M, provided the replacements for 1, 2, 3 and 4 are pairwise distinct. On the other hand, the target instance that is obtained from T by removing, say, the first tuple in $\mathit{BookInfo}^T$ is no solution for S^* under M. □

This is a natural point to introduce the concept of *labeled null values*. In relational data exchange, *labeled null values* (*nulls*, for short) are used as placeholders for *unknown* values in target instances (i.e., values for which the schema mapping just tells us that there is a value, but does not tell which particular one).

1.5 Example (Library database restructuring, continued)
In Example 1.4, one could as well use nulls $\perp_1$, $\perp_2$ etc. to represent the unknown book titles, book genres, and author IDs in a solution T^* for S^* under M:

$$\begin{aligned} \mathit{BookInfo}^{T^*} &= \{(\text{“0-201-53771-0”},\ \text{“Foundations of Databases”},\ \perp_1),\\ &\qquad (\text{“0-201-53082-1”},\ \perp_2,\ \perp_3)\},\\ \mathit{AuthorList}^{T^*} &= \{(\perp_4,\ \text{“Serge Abiteboul”}),\ (\perp_5,\ \text{“Richard Hull”}),\\ &\qquad (\perp_6,\ \text{“Victor Vianu”}),\ (\perp_7,\ \text{“Christos H. Papadimitriou”})\},\\ \mathit{WrittenBy}^{T^*} &= \{(\text{“0-201-53771-0”},\ \perp_4),\ (\text{“0-201-53771-0”},\ \perp_5),\\ &\qquad (\text{“0-201-53771-0”},\ \perp_6),\ (\text{“0-201-53082-1”},\ \perp_7)\}. \end{aligned}$$ □

Formally, we assume that the set *Dom* is the union of two disjoint infinite sets, *Const* and *Null*. The values in *Const* are called *constants* and correspond to the usual database values. The values in *Null* are the *nulls*. We use the letters a, b, c (possibly with subscripts and/or superscripts) to denote constants, and the symbol $\perp$ (possibly with subscripts and/or superscripts) to denote nulls. For an instance I, let $\text{const}(I) := \text{dom}(I) \cap \mathit{Const}$ and $\text{nulls}(I) := \text{dom}(I) \cap \mathit{Null}$.

1.6 Remark (Instances without nulls vs. instances with nulls)
From a conceptual point of view, the fact that instances may contain nulls seems to be unsatisfying. After all, there is a huge difference between instances without nulls (which correspond to instances as usually encountered in the database literature) and instances with nulls. While an instance I *without*

nulls is viewed as a collection of the relations that constitute I, an instance *with* nulls should be viewed as a representation of a set of instances without nulls [Abiteboul et al., 1995]. Treating instances with nulls in the same way as instances without nulls can lead to counter-intuitive results in query answering. We will come back to this issue in Section 1.3.

Nevertheless, to be consistent with the data exchange literature, we allow nulls to occur in instances. We should, however, bear in mind the difference between instances without nulls and instances with nulls. □

An instance without nulls is called *ground instance*. In the database literature, an instance that may contain nulls is also called *naive table* (see, e.g., Abiteboul et al. [1995]). In this thesis, we call an instance I *naive table* if we want to emphasize that I may contain nulls.

Throughout this thesis, we make the following assumptions which are common in relational data exchange (cf., e.g., Fagin et al. [2005a,b], Kolaitis [2005], Barceló [2009]):

1.7 Proviso (Source instances and target instances)
Source instances for schema mappings are *ground instances* unless stated otherwise. *Target instances* for schema mappings are *naive tables*.

Before we deal with the question of how schema mappings can be specified, let us introduce a technical, but natural, restriction on schema mappings. Intuitively, we would like schema mappings to depend on a finite number of constants only. In other words, it should be possible to associate each schema mapping M with a finite set C of constants, so that M is invariant under renamings of values in $Dom \setminus C$. Formally, this is captured by the notion of a *generic* schema mapping. A schema mapping M with source schema σ_{s} and target schema σ_{t} is called *C-generic* for a set $C \subseteq Const$ if and only if for all $(S, T) \in M$, and for all bijective mappings $\pi\colon Dom \to Dom$, where $\pi(c) = c$ for all $c \in C$, we have $(\pi(S), \pi(T)) \in M$:

$$\begin{array}{ccc} S & \xrightarrow{\;M\;} & T \\ {\scriptstyle\pi}\downarrow & & \downarrow{\scriptstyle\pi} \\ \pi(S) & \xrightarrow{\;M\;} & \pi(T) \end{array}$$

We call M *generic* if and only if there is a *finite* set $C \subseteq Const$ such that M is C-generic

1.8 Proviso (Schema mappings)
We consider *generic* schema mappings, unless stated otherwise.

Note that genericity implies that the precise choice of nulls in solutions does not matter. Intuitively, this is a desired property: For example, if we rename each null $\bot_i$ in the instance T^* from Example 1.5 to a null $\bot'_i$ (so that for distinct $i, j \in \{1, \ldots, 7\}$ we have $\bot'_i \neq \bot'_j$), then the resulting instance should be a solution for S^* under M, too, since nulls are just placeholders for constants. The property that two instances are the same up to renaming of nulls is formalized by *isomorphisms*:

1.9 Definition (Isomorphism, isomorphic)
Let I and J be instances over a schema σ.

- An *isomorphism* from I to J is a bijective mapping $f\colon \mathrm{dom}(I) \to \mathrm{dom}(J)$ such that $f(I) = J$, $f(c) = c$ for each constant $c \in \mathrm{const}(I)$, and $f(\bot)$ is a null for each null $\bot \in \mathrm{nulls}(I)$.
- We say that I and J are *isomorphic*, and we write $I \cong J$, if and only if there is an isomorphism from I to J.

In the following, we often do not distinguish between *isomorphic* instances. That is, given isomorphic instances I and J, we often view I and J as one and the same instance.

1.1.2 Logically Defined Schema Mappings

In relational data exchange, one typically considers *sets of formulas* of a certain logic for specifying schema mappings in a high-level declarative way. The sets of formulas considered in the data exchange literature for schema mapping specification consist of sentences of some small fragment of first-order logic (FO), or an extension thereof.

A *FO formula* over a schema σ is a FO formula that can refer to the relation symbols in σ and the constants in *Const*. More precisely, a FO formula over σ is built from atomic formulas of the form

- $R(\bar{t})$, where $R \in \sigma$, and $\bar{t}$ is an $\mathrm{ar}(R)$-tuple over $\mathit{Var} \cup \mathit{Const}$ (where *Var* is some fixed infinite set of variables disjoint to *Dom*), and
- $t_1 = t_2$, where t_1 and t_2 are elements of $\mathit{Var} \cup \mathit{Const}$,

using negation ($\neg$), disjunction ($\vee$), conjunction ($\wedge$), existential quantification ($\exists$), and universal quantification ($\forall$). We omit parentheses whenever this does not introduce any ambiguities, and use $\varphi_1 \to \varphi_2$ and $\varphi_1 \leftrightarrow \varphi_2$ as abbreviations for $\neg\varphi_1 \vee \varphi_2$, and $(\varphi_1 \wedge \varphi_2) \vee (\neg\varphi_1 \wedge \neg\varphi_2)$, respectively. The set of the *free variables* of a FO formula φ, denoted by $\mathrm{free}(\varphi)$, is defined as usual. The

notation $\varphi(x_1, \ldots, x_k)$ indicates that φ is a formula with $\text{free}(\varphi) = \{x_1, \ldots, x_k\}$, where $x_1, \ldots, x_k$ are pairwise distinct variables. A *sentence* is a formula without free variables.

For evaluating FO formulas in an instance I, the *natural semantics* is used, that is, quantified variables range over *Dom*, and each constant in *Const* is interpreted by itself. More precisely, let an *assignment for a set* $X \subseteq \mathit{Var}$ be a mapping $\alpha\colon X' \to \mathit{Dom}$, where $X' \subseteq \mathit{Var}$ and $X \subseteq X'$. An *assignment for a FO formula* φ is an assignment for $\text{free}(\varphi)$. For every instance I over a schema σ, every FO formula φ over σ, and every assignment α for φ, we define the satisfaction relation $I \models \varphi(\alpha)$ (in words: "I satisfies φ under α") in the usual way. In particular:

- If φ has the form $R(t_1, \ldots, t_{\text{ar}(R)})$, where $R \in \sigma$ and $(t_1, \ldots, t_{\text{ar}(R)})$ is a tuple over $\mathit{Var} \cup \mathit{Const}$, then $I \models \varphi(\alpha)$ if and only if $(\alpha^*(t_1), \ldots, \alpha^*(t_{\text{ar}(R)})) \in R^I$, where for each $i \in \{1, \ldots, \text{ar}(R)\}$, we have $\alpha^*(t_i) := \alpha(t_i)$ if $t_i \in \mathit{Var}$, and $\alpha^*(t_i) := t_i$ otherwise.

- If φ has the form $t_1 = t_2$, where t_1 and t_2 are elements of $\mathit{Var} \cup \mathit{Const}$, then $I \models \varphi(\alpha)$ if and only if $\alpha^*(t_1) = \alpha^*(t_2)$, where α^* is defined in an analogous way as above.

- If φ has the form $\exists x\, \psi$, where $x \in \mathit{Var}$ and ψ is a FO formula over σ, then $I \models \varphi(\alpha)$ if and only if there is some $u \in \mathit{Dom}$ such that $I \models \psi(\alpha\frac{u}{x})$, where $\alpha\frac{u}{x}$ is the assignment for ψ such that for all $x' \in X \cup \{x\}$, where X is the set of variables in α's domain:

$$\alpha\tfrac{u}{x}(x') \;:=\; \begin{cases} u, & \text{if } x' = x, \\ \alpha(x'), & \text{otherwise.} \end{cases}$$

- If φ has the form $\forall x\, \psi$, where $x \in \mathit{Var}$ and ψ is a FO formula over σ, then $I \models \varphi(\alpha)$ if and only if for all $u \in \mathit{Dom}$ we have $I \models \psi(\alpha\frac{u}{x})$, where $\alpha\frac{u}{x}$ is defined as above.

An assignment α for a formula $\varphi(x_1, \ldots, x_k)$ is often referred to by the tuple $(\alpha(x_1), \ldots, \alpha(x_k))$. So, for a FO formula $\varphi(x_1, \ldots, x_k)$ and a tuple $\bar{u} = (u_1, \ldots, u_k) \in \mathit{Dom}^k$, $I \models \varphi(\bar{u})$ is an abbreviation for $I \models \varphi(\alpha)$, where α maps each x_i to u_i. If φ is a sentence, we omit the assignment. That is, we just write $I \models \varphi$ instead of $I \models \varphi(\varepsilon)$, where ε is the empty tuple. For a set Φ of FO sentences over σ, we write $I \models \Phi$ if and only if for all $\varphi \in \Phi$ we have $I \models \varphi$.

1.10 Remark (Why nulls are prohibited in formulas)
Although both, constants and nulls, may occur in instances, formulas can refer

only to constants. The reason is the intended semantics of nulls. Recall that a null serves as a placeholder for an (unknown) constant. In particular, a null has no definite value. □

In later chapters, we also deal with the logic $L_{\infty\omega}$, which is the extension of FO logic, where disjunctions and conjunctions may range over an infinite number of formulas, instead of just two formulas. The semantics of such infinite disjunctions and conjunctions is a straightforward extension of the semantics of their binary counterparts. A precise definition of $L_{\infty\omega}$ formulas and their semantics is given directly preceding their first use. For the moment, the above informal description should suffice.

Note that the number of assignments satisfying a given formula can be infinite, and variables can be assigned to arbitrary elements of *Dom*. In the following, we restrict attention to formulas φ such that satisfaction of φ in an instance I depends only on $\mathrm{dom}(I)$ and $\mathrm{dom}(\varphi)$, where $\mathrm{dom}(\varphi)$ is the set of all constants that occur in φ. More precisely, we consider formulas that are *domain independent* in the following sense. Given an instance I, a set $D \subseteq \mathit{Dom}$, a formula φ, and an assignment α for φ, we define $I \models_D \varphi(\alpha)$ in the same way as $I \models \varphi(\alpha)$ above, except that the quantifiers in φ range over $\mathrm{dom}(I) \cup \mathrm{dom}(\varphi) \cup D$ instead of *Dom*. A formula φ is *domain independent* if and only if for all instances I, all sets $D \subseteq \mathit{Dom}$, and all assignments α for φ,

$$I \models \varphi(\alpha) \iff I \models_D \varphi(\alpha), \text{ and } \alpha(x) \in \mathrm{dom}(I) \cup \mathrm{dom}(\varphi) \cup D \text{ for all } x \in \mathrm{free}(\varphi).$$

In particular, it suffices to consider only assignments with range in $\mathrm{dom}(I) \cup \mathrm{dom}(\varphi)$. For syntactic restrictions of formulas that ensure domain independence, we refer the interested reader to Abiteboul et al. [1995].

1.11 Proviso (Formulas)
We consider *domain independent* formulas, unless stated otherwise.

Sets of FO sentences or $L_{\infty\omega}$ sentences specify schema mappings as follows:

1.12 Definition (Logically defined schema mapping)
Given *disjoint*[1] schemas σ_{s} and σ_{t}, and a *finite* set Σ of FO sentences (resp., $L_{\infty\omega}$ sentences) over $\sigma_{\mathrm{s}} \cup \sigma_{\mathrm{t}}$, we let $(\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$ be the schema mapping

$$\{(S, T) \in \mathrm{inst}(\sigma_{\mathrm{s}}) \times \mathrm{inst}(\sigma_{\mathrm{t}}) \mid S \cup T \models \Sigma\}.$$

Here, $S \cup T$ denotes the instance over $\sigma_{\mathrm{s}} \cup \sigma_{\mathrm{t}}$ such that for all $R \in \sigma_{\mathrm{s}}$ we have $R^{S \cup T} = R^S$, and for all $R \in \sigma_{\mathrm{t}}$ we have $R^{S \cup T} = R^T$.

[1] The restriction to disjoint source schemas and target schemas is necessary in order to distinguish between source and target in FO sentences, or $L_{\infty\omega}$ sentences, respectively.

Note that all schema mappings defined by sets of FO sentences are generic. Moreover, all schema mappings defined by sets of $L_{\infty\omega}$ sentences containing a finite number of constants are generic.

1.13 Example (Library database restructuring, continued)
For the schema mapping M from Example 1.4, we have $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$, where Σ contains the following FO sentences over $\sigma_{\mathrm{s}} \cup \sigma_{\mathrm{t}}$:

$$\begin{aligned}
\chi_1 &:= \forall x_1 \forall x_2 \Big(\mathit{Books}(x_1, x_2) \rightarrow \exists z\, \mathit{BookInfo}(x_1, x_2, z) \Big), \\
\chi_2 &:= \forall x_1 \forall x_2 \Big(\mathit{Authors}(x_1, x_2) \leftrightarrow \exists z (\mathit{AuthorList}(z, x_2) \wedge \mathit{WrittenBy}(x_1, z)) \Big), \\
\chi_3 &:= \forall x_1 \forall x_2 \forall x_3 \Big(\mathit{AuthorList}(x_1, x_2) \wedge \mathit{AuthorList}(x_1, x_3) \rightarrow x_2 = x_3 \Big), \\
\chi_4 &:= \forall x_1 \forall x_2 \Big(\mathit{WrittenBy}(x_1, x_2) \rightarrow \\
&\qquad\qquad \exists z_1 \exists z_2 \exists z_3 (\mathit{BookInfo}(x_1, z_1, z_2) \wedge \mathit{AuthorList}(x_2, z_3)) \Big).
\end{aligned}$$

Note that χ_1 expresses condition 1 in Example 1.4, χ_2 expresses conditions 2 and 3 in Example 1.4, χ_3 expresses condition 4 in Example 1.4, and χ_4 expresses condition 5 in Example 1.4. □

In practice, one has to be careful in the choice of the sentences used for specifying schema mappings. The following example exhibits a schema mapping M defined by a *single* FO sentence such that the problem

Existence-of-Solutions(M)	
Input:	a source instance S for M
Question:	Is there a solution for S under M?

is undecidable. This example is based on [Kolaitis et al., 2006, Theorem 3.6]. A different example based on the halting problem for Turing machines is sketched in Kolaitis [2005], and an example based on Post's correspondence problem is sketched in Hernich and Schweikardt [2010]. Note that the difficulty in proving the undecidability of Existence-of-Solutions(M) is that M is fixed. If M would be part of the input, undecidability would easily follow from Trakhtenbrot's Theorem (see, e.g., Ebbinghaus and Flum [1999], Libkin [2004]).

1.14 Example (Embedding problem for finite semigroups)
Consider the embedding problem for finite semigroups, which is defined as follows:

EMBEDDING PROBLEM FOR FINITE SEMIGROUPS

Input: a partial function $p\colon X^2 \to X$, where X is a finite set

Question: Is there a finite set $Y \supseteq X$ and a total function $f\colon Y^2 \to Y$ such that f is associative, and f extends p (i.e., if $p(x,y)$ is defined, then $f(x,y) = p(x,y)$)?

This problem was shown to be undecidable in Kolaitis et al. [2006].

An input $p\colon X^2 \to X$ to this problem can be represented by an instance S_p over the schema $\sigma_{\mathrm{s}} := \{R\}$, where $R^{S_p} = \{(x,y,z) \in X^3 \mid p(x,y) = z\}$. A solution $f\colon Y^2 \to Y$ can be represented by an instance T_f over the schema $\sigma_{\mathrm{t}} := \{\tilde{R}\}$, where $\tilde{R}^{T_f} = \{(x,y,z) \in Y^3 \mid f(x,y) = z\}$.

It is now easy to construct a FO sentence φ over $\sigma_{\mathrm{s}} \cup \sigma_{\mathrm{t}}$ such that for a given partial function $p\colon X^2 \to X$, there is a solution for S_p under the schema mapping $M = (\{R\}, \{\tilde{R}\}, \{\varphi\})$ if and only if there is a finite set $Y \supseteq X$ and a total function $f\colon Y^2 \to Y$ that is associative and extends p. Indeed, we can choose

$$\begin{aligned}
\varphi \;:=\; & \underbrace{\forall x \forall y \forall z \Big(R(x,y,z) \to \tilde{R}(x,y,z) \Big)}_{\text{“}R \subseteq \tilde{R}\text{”}} \\
& \wedge \underbrace{\forall x \forall y \forall z \forall z' \Big(\tilde{R}(x,y,z) \wedge \tilde{R}(x,y,z') \to z = z' \Big)}_{\text{“}\tilde{R} \text{ is the graph of a function } f\text{”}} \\
& \wedge \underbrace{\forall x \forall y \Big(\psi_{\tilde{R}}(x) \wedge \psi_{\tilde{R}}(y) \to \exists z \tilde{R}(x,y,z) \Big)}_{\text{“}f \text{ is a total function”}} \\
& \wedge \underbrace{\forall x \forall y \forall z \forall u \forall v \forall w (\tilde{R}(x,y,u) \wedge \tilde{R}(u,z,v) \wedge \tilde{R}(y,z,w) \to \tilde{R}(x,w,v))}_{\text{“}f \text{ is associative”}},
\end{aligned}$$

where

$$\psi_{\tilde{R}}(x) \;:=\; \exists x_1 \exists x_2 \exists x_3 \Big(\tilde{R}(x_1,x_2,x_3) \wedge \bigvee_{i=1}^{3} x = x_i \Big)$$

defines the set of values contained in $\tilde{R}$. □

One therefore considers less powerful, but still sufficiently expressive, fragments of FO logic to define schema mappings, such as the following types of FO sentences:

- *Source-to-target tuple-generating dependencies (st-tgds).* These are FO sentences of the form

$$\forall \bar{x} \forall \bar{y} (\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})),$$

where φ is a conjunction of relational atomic FO formulas (i.e., conjunctions of atomic FO formulas of the form $R(\bar{t})$) over the source schema, and ψ is a conjunction of relational atomic FO formulas over the target schema.

- *Target tuple-generating dependencies (t-tgds).* These are FO sentences of the form

$$\forall \bar{x} \forall \bar{y} (\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z})),$$

where φ and ψ are conjunctions of relational atomic FO formulas over the target schema.

- *Equality-generating dependencies (egds).* These are FO sentences of the form

$$\forall \bar{x} (\varphi(\bar{x}) \rightarrow x_i = x_j),$$

where φ is a conjunction of relational atomic FO formulas over the target schema, $\bar{x} = (x_1, \ldots, x_k)$ for some integer $k \geq 2$, and $i, j \in \{1, \ldots, k\}$.

By a *tgd* we mean a st-tgd or a t-tgd. A *full tgd* is a tgd without existentially quantified variables $\bar{z}$.

For *algorithmic* results, we restrict our attention to schema mappings defined by tgds and egds. On the other hand, for non-algorithmic results like query answering semantics, we consider schema mappings that are as general as possible.

In Example 1.13, χ_1 is a st-tgd, χ_3 is an egd, χ_4 is a t-tgd. However, χ_2 is neither a tgd nor an egd. A reformulation of the schema mapping from Example 1.13 using tgds and egds is given in the following example:

1.15 Example (Library database restructuring, continued)
The schema mapping from Example 1.13 (more precisely, an approximation to that schema mapping) can be specified as $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$, where Σ consists of:

$$\begin{aligned}
\chi_1 &= \forall x_1 \forall x_2 \Big(\mathit{Books}(x_1, x_2) \rightarrow \exists z\, \mathit{BookInfo}(x_1, x_2, z) \Big), \\
\chi_2' &= \forall x_1 \forall x_2 \Big(\mathit{Authors}(x_1, x_2) \rightarrow \exists z (\mathit{AuthorList}(z, x_2) \wedge \mathit{WrittenBy}(x_1, z)) \Big), \\
\chi_3 &= \forall x_1 \forall x_2 \forall x_3 \Big(\mathit{AuthorList}(x_1, x_2) \wedge \mathit{AuthorList}(x_1, x_3) \rightarrow x_2 = x_3 \Big), \\
\chi_4 &= \forall x_1 \forall x_2 \Big(\mathit{WrittenBy}(x_1, x_2) \rightarrow \\
&\qquad \exists z_1 \exists z_2 \exists z_3 (\mathit{BookInfo}(x_1, z_1, z_2) \wedge \mathit{AuthorList}(x_2, z_3)) \Big). \quad \square
\end{aligned}$$

Note that, given a source instance S for the schema mapping M from Example 1.15, every solution for S under M is a solution for S under the schema mapping from Example 1.13, but not vice versa. Usually, one does not intend to fully specify a schema mapping, as in Example 1.13. One rather specifies only certain properties of solutions, as in Example 1.15, and tries to compute a "good" solution that reflects the source instance as accurately as possible.

The question of which solutions are "good" solutions is of fundamental interest in relational data exchange. A desirable property of such "good" solutions is that they can be computed *efficiently* on important classes of schema mappings. This means that for *fixed* schema mappings M from such a class, there is a polynomial time algorithm that takes a source instance for M as input, and outputs a "good" solution if it exists. The assumption that the schema mapping is fixed is typical in data exchange (see, e.g., Fagin et al. [2005a,b], Kolaitis [2005], Barceló [2009], Hernich and Schweikardt [2010]), though there are exceptions such as, for example, Kolaitis et al. [2006]. In this thesis, we also make this assumption. Another desirable property of "good" solutions is that the result of a query against the target can be computed *from the solution*, without referring to the schema mapping or the source instance. This property is motivated by the basic assumption underlying (relational) data exchange that the schema mapping and the source data may not be available once the data translation has been performed (see, e.g., Fagin et al. [2005a]). Furthermore, the result of a query should intuitively not depend on a particular materialized solution. The importance of this property becomes clearer when we deal with the issue of how to answer queries in Section 1.3.

One particular kind of "good" solutions are the *universal solutions* introduced by Fagin et al. [2005a], which, intuitively, are "most general" solutions, and are studied in depth in Chapter 2 (strictly speaking, universal solutions are "good" with respect to properly restricted schema mappings specified by tgds and egds, and with respect to a large class of queries including unions of conjunctive queries). The solution T^* from Example 1.5 is a universal solution for the source instance S^* under the schema mapping M from Example 1.15.

1.2 Research Topics in Relational Data Exchange

Identifying classes of solutions that are "good" in the sense described at the end of Section 1.1 is certainly a fundamental research topic in relational data exchange. Several notions of potential "good" solutions have been proposed in the literature [Fagin et al., 2005a, Libkin, 2006, Hernich and Schweikardt, 2007, Libkin and Sirangelo, 2008, Afrati and Kolaitis, 2008]. We already mentioned the *universal solutions* from Fagin et al. [2005a], and we will present the other

solution concepts throughout this thesis, however, with a different focus. Hernich and Schweikardt [2010] surveys results on the question of which solutions are "good" solutions.

Results concerning the complexity of computing various kinds of solutions have been obtained in the literature [Kolaitis et al., 2006, Fagin et al., 2005a,b, Hernich and Schweikardt, 2007, Gottlob and Nash, 2008]. Some of these results will be mentioned in Chapter 2.

One topic that has gained a lot of attention in the past few years is the semantics of operators for manipulating schema mappings. In particular, *composition* [Fagin et al., 2005c, Libkin and Sirangelo, 2008] and *inversion* [Fagin, 2007, Fagin et al., 2008b, Arenas et al., 2009b, Fagin et al., 2009] of schema mappings have been considered. These operators are important in relational data exchange for composing specifications of schema mappings, or undoing the effect of data exchange. Moreover, they are basic components of the *model management* framework introduced by Bernstein [2003] to deal with meta-data problems. Besides the semantics of schema mapping composition and inversion, the following questions have been addressed by the articles cited above:

- Which logical languages are needed to define the composition or the inverse of schema mappings as a new schema mapping?
- What is the complexity of computing the composition or the inverse of schema mappings?

First steps towards a theory of schema mapping *optimization* (e.g., minimization)—a topic that is closely related to schema mapping composition and inversion—were made by Fagin et al. [2008a]. In particular, the authors studied various notions of equivalences between schema mappings.

Structural properties of schema mappings are the focus of Arenas et al. [2004] and ten Cate and Kolaitis [2009]. In Arenas et al. [2004], structural properties are used to show that certain queries cannot be answered over some "nice" universal solution (using the certain answers semantics of Fagin et al. [2005a], to be introduced in the next section). In ten Cate and Kolaitis [2009], languages for specifying schema mappings (such as the set of all st-tgds) are characterized in terms of structural properties of schema mappings.

A large part of the literature on relational data exchange deals with *query answering*. Some of these articles [Fagin et al., 2005a, Mądry, 2005, Arenas et al., 2004, 2009a] focus on the basic certain answers semantics by Fagin et al. [2005a], while others [Fagin et al., 2005b, Libkin, 2006, Hernich and Schweikardt, 2007, Libkin and Sirangelo, 2008, Afrati and Kolaitis, 2008, Hernich, 2010] focus on alternative query answering semantics. Let us now turn to the question of how to answer queries in relational data exchange.

1.3 How to Answer Queries in Relational Data Exchange?

One of the major issues in data exchange is query answering, which refers to the problem of answering queries that are posed against target data (see Fagin et al. [2005a]). Before going into the details of query answering in the context of relational data exchange, let us review some basic definitions regarding queries. A comprehensive treatment of queries can be found in Abiteboul et al. [1995].

A *query* over a schema σ is a mapping $q\colon \text{inst}(\sigma) \to \mathit{Dom}^k$, where k is a nonnegative integer, and $q(I)$ is finite for every instance I over σ. We also say that q is a *k-ary query over σ*, and call $\text{ar}(q) := k$ the *arity* of q. The set $q(I)$ is called the *result of q on I*. Answering (or evaluating) q on an instance I over σ means to compute $q(I)$.

In a similar way as for schema mappings, we restrict attention to *generic* queries. A query q over a schema σ is called *C-generic* for some set $C \subseteq \mathit{Const}$ if and only if for all instances I over σ, and for all bijective mappings $\pi\colon \mathit{Dom} \to \mathit{Dom}$, where $\pi(c) = c$ for all $c \in C$, we have $q(\pi(I)) = \pi(q(I))$:

$$\begin{array}{ccc} I & \xrightarrow{\pi} & \pi(I) \\ {\scriptstyle q}\downarrow & & \downarrow{\scriptstyle q} \\ q(I) & \xrightarrow{\pi} & \pi(q(I)) \end{array}$$

A query q over a schema σ is called *generic* if there is a *finite* set $C \subseteq \mathit{Const}$ such that q is C-generic.

1.16 Proviso (Queries)
We consider *generic* queries, unless stated otherwise.

We will almost always deal with queries defined by logical formulas. Given a formula φ (e.g., a FO formula, or a $L_{\infty\omega}$ formula) over a schema σ, and a tuple $\bar{x} = (x_1, \ldots, x_k)$ of pairwise distinct variables with $\text{free}(\varphi) = \{x_1, \ldots, x_k\}$, let $q_{\varphi,\bar{x}}$ be the k-ary query over σ with

$$q_{\varphi,\bar{x}}(I) \;:=\; \{(\alpha(x_1), \ldots, \alpha(x_k)) \mid \alpha \text{ is an assignment for } \varphi \text{ with } I \models \varphi(\alpha)\}$$

for every instance I over σ. Since we consider only domain-independent formulas φ, we have

$$q_{\varphi,\bar{x}}(I) \;=\; \{(\alpha(x_1), \ldots, \alpha(x_k)) \mid \alpha\colon \text{free}(\varphi) \to \text{dom}(I) \cup \text{dom}(\varphi),\, I \models \varphi(\alpha)\}.$$

We often identify $q_{\varphi,\bar{x}}$ with the formula φ, assuming that the ordering of the free variables $x_1, \ldots, x_k$ is clear from the context or does not matter. In this

case, we also write $\varphi(I)$ instead of $q_{\varphi,\bar{x}}(I)$. We write $\varphi(x_1,\ldots,x_k)$ to make the ordering $x_1,\ldots,x_k$ explicit. For example,

$$\varphi(x_1,x_2) \;:=\; \exists z\Big(E(x_1,z)\wedge E(z,x_2)\Big) \tag{1.1}$$

denotes the query $q_{\varphi,(x_1,x_2)}$ such that for every instance I over $\{E\}$,

$$\varphi(I) \;=\; q_{\varphi,(x_1,x_2)}(I) \;=\; \{(v_1,v_2)\in \mathrm{dom}(I)^2 \mid \text{there is some } w\in\mathrm{dom}(I) \text{ with } (v_1,w)\in E^I \text{ and } (w,v_2)\in E^I\}.$$

A *FO query* is a query $q_{\varphi,\bar{x}}$, where $\varphi(\bar{x})$ is a FO formula. For example, the query (1.1) is a FO query. It can be shown that all FO queries are generic. Moreover, all queries defined by $L_{\infty\omega}$ formulas containing a finite number of constants are generic.

1.17 Remark (Query languages)
A number of query languages of different expressive power have been considered in the database literature (see, e.g., Abiteboul et al. [1995]). I assume that the reader is familiar with the most basic query languages (such as *(unions of) conjunctive queries* and *Datalog queries*) and the complexity of evaluating such queries on an instance. If we give a definition of a certain type of query, we will do this just to fix a common notation. We will often tacitly use the fact that the *data complexity* of FO queries (in particular, unions of conjunctive queries) and of Datalog queries is in PTIME. That is, given a query q that is a FO query or a Datalog query, there is a polynomial time algorithm that takes an instance I as input and outputs $q(I)$. □

In the context of relational data exchange, query answering refers to the following problem: we are given a schema mapping M, a source instance S for M, and a query q over M's target schema, and the task is to answer q with respect to M and S. The key issues are:

1. Typically, there is more than one solution for S under M, so that it is not a priori clear what the answer to q with respect to M and S should be. This is illustrated by Example 1.18 below.

2. We cannot assume that M and S are available after the data exchange has been performed. Therefore, we have to identify a notion of "good" solutions that can be computed in order to obtain the answers to queries directly from that solution.

The following example illustrates the first issue.

1.18 Example (Library database restructuring, continued)
Recall the schema mapping M from Example 1.15 and the source instance S^* from Example 1.4. A natural FO query over M's target schema is

$$q_1(x) \;:=\; \exists y \exists z \exists u \Big(\mathit{BookInfo}(y, \text{“Foundations of Databases”}, z) \wedge \mathit{WrittenBy}(y, u) \wedge \mathit{AuthorList}(u, x) \Big),$$

which asks for the names of all authors of the book “Foundations of Databases”. It is easy to see that for every solution T for S^* under M, we have

$$\{\text{“Serge Abiteboul”}, \text{“Richard Hull”}, \text{“Victor Vianu”}\} \;\subseteq\; q_1(T). \tag{1.2}$$

For the solution T^* from Example 1.5, $q_1(T^*)$ contains precisely the three names “Serge Abiteboul”, “Richard Hull” and “Victor Vianu”. On the other hand, there are solutions T for S^* under M, where the inclusion (1.2) is strict. For example, for the solution T' for S^* under M with

$$\begin{aligned}
\mathit{BookInfo}^{T'} \;&=\; \{(\text{“0-201-53771-0”},\ \text{“Foundations of Databases”},\ \bot_1),\\
&\quad\ (\text{“0-201-53082-1”},\ \bot_2,\ \bot_3)\},\\
\mathit{AuthorList}^{T'} \;&=\; \{(\bot_4,\ \text{“Serge Abiteboul”}),\ (\bot_5,\ \text{“Richard Hull”}),\\
&\quad\ (\bot_6,\ \text{“Victor Vianu”}),\ (\bot_7,\ \text{“Christos H. Papadimitriou”}),\\
&\quad\ (\bot_8,\ \text{“Jeffrey D. Ullman”})\},\\
\mathit{WrittenBy}^{T'} \;&=\; \{(\text{“0-201-53771-0”},\ \bot_4),\ (\text{“0-201-53771-0”},\ \bot_5),\\
&\quad\ (\text{“0-201-53771-0”},\ \bot_6),\ (\text{“0-201-53082-1”},\ \bot_7),\\
&\quad\ (\text{“0-201-53771-0”},\ \bot_8)\},
\end{aligned}$$

$q_1(T')$ consists of the names “Serge Abiteboul”, “Richard Hull”, “Victor Vianu” and “Jeffrey D. Ullman”. So what should be the answer to q_1 with respect to M and S? □

One way to deal with the first issue is to separate the data translation task from query answering: If we are given a query q over M's target schema, we imagine that a solution T for S under M has already been produced, and just evaluate q on T. That is, we do not care about the fact that T is the result of translating S according to M—we just view T as an ordinary instance, and evaluate q on T as we usually do on such an instance. However, as demonstrated in Example 1.18, the result of the query then depends on the particular solution. Even more, Example 1.18 shows that for some solutions T, the result of q_1 on T contains information that is neither present in the schema mapping nor in the source instance. For example, the result of q_1 on T' tells us that “Jeffrey

D. Ullman" is one of the authors of the book "Foundations of Databases", but this information is neither contained in M nor in S^*.

As a work-around, we could compute (in some deterministic way) an "appropriate" solution (i.e., every source instance S is assigned to precisely one such "appropriate" solution T_S that is computed whenever S needs to be translated to the target schema), and declare the result of a query to be the result of the query on that solution. However, to answer the question of what an "appropriate" solution is, it seems to be indispensable to first identify an "appropriate" way of answering queries.

In Fagin et al. [2005a], an approach based on the notion of *the certain answers* has been proposed to answer queries in relational data exchange. This approach is described next.

1.3.1 The Certain Answers Semantics

The certain answers semantics has been introduced by Fagin et al. [2005a] to answer queries in relational data exchange. As the name suggests, it is based on the notion of *the certain answers*, which is commonly used for answering queries on *incomplete instances*.

An *incomplete instance* $\mathcal{I}$ over a schema σ is a set of *ground* instances over σ (see, e.g., Abiteboul et al. [1995]). Such an instance models an instance with *incomplete information*: just think of $\mathcal{I}$ as representing an *unknown* instance I, where I can turn out to be any of the instances in $\mathcal{I}$. The instances in $\mathcal{I}$ are called the *possible worlds* of $\mathcal{I}$. Given a query q over σ, several ways of evaluating q on $\mathcal{I}$ have been proposed in the literature (cf., van der Meyden [1998]). A common way is to take the *certain answers to q on $\mathcal{I}$*:

1.19 Definition (The certain answers for incomplete instances)
Let $\mathcal{I}$ be an incomplete instance over a schema σ, and let q be a query over σ. *The certain answers to q on $\mathcal{I}$* are defined as

$$cert(q, \mathcal{I}) \ := \ \bigcap \{q(I) \mid I \in \mathcal{I}\}.$$

Here, for a set X, $\bigcap X$ denotes the intersection of all sets in X. Thus, $cert(q, \mathcal{I})$ consists of all tuples that occur in $q(I)$ for *all* possible worlds I of $\mathcal{I}$.

Let M be a schema mapping, and let S be a source instance for M. Basically, the set $sol(M, S)$ can be regarded as an incomplete instance over M's target schema. Indeed, this makes perfect sense: think of the result of the translation of S according to M as an *unknown* solution for S under M; the unknown solution could turn out to be any of the solutions in $sol(M, S)$. Fagin et al. [2005a] therefore proposed to answer a query q with respect to M and S by

the certain answers to q on M and S, defined as the set of all tuples that are contained in $q(T)$ for all solutions T for S under M.

However, depending on the particular query, evaluating the query this way may lead to counter-intuitive answers. The reason is that $sol(M, S)$ may contain instances with nulls (i.e., naive tables), which, as already indicated in Section 1.1.1, serve as representations for incomplete instances. Indeed, recall that nulls are placeholders for unknown constant values. Thus, each instantiation of the nulls in a naive table T by constants gives rise to a possible world of T. Formally, let a *valuation of* T be a mapping $f\colon \mathrm{dom}(T) \to \mathit{Const}$ such that for all $c \in \mathrm{const}(T)$ we have $f(c) = c$. Then, the set of all possible worlds of (the incomplete instance represented by) T is

$$\mathit{poss}(T) \;:=\; \{f(T) \mid f \text{ is a valuation of } T\}.$$

If query answering on a naive table is not done with care, counter-intuitive results may arise (see, e.g., Abiteboul et al. [1995], Imielinski and Lipski, Jr. [1984]). Therefore, it is desirable to define query answering semantics based on *ground* solutions only. Actually, in my opinion, naive tables should not be taken into account for defining the semantics of query answering, since naive tables just serve as representations for incomplete instances.

In the case of the certain answers semantics of Fagin et al. [2005a], the certain answers can in fact be defined without referring to solutions with nulls (cf., Proposition 1.21). Therefore, we define:

1.20 Definition (The certain answers for schema mappings)
Let M be a schema mapping, let S be a source instance for M, and let q be a query over M's target schema. The *certain answers to q on M and S* are defined as

$$\mathit{cert}(q, M, S) \;:=\; \mathit{cert}(q, \mathit{sol}_{\mathrm{gr}}(M, S)),$$

where $\mathit{sol}_{\mathrm{gr}}(M, S)$ is the set of all *ground* solutions for S under M.

1.21 Proposition.
Let M be a schema mapping, let S be a source instance for M, and let q be a query over M's target schema. Then,

$$\mathit{cert}(q, M, S) \;=\; \bigcap \{q(T) \mid T \in \mathit{sol}(M, S)\}.$$

Proof. The inclusion from right to left is clear. To prove the inclusion from left to right, assume that $\bar{t} \in \mathit{cert}(q, M, S)$. We must show that $\bar{t} \in q(T)$ for every solution T for S under M.

Let T be a solution for S under M. Since q is generic, there is a finite set $C \subseteq \mathit{Const}$ such that q is C-generic. Let $X := \mathrm{const}(T) \cup C$, and let Y be the

set of all values $u \in \mathit{Dom}$ for which there is a tuple $(u_1, \ldots, u_k) \in q(T)$ with $u \in \{u_1, \ldots, u_k\}$. Pick bijective mappings $f_1 \colon \mathit{Dom} \to \mathit{Dom}$ and $f_2 \colon \mathit{Dom} \to \mathit{Dom}$ such that

- $f_1(c) = f_2(c) = c$ for each constant $c \in X$,
- f_1 and f_2 map each null in T to a constant, and
- $f_1(Y) \cap f_2(Y) \subseteq X$.

Such mappings exist: First pick an arbitrary bijective mapping $f_1 \colon \mathit{Dom} \to \mathit{Dom}$ such that $f_1(c) = c$ for each $c \in X$, and $f_1(\bot) \in \mathit{Const}$ for each $\bot \in \mathrm{nulls}(T)$. Then pick a bijective mapping $f_2 \colon \mathit{Dom} \to \mathit{Dom}$ such that $f_2(c) = c$ for each $c \in X$, $f_2(u) \in \mathit{Const} \setminus f_1(Y)$ for each $u \in Y \setminus X$, and $f_2(\bot) \in \mathit{Const}$ for each $\bot \in \mathrm{nulls}(T) \setminus Y$. This is possible, since $q(T)$ is finite by definition, and therefore, Y and $f_1(Y)$ are finite.

Let $i \in \{1, 2\}$. Since M is generic, $f_i(T)$ is a ground solution for S under M. Together with $\bar{t} \in \mathit{cert}(q, M, S)$, we have $\bar{t} \in q(f_i(T))$, and C-genericity of q implies $\bar{t} \in f_i(q(T))$. We now have $\bar{t} \in f_1(q(T)) \cap f_2(q(T))$, which, together with $f_1(Y) \cap f_2(Y) \subseteq X$, implies that all values that occur in $\bar{t}$ belong to X. Consequently, $\bar{t} = f_1^{-1}(\bar{t}) \in f_1^{-1}(f_1(q(T))) = q(T)$. □

1.22 Example (The certain answers to q_1 from Example 1.18)
Recall the schema mapping M from Example 1.15, the source instance S^* from Example 1.4, and the FO query q_1 from Example 1.18. Eq. (1.2) implies

$$\{\text{“Serge Abiteboul”}, \text{“Richard Hull”}, \text{“Victor Vianu”}\} \subseteq \mathit{cert}(q_1, M, S^*).$$

On the other hand, if T^* is the solution for S^* under M from Example 1.5, then $q_1(T^*)$ contains precisely the three names “Serge Abiteboul”, “Richard Hull” and “Victor Vianu”. Thus,

$$\mathit{cert}(q_1, M, S^*) = \{\text{“Serge Abiteboul”}, \text{“Richard Hull”}, \text{“Victor Vianu”}\},$$

which is certainly the result that one intuitively expects. □

1.23 Example (The certain answers to another simple query)
Consider the schema mapping M and the source instance S^* for M from Example 1.15 and Example 1.4, respectively. For the query

$$q_2(x) := \exists y\, \mathit{BookInfo}(y, x, \text{“DB”}),$$

which asks for the titles of all books with genre “DB”, we have $\mathit{cert}(q_2, M, S^*) = \emptyset$. Intuitively, this is the only reasonable result, since neither M nor S^* provides any information about book genres, except that each book has a genre. □

The certain answers semantics has been widely adopted as the right semantics for answering *monotonic* queries. Given instances I and J over a schema σ, we call I a *subinstance* of J, and we write $I \subseteq J$, if and only if for all $R \in \sigma$ we have $R^I \subseteq R^J$. Then, a query q over a schema σ is called *monotonic* if and only if for all instances I and J over σ with $I \subseteq J$, we have $q(I) \subseteq q(J)$. Monotonic queries form a fundamental class of queries that contains, for example, all *conjunctive queries* and all *unions of conjunctive queries*.

1.24 Definition (Conjunctive query, union of conjunctive queries)
A *union of conjunctive queries* over a schema σ is a FO query over σ of the form

$$\varphi(\bar{x}) \;=\; \exists \bar{y}_1\, \psi_1(\bar{x}, \bar{y}_1) \;\vee\; \cdots \;\vee\; \exists \bar{y}_m\, \psi_m(\bar{x}, \bar{y}_m), \tag{1.3}$$

where for all $i \in \{1, \ldots, m\}$, ψ_i is a FO formula of the form $R_1(\bar{u}_1) \wedge \cdots \wedge R_n(\bar{u}_n)$ (with $n \geq 1$ and $R_1, \ldots, R_n \in \sigma$), and each variable x in $\bar{x}$ is a free variable in ψ_i. A *conjunctive query* is a query of the form (1.3), where $m = 1$.

Surprisingly, although the definition of the certain answers involves an intersection over an infinite set, it has been shown in Fagin et al. [2005a] that the certain answers to unions of conjunctive queries can be computed from an arbitrary *universal* solution by first evaluating the query on the universal solution, and then removing all tuples with nulls from the result. Details on this are given in Chapter 2. In order to be able to answer unions of conjunctive queries on the target, it therefore suffices to translate source instances to universal solutions. A similar result holds for *monotonic* queries and an appropriate extension of the notion of universal solutions [Deutsch et al., 2008]. Details on this are given in Chapter 2 as well.

1.3.2 Coping With Implicit Information

It has been observed by several authors (e.g., Fagin et al. [2005a], Arenas et al. [2004], and Libkin [2006]) that, for some *non-monotonic* queries, the certain answers semantics leads to answers that intuitively do not seem to be accurate. Let us give a concrete example to illustrate this issue.

1.25 Example (Copying schema mappings)
Consider the schema mapping $M = (\{R\}, \{R'\}, \Sigma)$, where R and R' are binary relation symbols, and Σ consists of the st-tgd

$$\forall x_1 \forall x_2 \Big(R(x_1, x_2) \rightarrow R'(x_1, x_2) \Big).$$

Taking into account that M specifies a translation from source to target, a natural interpretation of M is "copy the relation R into the relation R'". This

is the reason why schema mappings like M are often called *copying schema mappings* (see, e.g., Arenas et al. [2004], Libkin [2006]). Given a source instance S for M, it therefore seems natural to assume that the only possible result of translating S to the target is the target instance T_S for M with $(R')^{T_S} = R^S$ (the result of *copying* R to R'). This corresponds to the *closed world assumption (CWA)* (see, e.g., Reiter [1978], Libkin [2006]).[2]

Admittedly, the CWA-based interpretation is a matter of opinion. In some situations, it may be more natural to assume nothing about tuples $\bar{t} \in Dom^2$ that do not occur in R^S. That is, it may be more natural to leave open the possibility whether such a tuple $\bar{t}$ belongs to R' or not, and consider *every* solution for S under M as a possible result of translating S to the target. This corresponds to the *open world assumption (OWA)* (see also Reiter [1978], Libkin [2006]). Note that under the OWA-based interpretation, the certain answers semantics leads to the expected query results. Nevertheless, in many situations, the CWA-based interpretation seems to be more natural.

If we adopt the CWA-based interpretation, then the certain answers semantics may lead to answers that intuitively do not seem to be accurate. Specifically, consider the query

$$q(x) \;:=\; \exists y\Big(R'(x,y) \wedge \neg R'(y,x)\Big)$$

over M's target schema. Since for a given source instance S for M, we assume that T_S is the only possible outcome of translating S to the target, it seems to be reasonable to expect that the answer to q on M and S is the answer to q on T_S. In particular, given the source instance S for M with $R^S = \{(a,b)\}$, we would expect the answer to be $q(T_S) = \{a\}$. However, contrary to this expectation, we have $cert(q, M, S) = \emptyset$, since the target instance T for M with $(R')^T = \{(a,b),(b,a)\}$ is a ground solution for S under M. □

Furthermore, the following example shows that the certain answers semantics behaves in a rather unexpected way on *Boolean* queries. Recall that a Boolean query is a query of arity 0. Thus, the result of a Boolean query is either the empty set (which is interpreted as *no*) or the set that contains the empty tuple (which is interpreted as *yes*).

1.26 Example (Arenas et al. [2004])
This example is an adaptation of Proposition 5.4 of Arenas et al. [2004] to the copying schema mapping $M = (\{R\}, \{R'\}, \Sigma)$ from Example 1.25. Let q be a non-trivial Boolean query over $\{R'\}$ (i.e., q is not always true, and q is not

[2]For the moment, we do not need the general definition of the CWA. We use *CWA* just to refer to the assumption that the only possible result of translating S to the target is T_S. A precise definition of the CWA and variants thereof will be given in Chapters 3, 4 and 5.

always false). It seems to be natural to expect that there is a source instance S_1 for M such that the answer to q on M and S_1 is true, and a source instance S_2 for M such that the answer to q on M and S_2 is false. However, contrary to this expectation, we have

- $\mathit{cert}(q, M, S) = \emptyset$ for all source instances S for M, or
- $\mathit{cert}(\neg q, M, S) = \emptyset$ for all source instances S for M.

That is, q or $\neg q$ has a trivial result that does not depend on the source instance.

Indeed, suppose that $\mathit{cert}(q, M, S) \neq \emptyset$ for some source instance S for M. Then for every ground solution T for S under M, we have $q(T) \neq \emptyset$. We use this to show that for all source instances S' for M we have $\mathit{cert}(\neg q, M, S') = \emptyset$. Let S' be a source instance for M. Pick a ground solution T_1 for S under M, and a ground solution T_2 for S' under M. Then the target instance T for M with $(R')^T = (R')^{T_1} \cup (R')^{T_2}$ is both a ground solution for S under M, and a ground solution for S' under M. Since T is a ground solution for S under M, we have $q(T) \neq \emptyset$, and in particular, $\neg q(T) = \emptyset$. Since T is a ground solution for S' under M, this implies $\mathit{cert}(\neg q, M, S') = \emptyset$. □

Taking into account the definition of logically defined schema mappings, it is no surprise that the certain answers semantics behaves this way. In fact, these effects are just consequences of underspecifying schema mappings. For example, to obtain the CWA-based interpretation of the schema mapping M in Example 1.25, one could add the FO sentence

$$\forall x_1 \forall x_2 \Big(\neg R(x_1, x_2) \to \neg R'(x_1, x_2) \Big)$$

to M. Then T_S would be the only solution for a source instance S under M, and the certain answers semantics intuitively yields the expected results. Note also that for the modified schema mapping, the certain answers semantics does not exhibit the "strange" behavior demonstrated in Example 1.26. However, while this approach is feasible for simple schema mappings such as those in Example 1.25, it seems to be infeasible or at least tedious for more complex schema mappings. Furthermore, one would have to fully specify schema mappings, which is not what one usually wants to do in practice.

Nevertheless, the above examples show that it is often natural to interpret schema mappings M and source instances S for M with respect to certain additional information on the target, which can be imagined to be *implicit* in M and S. Namely, when adopting the CWA, since M and S specify a translation of S from the source schema to the target schema, it is natural to interpret M and S as if all data, for which M and S do not *explicitly* mention the possibility of occurring in the target, do *not* occur in the target. In Example 1.25, we

implicitly assumed that none of the tuples in $Dom^2 \setminus R^S$—of which M and S do not *explicitly* mention that they are contained in R'—belongs to R'. If, in Example 1.25, we would answer q on M and S by the certain answers to q on $\{T_S\}$ (the set of all possible outcomes of translating S to the target with respect to the information that none of the tuples in $Dom^2 \setminus R^S$ belongs to R'), we would obtain the expected query result $\{a\}$. Furthermore, answering queries on M and S by the certain answers on $\{T_S\}$ does not exhibit the behavior of the certain answers semantics described in Example 1.26. A further example is:

1.27 Example (Library database restructuring, continued)
Consider the schema mapping M and the source instance S^* for M from Example 1.15 and Example 1.4, respectively. Consider also the FO query

$$\begin{aligned} q_3(x) \;:=\; & \exists y \exists z\, \mathit{BookInfo}(x, y, z) \wedge \\ & \neg \exists y \Big(\mathit{WrittenBy}(x, y) \wedge \mathit{AuthorList}(y, \text{“Serge Abiteboul”}) \Big), \end{aligned}$$

which asks for all ISBN numbers of books where “Serge Abiteboul” is *not* an author. Since “0-201-53082-1” is the only ISBN number in S^* that is not associated with “Serge Abiteboul”, and M and S^* intuitively do not explicitly mention the possibility of an association of “0-201-53082-1” and “Serge Abiteboul” in the target, it seems to be natural to assume the answer to q_3 on M and S^* to be $\{$0-201-53082-1$\}$. Here, we implicitly assume that there are no tuples in *AuthorList* and *WrittenBy* that associate “Serge Abiteboul” with the ISBN number “0-201-53082-1”. We do this, since, intuitively, M and S^* do not explicitly mention the possibility of such an association. □

Of course, as argued in Example 1.25, it may make perfect sense to assume no additional information on the target, in which case the certain answers semantics leads to the desired results. However, in the remainder of this thesis, we consider the case that schema mappings and source instances are interpreted with respect to certain *implicit* information as described above.

Note that, being an intuitive notion, there are several ways of capturing implicit information in schema mappings and source instances. In fact, several semantics for query answering using implicit information have been proposed in the literature [Libkin, 2006, Hernich and Schweikardt, 2007, Libkin and Sirangelo, 2008, Afrati and Kolaitis, 2008, Hernich, 2010]. Depending on the particular application and one's point of view, one or the other of these semantics may be useful. In this thesis, I will describe these semantics in more detail. All of these semantics are based on the following ideas. Given a schema mapping M and a source instance S for M:

1. We identify a subset $\mathcal{S}$ of all solutions for S under M that is intended to be the set of all possible outcomes of translating S to the target if implicit information—in the formalized sense—is taken into account.

2. Given a query q over M's target schema, we answer q on M and S using the set $\mathcal{S}$, typically by taking the certain answers to q on $\mathcal{S}$.

For example, given the copying schema mapping M from Example 1.25 and a source instance S for M, we would like to have $\mathcal{S} = \{T_S\}$. Then, the certain answers to q on $\mathcal{S}$ yield the expected result $q(T_S)$.

It should be mentioned here that Fagin et al. [2005b] also proposed a semantics for overcoming the counter-intuitive behavior of the certain answers semantics on non-monotonic queries. Their semantics is based on universal solutions. More precisely, queries are answered by the certain answers over universal solutions. This semantics has very nice properties. For example, the answers to an existential query can be computed by first evaluating the query on the core of the universal solutions, and then removing from the set of answers all tuples containing a null. However, as observed by Arenas et al. [2004] (see the full version) and mentioned by Libkin [2006], that semantics exhibits counter-intuitive behavior, too. To learn more about the universal solution-based semantics, the interested reader is referred to Fagin et al. [2005b].

1.4 Contributions of this Thesis

The contributions of this thesis can be subdivided into three parts: the complexity of computing universal solutions, semantics for query answering using implicit information, and the complexity of evaluating queries using implicit information. In the following, these three parts are described in more detail.

The Complexity of Computing Universal Solutions

Universal solutions, which are studied in depth in Chapter 2, are "good" solutions in the sense that they are "most general", and that the certain answers to a large number of queries, including unions of conjunctive queries, can be computed directly from a universal solution [Fagin et al., 2005a,b, Arenas et al., 2009a]. This thesis' main contribution concerning the complexity of computing universal solutions is that there is a *fixed* schema mapping M defined by *tgds only* such that it is undecidable whether a given source instance for M has a universal solution under M (Theorem 2.33). This result appeared in Hernich and Schweikardt [2007], and strengthens a result of Deutsch et al. [2008].

The proof of this theorem has several consequences concerning *chase termination*. The *chase*—a standard tool in database theory—is a procedure that takes an instance I and a set Σ of tgds and egds, and iteratively tries to modify I so that the resulting instance satisfies Σ. Since it was shown in Fagin et al. [2005a] that the chase can be used to compute universal solutions for schema

mappings defined by tgds and egds, the chase has proved to be an indispensable tool in relational data exchange. Unfortunately, the chase does not always terminate. To this end, various conditions for chase termination have been proposed in the literature. All of these conditions are *sufficient*, but not necessary for chase termination. In fact, the proof of Theorem 2.33 implies that there is no *decidable* condition that is both sufficient and necessary for chase termination: chase termination is undecidable, even with respect to some fixed set of tgds (Corollary 2.34). This also strengthens a result of Deutsch et al. [2008].

Semantics for Query Answering Using Implicit Information

This thesis introduces and studies a number of query answering semantics that take into account implicit information in schema mappings and source instances. In the following, semantics contributed by this thesis are listed. These semantics can be divided into two groups.

The first group consists of semantics based on *CWA-solutions*. CWA-solutions are particular solutions that have been proposed by Libkin [2006] (together with corresponding query answering semantics) to answer queries in relational data exchange. Libkin designed CWA-solutions for the case of schema mappings defined by st-tgds. CWA-solutions are based on the CWA in the following sense:

1. Every tuple must be justified in some sense by the schema mapping and the source instance.

2. Each justification is used at most once.

3. A CWA-solution contains only "facts" that follow from the schema mapping and the source instance.

This thesis extends the definition of CWA-solutions to the more general case of schema mappings defined by tgds and egds (see Chapter 3). The main difficulty is to formalize the first two requirements. These are characterized by a derivation-based approach based on a suitably controlled version of the chase, and by a game-based approach. Moreover, the structure of the set of all CWA-solutions for given source instances and schema mappings, and the complexity of computing CWA-solutions is analyzed. In particular, it is shown that

- CWA-solutions are universal solutions that are derivable in some sense from the source instance using the tgds and egds,

- CWA-solutions exist if and only if universal solutions exist, and

- the *core of the universal solutions* (the "smallest" universal solution in the case of schema mappings defined by tgds and egds; cf., Fagin et al. [2005b], or Chapter 2) is the "smallest" CWA-solution.

Finally, results on the complexity of evaluating queries under the CWA-solution-based semantics are obtained. Details on this are given below. All of the above-mentioned results were published in Hernich and Schweikardt [2007].

The assumptions underlying the CWA-solution-based semantics reflect the operational point of view on tgds and egds. That is, tgds are considered as rules for deriving tuples, and egds are considered as rules for identifying values. Consequently, these semantics do not take into account logical equivalence of schema mappings. This means that there are schema mappings $M_1 = (\sigma_s, \sigma_t, \Sigma_1)$ and $M_2 = (\sigma_s, \sigma_t, \Sigma_2)$, where Σ_1 and Σ_2 are logically equivalent sets of tgds and egds, an instance S over σ_s, and a query q over σ_t such that the answer to q on M_1 and S differs from the answer to q on M_2 and S. Furthermore, query answers with respect to the CWA-solution-based semantics do not necessarily reflect the standard semantics of first-order quantifiers (e.g., existential quantifiers express that there are one, two, three or more elements that satisfy the given property, but this is not necessarily reflected by the CWA-solution-based semantics). Examples are given in Section 3.6.

The second group of semantics contributed by this thesis have been considered with the goal of finding semantics that

1. take into account implicit information in schema mappings and source instances,

2. respect logical equivalence of schema mappings, and

3. reflect the standard semantics of FO quantifiers.

First, semantics for query answering on deductive databases are studied in the framework of relational data exchange (see Chapter 4). In particular, semantics based on Reiter's CWA [Reiter, 1978], the generalized CWA (GCWA, for short) [Minker, 1982], the extended GCWA (EGCWA, for short) [Yahya and Henschen, 1985], and the possible worlds semantics (PWS, for short) [Chan, 1993] are studied. It turns out that in the context of relational data exchange, these semantics seem to be either too weak (GCWA) or too strong (CWA, EGCWA), and in one case (PWS), logical equivalence of schema mappings is not respected.

Inspired by the GCWA-semantics and the EGCWA-semantics, the *GCWA*-answers semantics* is developed in Chapter 5. Under the GCWA*-answers semantics, queries are answered by the certain answers on *GCWA*-solutions*. In contrast to the above-mentioned semantics and solution concepts, the GCWA*-answers semantics and GCWA*-solutions are defined for *all* schema mappings (rather than for schema mappings defined by tgds and egds). For schema mappings defined by st-tgds and egds, GCWA*-solutions are just ground solutions

that are unions of $\subseteq$-*minimal solutions* (solutions for which no proper subinstance is a solution). Finally, the GCWA*-answers semantics is designed in such a way that, intuitively, it meets the three goals 1–3 mentioned above.

The results concerning the second group of semantics were published in Hernich [2010].

The Complexity of Evaluating Queries Using Implicit Information

This thesis also investigates the complexity of evaluating queries under query answering semantics that take into account implicit information in schema mappings and source instances. We concentrate on the *data complexity*, that is, the complexity with respect to *fixed* schema mappings and *fixed* queries (schema mappings and queries do not belong to the input).

Concerning the CWA-solution-based query answering semantics, we show the following. First, it turns out that two of the CWA-solution-based semantics coincide with the certain answers semantics on a large class of monotonic queries, including unions of conjunctive queries, and source instances that have a CWA-solution (see Section 3.4.1). Hence, results on evaluating the certain answers to such queries obtained in the literature carry over to those semantics. For each of the CWA-solution-based semantics, it is then shown that for properly restricted schema mappings M, and FO queries q, the complexity of the following problem is in co-NP or NP, depending on the particular semantics: given a source instance S for M and a tuple $\bar{t} \in Dom^{\mathrm{ar}(q)}$, does $\bar{t}$ belong to the set of answers to q on M and S under the corresponding semantics (Theorem 3.44)? Furthermore, there is a simple schema mapping M, and a conjunctive query q with one additional inequality such that this problem may be complete for the corresponding complexity class (Theorem 3.45).

A larger part of this thesis deals with the complexity of evaluating queries with respect to the GCWA*-answers semantics. It is shown that the GCWA*-answers semantics and the certain answers semantics coincide on monotonic queries (Proposition 5.10), so that all results on computing the certain answers to such queries obtained in the literature directly apply to the GCWA*-answers semantics. However, there are simple schema mappings M defined by st-tgds and simple *existential queries* q of the form $\exists \bar{x}\, \varphi$, where φ is a conjunction of three relational atomic formulas and just one negated relational atomic formula, such that the problem

EVAL(M, q)	
Input:	a source instance S for M, and a tuple $\bar{t} \in Dom^{\mathrm{ar}(q)}$
Question:	Is $\bar{t}$ an answer to q with respect to the GCWA*-answers semantics?

is co-NP-hard (Proposition 5.11). Permitting only one additional universal quantifier can make this problem undecidable (Proposition 5.12), which is even true for schema mappings defined by st-tgds. It seems then surprising that $\textsc{Eval}(M, q)$ is in PTIME for *universal queries* q (queries of the form $\forall \bar{x}\, \varphi$, where φ is quantifier-free) and suitable restrictions on M. This follows from the following result.

This thesis' main contribution concerning the complexity of evaluating queries using implicit information is Theorem 5.15:

> For every schema mapping M defined by certain st-tgds (called *packed st-tgds*), and for each universal query q over M's target schema, there is a polynomial time algorithm that takes the core of the universal solutions for some source instance S under M as input and computes the GCWA*-answers to q on M and S.

In my opinion, this is also the technically most challenging contribution of this thesis. By standard results on computing the core of the universal solutions (see, e.g., Fagin et al. [2005b] or Chapter 2), it implies that $\textsc{Eval}(M, q)$ is in PTIME if M is defined by packed st-tgds and q is a universal query over M's target schema.

In the following, we give some details on how Theorem 5.15 is proved. We first observe that it suffices to develop a polynomial time algorithm for the following problem: given the core of the universal solutions for some source instance S for M, and a tuple $\bar{t} \in \mathit{Const}^{\mathrm{ar}(q)}$, decide whether $\bar{t}$ belongs to the GCWA*-answers to q on M and S. We then reduce this problem to the problem of checking whether there is a union T of one or more $\subseteq$-minimal instances in $\mathit{poss}(T_0)$ (see Section 1.3.1 for the definition), where T_0 is the core of the universal solutions for S under M, such that $T \models \neg q(\bar{t})$. The key step to this result is a characterization of ground $\subseteq$-minimal solutions for S under M as $\subseteq$-minimal instances in $\mathit{poss}(T_0)$. By transforming $\neg q$ into a kind of disjunctive normal form, we can then focus without loss of generality on the case that $\neg q$ is logically equivalent to a formula $\bar{q} = \exists \bar{x}\, \varphi$, where φ is a conjunction of atomic formulas and negations of atomic formulas. If φ consists of one atomic formula and $\bar{x}$ is the empty tuple, then the problem can be solved as follows. First, we reduce the infinite set of all $\subseteq$-minimal instances in $\mathit{poss}(T_0)$ to a set $\mathcal{S}$ of possibly exponential size. The technically difficult part is then to identify a particular subset $\mathcal{S}'$ of $\mathcal{S}$ of polynomial size with the "right" properties so that all tuples that occur in $\subseteq$-minimal instances of $\mathit{poss}(T_0)$ can be reconstructed from the instances in $\mathcal{S}'$. Finally, for solving the general problem, we combine partial solutions to a solution to the whole problem, where the results proved for the case of one atomic formula help us to prove correctness of this construction.

1.5 Structure of this Thesis

This thesis consists of four main chapters, plus the introduction and the conclusion.

Chapter 2 deals with computing the certain answers to *monotonic queries*. Most of this chapter is devoted to the special case of computing the certain answers to queries that are *preserved under homomorphisms*—a fundamental class of queries that contains, for example, all unions of conjunctive queries. In particular, we review *universal solutions* and *the core of the universal solutions*, which are the basic tools for computing the certain answers to such queries, and are also used in other chapters of this thesis. Furthermore, we review techniques and results on computing universal solutions. This chapter also presents one of the main results of this thesis: that there is a schema mapping M defined by tgds for which it is undecidable whether a given source instance for M has a universal solution under M.

Chapter 3 introduces and studies CWA-solutions and the corresponding query answering semantics. It also discusses generalizations as well as specializations of CWA-solutions that appeared in the literature. The final section discusses some limitations of the CWA-solution-based semantics.

Chapter 4 studies several semantics for query answering on deductive databases in the context of relational data exchange.

In Chapter 5, we finally introduce and study the GCWA*-answers semantics. This chapter comprises the technically most challenging result of this thesis, namely that for each schema mapping M defined by packed st-tgds and for each universal query q, there is a polynomial time algorithm that takes the core of the universal solutions for S under M as input, and computes the GCWA*-answers to q on M and S.

2 Computing the Certain Answers to Monotonic Queries

This chapter deals with computing the certain answers to monotonic queries. Recall from Section 1.3.1 that a query q over a schema σ is *monotonic* if and only if for all instances I and J over σ with $I \subseteq J$, we have $q(I) \subseteq q(J)$. Monotonic queries form a fundamental class of queries that contains, for example, all *unions of conjunctive queries* and all *Datalog queries*. As mentioned in Section 1.3.1, the certain answers semantics has been widely adopted as the right semantics for answering monotonic queries.

Up to date, the special case of computing the certain answers to queries that are *preserved under homomorphisms* is best understood. Most of this chapter (Section 2.1 to Section 2.3) is devoted to this problem. We will define in Section 2.1 what it means for a query to be preserved under homomorphisms. Examples of queries preserved under homomorphisms are unions of conjunctive queries, and, more generally, Datalog queries. From Rossman [2008] and Atserias et al. [2006] it even follows that every FO query that is preserved under homomorphisms is logically equivalent to a union of conjunctive queries!

The central tool for computing the certain answers to queries preserved under homomorphisms are *universal solutions*. Universal solutions were introduced by Fagin et al. [2005a] as a formalization of the intuitive notion of "most general solutions", and they showed that the problem of computing the certain answers to a number of queries preserved under homomorphisms (like unions of conjunctive queries, and Datalog queries) essentially *reduces* to the problem of query answering on *universal solutions*. In fact, the certain answers to queries preserved under homomorphisms can be obtained by evaluating the query on a universal solution, and removing all tuples from the result that contain a null. A particularly important role plays the *core of the universal solutions*, introduced in Fagin et al. [2005b], which, for schema mappings defined by tgds and egds, is the "unique smallest" universal solution (up to isomorphism). A formal definition of universal solutions and their core, and a more detailed explanation of how universal solutions can be used to compute the certain answers to queries preserved under homomorphisms are given in Section 2.1.

As already mentioned in Section 1.1.2, for *algorithmic* results, we restrict attention to schema mappings defined by tgds and egds. For computing universal solutions under such schema mappings, the *chase*—a standard tool in database theory and proposed for computing universal solutions in Fagin et al. [2005a]—

is essential. The chase is a procedure that takes an instance I and a finite set Σ of tgds and egds as input, and iteratively tries to modify I so that the resulting instance satisfies Σ. Unfortunately, the chase does not always terminate. To this end, various sufficient conditions have been proposed in the literature that, imposed on a set Σ of tgds and egds, ensure chase termination for *every* given instance I. All these conditions are sufficient, but not necessary for chase termination. On the positive side, given a fixed schema mapping $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$, where Σ satisfies one of these chase termination conditions, there is a polynomial time algorithm that takes a source instance S for M as input, and either outputs a universal solution for S under M if a solution for S under M exists, or outputs that there is no solution for S under M. A corresponding result concerning the core of the universal solutions (and one particular of these chase termination conditions) has been proved by Gottlob and Nash [2008]. Section 2.2 formally defines the chase, and explains its application to computing universal solutions in more detail. It also presents some of the conditions for chase termination, and surveys state-of-the-art algorithms for computing the core of the universal solutions.

Section 2.3 then presents one of the main results of this thesis. Namely, we show that chase termination is undecidable, even with respect to some fixed schema mapping defined by tgds. Even more, we show that there is a *fixed* schema mapping M defined by *tgds only* such that the following problem is undecidable: given a source instance S for M, is there a universal solution for S under M? Parts of the results of Section 2.3 were published in Hernich and Schweikardt [2007].

Finally, Section 2.4 reviews standard techniques and results on evaluating queries that are not preserved under homomorphisms.

2.1 Review of Universal Solutions and Their Core

Universal solutions were introduced by Fagin et al. [2005a] as a formalization of the intuitive notion of "most general solutions". The formal definition of universal solutions is based on *homomorphisms*:

2.1 Definition (Homomorphism)
Let I and J be instances over a schema σ. A *homomorphism* from I to J is a mapping $h\colon \mathrm{dom}(I) \to \mathrm{dom}(J)$ such that

- for all $R \in \sigma$ and all $\bar{t} \in R^I$, we have $h(\bar{t}) \in R^J$, and
- for all $c \in \mathrm{const}(I)$, we have $h(c) = c$.

2.2 Definition (Universal solution)
Let M be a schema mapping, and let S be a source instance for M. A *universal solution* for S under M is a solution T for S under M such that for all solutions T' for S under M there is a homomorphism from T to T'.

2.3 Example (Library database restructuring, continued)
Let M be the schema mapping from Example 1.15, and let S^* be the source instance for M from Example 1.4. Then it is not hard to see that the target instance T^* from Example 1.5, with

$$\begin{aligned}
\mathit{BookInfo}^{T^*} &= \{(\text{“0-201-53771-0”},\ \text{“Foundations of Databases”},\ \bot_1),\\
&\quad\ (\text{“0-201-53082-1”},\ \bot_2,\ \bot_3)\},\\
\mathit{Authors}^{T^*} &= \{(\bot_4,\ \text{“Serge Abiteboul”}),\ (\bot_5,\ \text{“Richard Hull”}),\\
&\quad\ (\bot_6,\ \text{“Victor Vianu”}),\ (\bot_7,\ \text{“Christos H. Papadimitriou”})\},\\
\mathit{WrittenBy}^{T^*} &= \{(\text{“0-201-53771-0”},\ \bot_4),\ (\text{“0-201-53771-0”},\ \bot_5),\\
&\quad\ (\text{“0-201-53771-0”},\ \bot_6),\ (\text{“0-201-53082-1”},\ \bot_7)\},
\end{aligned}$$

is a universal solution for S^* under M.

Indeed, let T be an arbitrary solution for S^* under M. Since $S^* \cup T$ satisfies the tgd χ'_1 (see Example 1.15 for the definition of χ'_1), there must be some value u_1 such that $\mathit{BookInfo}^T$ contains the tuple

$$(\text{“0-201-53771-0”},\ \text{“Foundations of Databases”},\ u_1).$$

Furthermore, since $S^* \cup T$ satisfies the tgd χ'_2, there must be values u_4, u_5, u_6 and u_7 such that $\mathit{Authors}^T$ contains the tuples

$$(u_4,\ \text{“Serge Abiteboul”}),\ (u_5,\ \text{“Richard Hull”}),\ (u_6,\ \text{“Victor Vianu”}),\\ (u_7,\ \text{“Christos H. Papadimitriou”}),$$

and $\mathit{WrittenBy}^T$ contains the tuples

$$(\text{“0-201-53771-0”},\ u_4),\ (\text{“0-201-53771-0”},\ u_5),\ (\text{“0-201-53771-0”},\ u_6),\\ (\text{“0-201-53082-1”},\ u_7).$$

Finally, since $S^* \cup T$ satisfies χ_3 and χ_4, there must be values u_2 and u_3 such that $\mathit{BookInfo}^T$ contains the tuple

$$(\text{“0-201-53082-1”},\ u_2,\ u_3).$$

Thus, the mapping $h\colon \mathrm{dom}(T^*) \to \mathrm{dom}(T)$ which satisfies $h(\bot_i) = u_i$ for each $i \in \{1, \ldots, 7\}$, and $h(c) = c$ for each constant $c \in \mathrm{const}(T^*)$, is a homomorphism from T^* to T.

Note that T^* is not the only universal solution for S^* under M. For example, if $\perp, \perp', \perp''$ are distinct nulls that do not occur in T^*, the instance obtained from T^* by adding the tuple $(\perp, \perp', \perp'')$ to $\mathit{BookInfo}^{T^*}$ is a universal solution for S^* under M, since a homomorphism can map the tuple $(\perp, \perp', \perp'')$ to the tuple ("0-201-53771-0", "Foundations of Databases", $\perp_1$) in $\mathit{BookInfo}^{T^*}$. □

Given a schema mapping M and a source instance S for M, there may be many different universal solutions for S under M, or none (cf., Example 2.12); furthermore, there may be universal solutions of different sizes (cf., Example 2.3). On the other hand, Definition 2.2 immediately implies that any two universal solutions for S under M are *homomorphically equivalent*:

2.4 Definition (Homomorphically equivalent)
We call instances I and J *homomorphically equivalent* if and only if there is a homomorphism from I to J, and a homomorphism from J to I.

In some cases, however, homomorphic equivalence is not enough: one would rather like to compute a "unique smallest" universal solution. To this end, Fagin et al. [2005b] proposed to consider *the core of the universal solutions*, which, for schema mappings defined by tgds and egds, is the "unique smallest" universal solution (up to isomorphism). For example, the instance T^* in Example 2.3 is the core of the universal solutions for S^* under M. The formal definition is based on the notion of *cores*:

2.5 Definition (Core, core of an instance)
Let I be an instance. We say that I is a *core* if and only if there is no homomorphism from I to an instance $J \subseteq I$ with $J \neq I$. A *core of* I is an instance $J \subseteq I$ such that J is a core, and there is a homomorphism from I to J.

The following basic properties of cores have been obtained by Hell and Nešetřil [1992] in the context of graphs (i.e., for instances over $\{E\}$, where E is binary), and easily carry over to arbitrary instances.

2.6 Theorem (Hell and Nešetřil [1992]).

1. *Every instance has a core.*
2. *Let I_1 and I_2 be homomorphically equivalent instances, let J_1 be a core of I_1, and let J_2 be a core of I_2. Then J_1 and J_2 are* isomorphic. *In particular, any two cores of an instance are isomorphic.*
3. *If J is a core of an instance I, then there is a homomorphism h from I to J that is the identity on* $\mathrm{dom}(J)$. *In particular, $h(I) = J$.*

We can now give the definition of the core of the universal solutions. Let M be a schema mapping, and let S be a source instance for M. Suppose that there is a universal solution for S under M. Since any two universal solutions for S under M are homomorphically equivalent, Theorem 2.6(2) implies that there is an instance T_0 for M such that for every universal solution T' for S under M and each core T'_0 of T', we have $T_0 \cong T'_0$. Since T_0 is unique up to isomorphism, we call T_0 *the core of the universal solutions* for S under M. In Example 2.3, T^* is the core of the universal solutions for S^* under M. If there is no universal solution for S under M, then the core of the universal solutions is undefined.

2.7 Definition (core of the universal solutions)
For every schema mapping M, and for every source instance S for M, let $\mathrm{Core}(M, S)$ be a target instance for M that is isomorphic to the core of the universal solutions for S under M if there is at least one universal solution for S under M. Otherwise, let $\mathrm{Core}(M, S)$ be undefined.

As mentioned above, $\mathrm{Core}(M, S)$ is a universal solution for S under M if M is defined by tgds and egds, but it may fail to be a solution otherwise:

2.8 Proposition (Fagin et al. [2005b]).

1. *If $M = (\sigma_s, \sigma_t, \Sigma)$ is a schema mapping, where Σ consists of tgds and egds, and if S is a source instance for M for which $\mathrm{Core}(M, S)$ is defined, then $\mathrm{Core}(M, S)$ is a universal solution for S under M.*

2. *There is a schema mapping M and a source instance S for M such that $\mathrm{Core}(M, S)$ is defined, and $\mathrm{Core}(M, S)$ is no solution for S under M.*

Universal solutions are interesting for relational data exchange not only because they are "most general" solutions, but because they can be used to compute the certain answers to queries that are *preserved under homomorphisms*, as shown by Theorem 2.10 below.

2.9 Definition (homomorphism-preserved query)
We say that a query q over a schema σ is *preserved under homomorphisms* if and only if for all instances I and J over σ, and every homomorphism h from I to J, the following is true: if $\bar{t} \in q(I)$, then $h(\bar{t}) \in q(J)$.

Note that all homomorphism-preserved queries are monotonic, and that the class of all queries that are preserved under homomorphisms contains all unions of conjunctive queries, and, more generally, all Datalog queries. From Rossman [2008] and Atserias et al. [2006], it even follows that every FO query that is preserved under homomorphisms is logically equivalent to a union of conjunctive queries!

2.10 Theorem (essentially in Fagin et al. [2005a]).
Let M be a schema mapping, let S be a source instance for M, and let T be a universal solution for S under M. Then for each homomorphism-preserved query q over M's target schema, we have

$$\mathit{cert}(q, M, S) \;=\; \{\bar{t} \in q(T) \mid \bar{t} \text{ contains only constants}\}.$$

Proof. The proof is a straightforward generalization of the analogous result for *unions of conjunctive queries* obtained by Fagin et al. [2005a], taking into account genericity of M and q.

Let us first prove the inclusion from right to left. Assume that $\bar{t} \in q(T)$, and that $\bar{t}$ contains only constants. Let T' be a solution for S under M, and let h be a homomorphism from T to T'. Since q is preserved under homomorphisms, we have $h(\bar{t}) \in q(T')$. Thus, since $\bar{t}$ contains only constants, and homomorphisms are the identity on constants, we have $\bar{t} \in q(T')$.

Let us now prove the inclusion from left to right. To this end, assume that $\bar{t} \in \mathit{cert}(q, M, S)$. By Proposition 1.21, we have $\bar{t} \in q(T)$. It remains to prove that $\bar{t}$ contains only constants.

This follows immediately from the genericity of M and q: Let $C \subseteq \mathit{Const}$ be such that q is C-generic, let $X := \mathrm{const}(T) \cup C$, and let Y be the set of values (i.e., constants and nulls) that occur in $q(T)$. Pick a bijective mapping $f\colon \mathit{Dom} \to \mathit{Dom}$ such that

- $f(c) = c$ for each $c \in X$, and
- $f(Y) \cap Y \subseteq X$.

Since M is generic, $f(T)$ is a solution for S under M, so by Proposition 1.21 and $\bar{t} \in \mathit{cert}(q, M, S)$, we have $\bar{t} \in q(f(T))$. Since q is C-generic, this leads to $\bar{t} \in f(q(T))$. Consequently, $\bar{t} \in q(T) \cap f(q(T))$, which, together with $f(Y) \cap Y \subseteq X$ implies that all values that occur in $\bar{t}$ belong to the set X of constants. □

We thus have:

2.11 Corollary.
Let M be a schema mapping, and let q be a union of conjunctive queries over M's target schema. Then there is a polynomial time algorithm that, given a universal solution for some source instance S for M as input, outputs $\mathit{cert}(q, M, S)$.

Proof. Let T be a universal solution for some source instance S for M. Given T, the algorithm computes $X := q(T)$, which is possible in time polynomial in the size of T since q is fixed, and outputs all tuples $\bar{t} \in X$ that contain only constants. By Theorem 2.10, the set of all these tuples is $\mathit{cert}(q, M, S)$. □

The corollary holds, more generally, for all queries q that are preserved under homomorphisms, and which have polynomial time data complexity (e.g., unions of conjunctive queries, or Datalog queries). Thus, the crucial step in computing the certain answers to such queries is to compute a universal solution.

2.2 How to Compute Universal Solutions?

In this section, we deal with the problem of computing universal solutions.

First observe that, even if a source instance S has a solution under a schema mapping M, it may have no universal solution under M:

2.12 Example (Universal solutions may not exist)
Consider the schema mapping $M = (\{P\}, \{E\}, \Sigma)$, where Σ consists of the tgds

$$\chi_1 \;=\; \forall x(P(x) \rightarrow \exists y\, E(x, y)),$$

and

$$\chi_2 \;=\; \forall x \forall y(E(x, y) \rightarrow \exists z\, E(y, z)).$$

Let S be the source instance for M with $P^S = \{a\}$. We claim that there is no universal solution for S under M.

Suppose, to the contrary, that there is a universal solution T for S under M. Let n be the largest integer for which there are pairwise distinct values $u_0, u_1, \ldots, u_n$ such that $u_0 = a$, and $(u_i, u_{i+1}) \in E^T$ for all $i \in \{0, 1, \ldots, n-1\}$. Such an n exists, because T is finite. Since T satisfies Σ, we must have $(u_n, u_i) \in E^T$ for some $i \in \{0, 1, \ldots, n\}$.

Now let T' be the solution for S under M, where $E^{T'}$ consists of an $(n+2)$-cycle. That is, there are pairwise distinct values $u'_0, u'_1, \ldots, u'_{n+1}$ such that $u'_0 = a$, $(u'_i, u'_{i+1}) \in E^{T'}$ for each $i \in \{0, 1, \ldots, n\}$, and $(u'_{n+1}, u'_0) \in E^{T'}$. Then there is no homomorphism from T to T'. In particular, T is no universal solution for S under M. □

However, in many cases, universal solutions can be obtained using *the chase*. In the following, we consider only schema mappings defined by tgds and egds.

2.2.1 Review of the Chase

The chase is the central tool for computing universal solutions for schema mappings defined by tgds and egds (see Fagin et al. [2005a]; for other applications of the chase, see, e.g., Abiteboul et al. [1995]). The chase is a procedure that

takes an instance I and a finite set Σ of tgds and egds as input, and iteratively tries to modify I so that the resulting instance satisfies Σ.

For presenting the chase, it is convenient to view an instance as a set of atoms. An *atom* is an expression $R(\bar{u})$, where R is a relation symbol, and $\bar{u} \in Dom^{\mathrm{ar}(R)}$. An *atom of an instance* I over a schema σ is an atom $R(\bar{u})$ with $R \in \sigma$ and $\bar{u} \in R^I$. The *set of all atoms of* I is denoted by

$$\mathrm{atoms}(I) \;:=\; \{R(\bar{u}) \mid R(\bar{u}) \text{ is an atom of } I\}.$$

We will often identify an instance I with the set $\mathrm{atoms}(I)$. That is, we view I and $\mathrm{atoms}(I)$ as one and the same object. In particular, we can write $R(\bar{u}) \in I$ instead of $\bar{u} \in R^I$, and $R(\bar{u}) \notin I$ instead of $\bar{u} \notin R^I$. Furthermore, operations and relations defined on sets carry over to instances. For example, given instances I and J, we write $I \cup J$ for the instance K with $\mathrm{atoms}(K) = \mathrm{atoms}(I) \cup \mathrm{atoms}(J)$, and $I \setminus J$ for the instance K with $\mathrm{atoms}(K) = \mathrm{atoms}(I) \setminus \mathrm{atoms}(J)$.

Given an instance I over a schema σ, and a finite set Σ of tgds and egds over σ, the chase starts with $I_0 := I$, and proceeds in steps $s = 1, 2, 3, \ldots$ as follows. In step $s \geq 1$, an instance I_{s-1} has already been computed. If $I_{s-1} \models \Sigma$, then the chase stops and outputs I_{s-1}. We say that I_{s-1} is the *result* of the chase on I and Σ. If $I_{s-1} \not\models \Sigma$, there must be some tgd or egd $\chi \in \Sigma$ with $I_{s-1} \not\models \chi$. The chase selects one such χ, and modifies I_{s-1} as follows.

Suppose that χ is a tgd of the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$. Since $I_{s-1} \not\models \chi$, there are tuples $\bar{u}, \bar{v}$ such that $I_{s-1} \models \varphi(\bar{u},\bar{v})$, and there is no tuple $\bar{w}$ with $I_{s-1} \models \psi(\bar{u},\bar{w})$. We say that χ *applies to* I_{s-1} *with* $\bar{u}, \bar{v}$. The chase now picks an arbitrary tuple $\bar{w}$ of pairwise distinct nulls that do not occur in I_{s-1}, and adds just those atoms to I_{s-1} so that the resulting instance I_s satisfies ψ under $\bar{u}, \bar{w}$. Formally, $I_s = I_{s-1} \cup \psi[\bar{u},\bar{w}]$, where $\psi[\cdot]$ is defined as follows:

2.13 Notation ($\varphi[\alpha]$, $\varphi[\bar{u}]$)
Let $\varphi(x_1, \ldots, x_k)$ be a FO formula of the form $R_1(\bar{u}_1) \wedge \cdots \wedge R_n(\bar{u}_n)$, and let α be an assignment for $\{x_1, \ldots, x_k\}$. Then,

$$\varphi[\alpha] \;:=\; \{R_i(\alpha^*(\bar{u}_i)) \mid 1 \leq i \leq n\}.$$

(Recall that α^* is the extension of α to the identity on the constants that occur in φ.) For a tuple $\bar{u} = (u_1, \ldots, u_k) \in Dom^k$, we use $\varphi[\bar{u}]$ as an abbreviation for $\varphi[\alpha]$, where α maps each x_i to u_i.

Formally, such a chase step is captured by a *tgd chase step:*

2.14 Definition (Tgd chase step)
Let I be an instance, let χ be a tgd of the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$, and let α be an assignment for the variables in $\bar{x}$ and $\bar{y}$.

- We say that χ *applies to* I *with* α if and only if $I \models \varphi(\alpha)$, and there is no tuple $\bar{w} \in Dom^{|\bar{z}|}$ such that $I \models \psi(\alpha(\bar{x}), \bar{w})$.

- Given an instance J, we write $I \vdash_{\chi,\alpha} J$ if and only if χ applies to I with α, and $J = I \cup \psi[\alpha(\bar{x}), \bar{w}]$, where $\bar{w}$ is a tuple of pairwise distinct nulls from $Null \setminus \mathrm{dom}(I)$.

Suppose now that χ is an egd of the form $\forall\bar{x}(\varphi(\bar{x}) \rightarrow x_i = x_j)$, where $\bar{x} = (x_1, \ldots, x_k)$. Since $I_{s-1} \not\models \chi$, there is a tuple $\bar{u} = (u_1, \ldots, u_k) \in Dom^k$ such that $I_{s-1} \models \varphi(\bar{u})$ and $u_i \neq u_j$. If u_i and u_j are constants, then the chase *fails*. Otherwise, the chase identifies the two values u_i and u_j. Such a chase step is formally captured by an *egd chase step*:

2.15 Definition (Egd chase step)
Let I be an instance, let χ be an egd of the form $\forall\bar{x}(\varphi(\bar{x}) \rightarrow x_i = x_j)$, let α be an assignment for the variables in $\bar{x}$, let $u_i := \alpha(x_i)$, and let $u_j := \alpha(x_j)$.

- We say that χ *applies to* I *with* α if and only if $I \models \varphi(\alpha)$ and $u_i \neq u_j$. If χ applies to I with α, and both u_i and u_j are constants, we say that χ *fails on* I *with* α.

- Given an instance J, we write $I \vdash_{\chi,\alpha} J$ if and only if χ applies to I with α, χ does not fail on I with α, and $J = f(I)$, where $f\colon \mathrm{dom}(I) \rightarrow \mathrm{dom}(I)$ is such that
  - $f(u) = f(v) = v$, where u is one of the nulls in $\{u_i, u_j\}$, and v is the other value in $\{u_i, u_j\}$, and
  - $f(u) = u$ for all $u \in \mathrm{dom}(I) \setminus \{u_i, u_j\}$.

In general, we denote a chase step from an instance I to an instance J using a tgd or an egd in Σ by $I \vdash_\Sigma J$. That is, we write $I \vdash_\Sigma J$ if and only if there is some $\chi \in \Sigma$ and an appropriate assignment α such that $I \vdash_{\chi,\alpha} J$.

The possible "runs" of the chase are formally described by *chase sequences*:

2.16 Definition (chase sequence)
Let I be an instance over a schema σ, and let Σ be a finite set of tgds and egds over σ.

- A *chase sequence* on I and Σ is a (finite or infinite) sequence $C = I_0, I_1, \ldots$ of instances over σ such that $I_0 = I$, and for every instance I_s in C with $s \geq 1$ we have $I_{s-1} \vdash_\Sigma I_s$.

- Let $C = (I_0, I_1, \ldots, I_l)$ be a finite chase sequence on I and Σ. We call I_l the *result* of C, and l the *length* of C.

- A *complete chase sequence* on I and Σ is a finite chase sequence $C = (I_0, I_1, \ldots, I_l)$ on I and Σ such that there is no instance J with $I_l \vdash_\Sigma J$.
- A *successful chase sequence* on I and Σ is a complete chase sequence C on I and Σ such that the result of C satisfies Σ.
- A *failing chase sequence* on I and Σ is a complete chase sequence C on I and Σ such that the result of C does not satisfy Σ.

2.17 Example (Library database restructuring, continued)
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be the schema mapping from Example 1.15, and let S^* be the source instance for M from Example 1.4. Consider the sequence $C = (I_0, I_1, \ldots, I_7)$ with

- $I_0 = S^*$,
- $I_1 = I_0 \cup \{\mathit{BookInfo}(\text{“0-201-53771-0”}, \text{“Found. of Databases”}, \perp_1)\}$,
- $I_2 = I_1 \cup \{\mathit{Authors}(\perp_4, \text{“Serge Abiteboul”}), \mathit{WrittenBy}(\text{“0-201-53771-0”}, \perp_4)\}$,
- $I_3 = I_2 \cup \{\mathit{Authors}(\perp_5, \text{“Richard Hull”}), \mathit{WrittenBy}(\text{“0-201-53771-0”}, \perp_5)\}$,
- $I_4 = I_3 \cup \{\mathit{Authors}(\perp_6, \text{“Victor Vianu”}), \mathit{WrittenBy}(\text{“0-201-53771-0”}, \perp_6)\}$,
- $I_5 = I_4 \cup \{\mathit{Authors}(\perp_7, \text{“Christos H. Papadimitriou”}), \mathit{WrittenBy}(\text{“0-201-53082-1”}, \perp_7)\}$,
- $I_6 = I_5 \cup \{\mathit{BookInfo}(\text{“0-201-53082-1”}, \perp_2, \perp_3), \mathit{Authors}(\perp_7, \perp_8)\}$,
- $I_7 = I_6 \setminus \{\mathit{Authors}(\perp_7, \perp_8)\}$.

It is easy to see that C is a chase sequence on S^* and Σ. For example, we have $I_0 \vdash_{\chi'_1,\alpha} I_1$, where $\alpha(x_1) = \text{“0-201-53771-0”}$ and $\alpha(x_2) = \text{“Found. of Databases”}$. The instances I_2 to I_6 can be obtained similarly using χ'_2 (for I_2 to I_5) and χ_4 (for I_6). Finally, we have $I_6 \vdash_{\chi_3,\alpha'} I_7$, where $\alpha'(x_1) = \perp_7$, $\alpha'(x_2) = \text{“Christos H. Papadimitriou”}$ and $\alpha'(x_3) = \perp_8$.

Note that $I_7 = S^* \cup T^*$, where T^* is the universal solution for S^* under M from Example 1.5. It follows that C is successful. □

We can now state the crucial properties of the chase that connect the chase with the problem of computing universal solutions:

2.18 Theorem (Fagin et al. [2005a]).
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ is a set of tgds and egds, and let S be a source instance for M. Then we have:

1. *Let I be the result of a successful chase sequence on S and Σ. Then $I \setminus S$ is a universal solution for S under M.*

2. *If there is a failing chase sequence on S and Σ, then there is no solution for S under M.*

2.19 Example (Library database restructuring, continued)
Recall the setting from Example 2.17. We have shown that $I_7 = S^* \cup T^*$ is the result of a successful chase sequence on S^* and Σ, and indeed, $I_7 \setminus S^* = T^*$ is a universal solution for S^* under M. □

Sometimes, we also need to refer to the following lemma, which is the key lemma for the proof of Theorem 2.18 in Fagin et al. [2005a]:

2.20 Lemma (Fagin et al. [2005a]).
Suppose that $I \vdash_{\chi,\alpha} J$, where χ is a tgd or an egd, and α is an appropriate assignment. Let K be an instance such that $K \models \chi$, and there is a homomorphism from I to K. Then there is a homomorphism from J to K.

Lemma 2.20, in particular, implies:

2.21 Corollary.
If C is an arbitrary chase sequence on an instance I and a finite set Σ of tgds and egds, K is an instance with $K \models \Sigma$, and there is a homomorphism from I to K, then there is a homomorphism from the result of C to K.

From Theorem 2.18 we know that universal solutions can be computed using the chase. Unfortunately, the chase does not always terminate:

2.22 Example (The chase may not terminate)
Consider the schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$ and the source instance S for M from Example 2.12. Then there is no complete chase sequence C on S and Σ. If there was such a sequence C, then C would be successful, since Σ contains only tgds, and by Theorem 2.18, the result of C would lead to a universal solution for S under M. However, in Example 2.12, we have shown that such a universal solution does not exist.

Let us now add the t-tgd

$$\chi_3 \;:=\; \forall x_1 \forall x_2 \forall x_3 (E(x_1, x_2) \wedge E(x_2, x_3) \rightarrow E(x_1, x_1))$$

to M. That is, let $M' = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma \cup \{\chi_3\})$. Then S has a universal solution under M', namely the target instance T for M' with $E^T = \{(a, a)\}$. However, it is not hard to verify that there is no successful chase sequence on S and $\Sigma \cup \{\chi_3\}$. □

Even worse, we show in Section 2.3 that chase termination is *undecidable*, even with respect to some *fixed* finite set Σ of *tgds only* !

2.2.2 Sufficient Conditions for Chase Termination

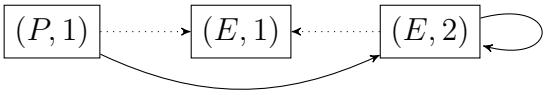
A number of sufficient conditions for chase termination have been exposed in the literature. The first such condition was *weak acyclicity*:

2.23 Definition (Weak acyclicity)
Let Σ be a finite set of tgds over a schema σ.

- A *position* over σ is a pair (R, p) such that $R \in \sigma$ and $p \in \{1, \ldots, \mathrm{ar}(R)\}$.
- Let φ be a FO formula over σ of the form $R_1(\bar{u}_1) \wedge \cdots \wedge R_k(\bar{u}_k)$, and let x be a variable. We say that x *appears at position* (R, p) *in* φ if and only if there is some $i \in \{1, \ldots, k\}$ such that $R_i = R$, $\bar{u}_i$ has the form $(u_{i,1}, \ldots, u_{i,\mathrm{ar}(R)})$, and $u_{i,p} = x$.
- The *dependency graph* of Σ is a directed graph, where the vertices are the positions over σ, and for each tgd $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$ in Σ, each variable x in $\bar{x}$, and each position (R, p) at which x appears in φ, there is
  - a *copying* edge from (R, p) to every position at which x appears in ψ, and
  - an *existential edge* from (R, p) to every position at which some variable from $\bar{z}$ appears in ψ.
- Σ is called *weakly acyclic* if the dependency graph of Σ contains no cycle with an existential edge.

Intuitively, weak acyclicity prevents the chase from cascading the creation of nulls, that is, that a null is created at some position based on a null at the same position.

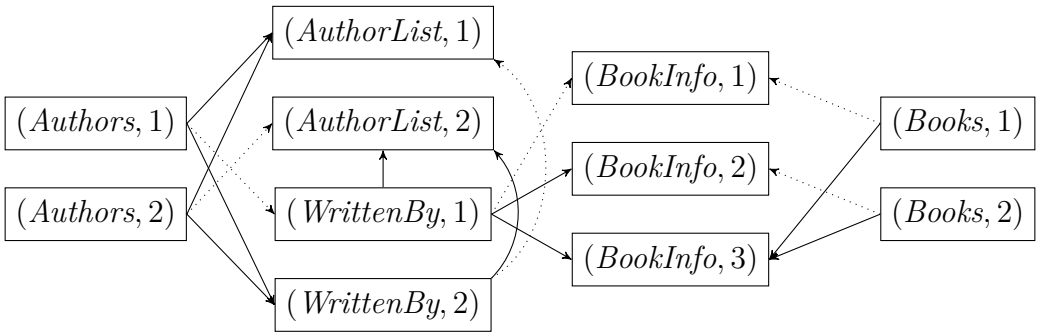
2.24 Example
Consider the schema mapping $M = (\{P\}, \{E\}, \Sigma)$ from Example 2.12. The dependency graph of Σ is shown below:



Here, solid edges represent existential edges, while dotted edges represent copying edges. The loop at $(E, 2)$ is a cycle through the existential edge. Thus, Σ is not weakly acyclic. □

We call a schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$ *weakly acyclic* if and only if Σ is the union of a weakly acyclic set of tgds, and a set of egds.

2.25 Example (Library database restructuring, continued)
Recall the schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$ from Example 1.15. The dependency graph of the set Σ' of all tgds in Σ (i.e., $\Sigma' = \{\chi'_1, \chi'_2, \chi_4\}$) is shown below:



As above, solid edges represent existential edges, while dotted edges represent copying edges. Since the dependency graph of Σ' contains no cycle through an existential edge, Σ' is weakly acyclic. Consequently, M is weakly acyclic. □

For weakly acyclic schema mappings, it is known that the chase terminates within a number of steps that is polynomial in the size $\|T\|$ of T, where the size of an instance I over a schema σ is defined as

$$\|I\| \;:=\; \sum_{R \in \sigma} |R^I| \cdot \mathrm{ar}(R).$$

Furthermore, with respect to weakly acyclic schema mappings, universal solutions exist if and only if solutions exist:

2.26 Theorem (Fagin et al. [2005a]).
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a weakly acyclic schema mapping.

1. *There is a polynomial p_M such that, given a source instance S for M, each chase sequence of S with Σ has length at most $p_M(\|S\|)$.*

2. *Let S be a source instance for M. Then there is a universal solution for S under M if and only if there is a solution for S under M.*

A matching lower bound follows from Theorem 2.26(2) and:

2.27 Theorem (Kolaitis et al. [2006]).
There is a weakly acyclic schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$, where Σ consists of full tgds and a single egd, such that Existence-of-Solutions(M) *is* PTIME*-complete.*

Weak acyclicity has been generalized in a number of ways. A first such generalization, called *stratification*, was identified by Deutsch et al. [2008]. Stratification was further generalized by Lausen et al. [2009]. In fact, Lausen et al. [2009] identified an infinite hierarchy of generalizations of weak acyclicity. Theorem 2.26 holds as well for schema mappings satisfying one of these conditions. Finally, Marnette [2009] identified a class of schema mappings that strictly generalizes the class of all weakly acyclic schema mappings, and for which the *oblivious chase*, a generalization of the chase, is guaranteed to terminate within a polynomial number of steps on every given source instance.

2.2.3 How to Compute the Core of the Universal Solutions?

Let us now turn to the problem of computing the core of the universal solutions. In general, computing a core of a given instance is a hard problem. Using a simple reduction from the 3-colorability problem (see, e.g., Garey and Johnson [1979]), one can show that the corresponding decision problem (given instances I and J, is I a core of J) is NP-hard [Fagin et al., 2005b]. It was established by Fagin et al. [2005b] that this decision problem is actually DP-complete, where DP is the class of all problems that can be written as the intersection of a problem in NP and a problem in co-NP. For more background on the class DP, see Fagin et al. [2005b] or Papadimitriou [1994].

Surprisingly, in a number of cases it is nevertheless possible to compute the core of the universal solutions. A first such result was:

2.28 Theorem (Fagin et al. [2005b]).
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of st-tgds. Then there is a polynomial time algorithm that takes a source instance S for M as input and outputs Core(M, S).

Let $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$ be a schema mapping, where Σ consists of st-tgds, and let S be a source instance for M. The algorithm guaranteed by Theorem 2.28 exploits that universal solutions obtained from a successful chase sequence on S and Σ have a very nice structure that can be described in terms of *blocks*:

2.29 Definition (Block)
Let I be an instance.

- The *Gaifman graph of the nulls of I* is the undirected graph whose vertices are the nulls of I, and which has an edge between two nulls $\perp, \perp' \in \text{nulls}(I)$ if and only if $\perp \neq \perp'$, and there is an atom $R(\bar{u})$ of I such that both $\perp$ and $\perp'$ occur in $\bar{u}$.
- A *block* of I is the set of nulls in a connected component of the Gaifman graph of the nulls of I.

2.30 Lemma (implicit in Fagin et al. [2005b]).
For every schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$, where Σ consists of st-tgds, there is a positive integer b with the following property: If S is a source instance for M, and T is a solution for S under M such that $S \cup T$ is the result of a successful chase sequence on S and Σ, then each block of T contains at most b nulls.

As a first step, the algorithm from Theorem 2.28 computes the result I of a successful chase sequence on S and Σ. Note that Σ is a fixed set of st-tgds, and therefore, every chase sequence on S and Σ has length polynomial in the size of S, and every complete chase sequence on S and Σ is successful. The instance $T := I \setminus S$ is then a universal solution for S under M, and by Lemma 2.30, each block of T contains at most b nulls for some positive integer b that depends only on M. Subsequently, the algorithm computes a core of T using the algorithm guaranteed by:

2.31 Lemma (implicit in Fagin et al. [2005b]).
There is an algorithm that takes an instance I as input, and outputs a core $J \subseteq I$ of I in time $O(\|I\|^{b+3})$, where b is the maximum number of nulls in a block of I.

Proof. The following algorithm $\mathbb{A}$ is a modification of the *blocks algorithm* from Fagin et al. [2005b]. To obtain the blocks algorithm from $\mathbb{A}$, one first computes T using a successful chase sequence on S and Σ as described above, and replaces the instance I in step 1 of $\mathbb{A}$ with T.

On input I, $\mathbb{A}$ proceeds as follows:

1. Compute a list $B_1, \ldots, B_m$ of all blocks of I, and initialize J to be I.
2. Check whether there is a homomorphism h from J to J such that h is not injective, and there is some $i \in \{1, \ldots, m\}$ such that $h(u) = u$ for each $u \in \text{dom}(J) \setminus B_i$.
3. If such a h exists, then update J to be $h(J)$, and go to step 2.

4. Output J.

In Fagin et al. [2005b], it was shown that this algorithm correctly computes a core of I, and that steps 2 to 4 can be accomplished in time $O(\|I\|^{b+3})$. In particular, this dominates the time needed to accomplish step 1. □

Using more sophisticated techniques, Theorem 2.28 can be extended to schema mappings defined by st-tgds and egds (see Fagin et al. [2005b]), and even to *weakly acyclic schema mappings*:

2.32 Theorem (Gottlob and Nash [2008]).
Let M be a weakly acyclic schema mapping. Then there is a polynomial time algorithm that takes a source instance S for M as input, and either outputs $\mathrm{Core}(M, S)$ *if it exists, or tells us that* $\mathrm{Core}(M, S)$ *does not exist.*

Theorem 2.32 was further generalized by Marnette [2009].

2.3 Undecidability of the Existence of Universal Solutions

In Section 2.2, we reviewed results from [Fagin et al., 2005a, Deutsch et al., 2008, Lausen et al., 2009, Marnette, 2009] showing that for every fixed schema mapping M that is defined by tgds and egds, and that satisfies certain chase termination conditions (e.g., weak acyclicity, stratification, or safe restriction), there is an algorithm that takes a source instance S for M as input and either computes some universal solution for S under M if there is any, or outputs that there is no such universal solution. In this section, we show that for general schema mappings, even for schema mappings defined by tgds only, such an algorithm may not exist.

For a schema mapping M, consider the decision problem

Existence-of-Universal-Solutions(M)	
Input:	a source instance S for M
Question:	Is there a universal solution for S under M?

The following theorem was published in Hernich and Schweikardt [2007]:

2.33 Theorem.
There is a schema mapping $M_{\mathrm{Halt}} = (\sigma_s, \sigma_t, \Sigma)$, where Σ is a set of tgds, such that Existence-of-Universal-Solutions(M_{Halt}) *is undecidable.*

Proof. We will construct M_{HALT} in such a way that Turing machines can be encoded as source instances for M_{HALT}, and universal solutions correspond to halting computations of Turing machines on the empty input.

In the following, we consider *deterministic* Turing machines with a *single tape* that is infinite *only to the right*. Such Turing machines will be formally represented by tuples (Q, A, δ, q_0, Q_F), where Q is the set of states, A is the tape alphabet, $\delta\colon (Q \setminus Q_F) \times A \to Q \times A \times \{\text{L}, \text{R}\}$ is the *total* transition function, $q_0 \in Q$ is the start state, and $Q_F \subseteq Q$ is the set of final states. Clearly, the following problem is undecidable (see, e.g., Papadimitriou [1994]):

HALT	
Input:	a deterministic Turing machine $\mathbb{M}$ with a single tape that is infinite only to the right
Question:	Does $\mathbb{M}$ halt on the empty input?

Let $\mathbb{M} = (Q, A, \delta, q_0, Q_F)$ be a deterministic Turing machine with a single tape that is infinite only to the right. We can encode $\mathbb{M}$ by an instance $S_{\mathbb{M}}$ over the schema $\sigma_{\text{s}} = \{\Delta, Q_0\}$, where

- $\Delta^{S_{\mathbb{M}}} = \{(q, a, q', a', d) \mid q \in Q,\ a \in A,\ (q', a', d) = \delta(q, a)\}$ is the graph of the transition function δ,[1] and
- $Q_0^{S_{\mathbb{M}}} = \{q_0\}$ contains the start state q_0.

We choose σ_{s} as the *source schema* of M_{HALT}.

Our next goal is to construct the *target schema* σ_{t} of M_{HALT}, so that finite computations of $\mathbb{M}$ can be encoded as instances over σ_{t}.

Let us first fix some basic notation concerning Turing machine computations. Recall that a *computation* γ of M on the empty input is a sequence $\gamma = (\gamma_0, \gamma_1, \ldots, \gamma_n)$ of *configurations* of M, where γ_0 is the *start configuration* of M on the empty input, and for each $s \in \{0, 1, \ldots, n-1\}$, γ_{s+1} is the *successor configuration* of γ_s. We represent each configuration γ_s by a triple (q_s, p_s, a_s), where

- $q_s \in Q$ is the state in step s of γ,
- p_s is the head position in step s of γ (w.l.o.g. a nonnegative integer, since $\mathbb{M}$'s tape is infinite only to the right), and

[1] We assume, without loss of generality, that $Q \cup A \cup \{\text{L}, \text{R}\} \subseteq \mathit{Const}$, and that the sets Q, A and $\{\text{L}, \text{R}\}$ are pairwise disjoint.

- $a_s = a_{s,0}a_{s,1}\cdots a_{s,s+1} \in A^{s+2}$ is the inscription of the tape at positions 0 to $s+1$ in step s of γ (we only list the inscriptions of the first $s+2$ tape cells, since $\mathbb{M}$ can visit at most s tape cells in s steps, so that all other tape cells contain blanks; we add 2 to simplify the presentation).

Note that $\gamma_0 = (q_0, 0, \square\square)$, where $\square$ denotes the *blank symbol*, which is assumed to be in A. A configuration (q, p, a) is called *final* if and only if $q \in Q_F$. Note that a final configuration has no successor configuration.

Now let $\gamma = (\gamma_0, \gamma_1, \ldots, \gamma_n)$ be a computation of $\mathbb{M}$ on the empty input, where $\gamma_s = (q_s, p_s, a_{s,0}\cdots a_{s,s+1})$ for each $s \in \{0, 1, \ldots, n\}$. To represent γ as an instance—and also for the whole remaining part of this proof—it may be helpful to view γ as a matrix as in Figure 2.1. The sth row of this matrix is

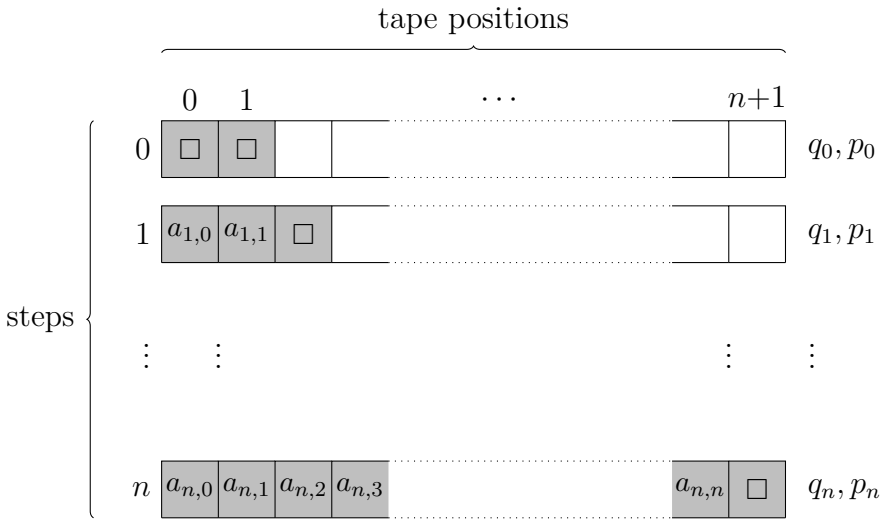


Figure 2.1: Representation of the computation γ in matrix form

labeled with the state q_s and the head position p_s in configuration γ_s, and the (highlighted) first $s+2$ entries of row s correspond to the inscriptions of the first $s+2$ tape cells in configuration γ_s.

Let us pick pairwise distinct values $u_0, u_1, \ldots, u_n, v_0, \ldots, v_{n+1} \in \mathit{Dom}$, where $u_0, u_1, \ldots, u_n$ will be used to encode the steps $0, 1, \ldots, n$, and $v_0, v_1, \ldots, v_{n+1}$ will be used to encode the tape positions $0, 1, \ldots, n+1$. Then we can represent γ by an instance T'_γ over the schema $\sigma'_{\mathrm{t}} = \{\mathit{Succ}_t, \mathit{Succ}_p, \mathit{State}, \mathit{Ins}\}$, where:

- $\mathit{Succ}_t^{T'_\gamma} = \{(u_s, u_{s+1}) \mid 0 \leq s \leq n-1\}$ is the successor relation on the "steps" $u_0, u_1, \ldots, u_n$.

- $Succ_p^{T'_\gamma} = \{(u_s, v_p, v_{p+1}) \mid 0 \leq s \leq n,\ 0 \leq p \leq s\}$ contains the successor relations on the "tape positions" $v_0, v_1, \ldots, v_{s+1}$ in each configuration γ_s.
- $State^{T'_\gamma} = \{(u_s, q_s, v_{p_s}) \mid 0 \leq s \leq n\}$ contains information about the state q_s and the "head position" v_{p_s} in each configuration γ_s. Note that the unique tuple of the form $(u_s, \cdot, \cdot)$ in $State^{T'_\gamma}$ corresponds to the label of row s in Figure 2.1.
- $Ins^{T'_\gamma} = \{(u_s, v_p, a_{s,p}) \mid 0 \leq s \leq n,\ 0 \leq p \leq s+1\}$ contains information about the inscriptions $a_{s,p}$ at each "position" v_p in each configuration γ_s. Note that the tuples $(u_s, \cdot, \cdot)$ in $Ins^{T'_\gamma}$ correspond to the first $s+2$ entries in row s in Figure 2.1.

Later, it will be important that u_0, v_0, v_1 are *constants*, and the remaining values $u_1, \ldots, u_n, v_2, \ldots, v_{n+1}$ are *nulls*. Therefore, we fix $u_0 := 0$ (for step 0), $v_0 := 0$ (for tape position 0), $v_1 := 1$ (for tape position 1), and we let all other values $u_1, \ldots, u_n, v_2, \ldots, v_{n+1}$ be nulls.

To support tgds in verifying that the relations $Succ_p^{T'_\gamma}$, $Succ_p^{T'_\gamma}$, $State^{T'_\gamma}$ and $Ins^{T'_\gamma}$ indeed encode a computation, we need to extend T'_γ by some auxiliary relations. As the *target schema* σ_{t} of M_{HALT}, we therefore choose the schema $\sigma'_{\mathrm{t}} \cup \{\tilde{\Delta}, Copy_L, Copy_R, End\}$. Then, γ can be represented by an instance T_γ over σ_{t}, where T_γ coincides with T'_γ on all relation symbols in σ'_{t}, and:

- $\tilde{\Delta}^{T_\gamma} = \Delta^{S_{\mathbb{M}}}$ is the graph of the transition function δ,
- $Copy_L^{T_\gamma} = \{(u_s, u_{s-1}, v_p) \mid 1 \leq s \leq n,\ 0 \leq p \leq p_s\}$ will be used to verify that configurations γ_s and γ_{s-1} coincide on all "positions" to the left of "position" v_{p_s} (excluding position v_{p_s}),
- $Copy_R^{T_\gamma} = \{(u_s, u_{s-1}, v_p) \mid 1 \leq s \leq n,\ p_s \leq p \leq s\}$ will be used to verify that configurations γ_s and γ_{s-1} coincide on all "positions" to the right of "position" v_{p_s} (excluding position v_{p_s}), and
- $End^{T_\gamma} = \{(u_s, v_{s+1}) \mid 0 \leq s \leq n\}$ marks "tape position" v_{s+1} in each configuration γ_s.

It remains to construct a finite set Σ of tgds such that universal solutions for $S_{\mathbb{M}}$ under M_{HALT} represent halting computations of $\mathbb{M}$ on the empty input. To simplify notation, we omit the universal quantifiers in front of tgds.

We start with an empty set Σ and add tgds to Σ as follows. To "copy" Δ to $\tilde{\Delta}$, and to "produce" a representation of the start configuration of $\mathbb{M}$ on the empty input, we add the st-tgds

$$\Delta(q, a, q', a', d) \rightarrow \tilde{\Delta}(q, a, q', a', d) \tag{2.1}$$

and

$$Q_0(q) \to \mathit{State}(0, q, 0) \wedge \mathit{Ins}(0, 0, \square) \wedge \mathit{Ins}(0, 1, \square) \\ \wedge \mathit{Succ}_p(0, 0, 1) \wedge \mathit{End}(0, 1) \tag{2.2}$$

to Σ. Note that tgd (2.2) enforces that a solution for $S_{\mathbb{M}}$ under M_{HALT} contains the atoms $\mathit{State}(0, q_0, 0)$, $\mathit{Ins}(0, 0, \square)$, $\mathit{Ins}(0, 1, \square)$, $\mathit{Succ}_p(0, 0, 1)$ and $\mathit{End}(0, 1)$, which correspond to the start configuration $(q_0, 0, \square\square)$ of $\mathbb{M}$ on the empty input.

To simulate a transition (where the head moves to the left, or where the head moves to the right), we add the t-tgds

$$\varphi_{\text{L}}(t, q, q', p, p', a, a') \to \exists t'\, \psi(t, t', q', p, p', a, a'), \tag{2.3}$$
$$\varphi_{\text{R}}(t, q, q', p, p', a, a') \to \exists t'\, \psi(t, t', q', p, p', a, a') \tag{2.4}$$

to Σ, where

$$\varphi_d := \begin{cases} \mathit{State}(t, q, p) \wedge \mathit{Ins}(t, p, a) \wedge \mathit{Succ}_p(t, p', p) \wedge \tilde{\Delta}(q, a, q', a', d), & \text{if } d = \text{L} \\ \mathit{State}(t, q, p) \wedge \mathit{Ins}(t, p, a) \wedge \mathit{Succ}_p(t, p, p') \wedge \tilde{\Delta}(q, a, q', a', d), & \text{if } d = \text{R} \end{cases}$$

states that a transition from the current configuration γ_t in step t to its successor configuration $\gamma_{t'}$ in step t' is possible, where the head moves to the left if $d = \text{L}$, or to the right if $d = \text{R}$; and where

$$\psi := \mathit{Succ}_t(t, t') \wedge \mathit{State}(t', q', p') \wedge \mathit{Ins}(t', p, a') \wedge \mathit{Copy}_L(t', t, p) \wedge \mathit{Copy}_R(t', t, p)$$

"creates" an initial part of the configuration $\gamma_{t'}$, and links γ_t to $\gamma_{t'}$.

To enforce that the tape inscriptions in a configuration and its successor configurations coincide on all unmodified tape cells, and to ensure that all tape positions from step t occur in step t' in the same order, we add the t-tgds

$$\mathit{Copy}_L(t', t, p) \wedge \mathit{Succ}_p(t, p', p) \wedge \mathit{Ins}(t, p', a) \;\to \\ \mathit{Copy}_L(t', t, p') \wedge \mathit{Succ}_p(t', p', p) \wedge \mathit{Ins}(t', p', a), \tag{2.5}$$

$$\mathit{Copy}_R(t', t, p) \wedge \mathit{Succ}_p(t, p, p') \wedge \mathit{Ins}(t, p', a) \;\to \\ \mathit{Copy}_R(t', t, p') \wedge \mathit{Succ}_p(t', p, p') \wedge \mathit{Ins}(t', p', a) \tag{2.6}$$

to Σ. Finally, to add a new tape cell to the end of the tape in step t', and to mark this cell the last tape cell for step t', we add the t-tgd

$$\mathit{Succ}_t(t, t') \wedge \mathit{End}(t, p) \;\to \\ \exists p' \Big(\mathit{Succ}_p(t', p, p') \wedge \mathit{Ins}(t', p', \square) \wedge \mathit{End}(t', p') \Big). \tag{2.7}$$

This finishes the construction of Σ.

Let $M_{\text{HALT}} = (\sigma_s, \sigma_t, \Sigma)$. We claim that $\mathbb{M}$ halts on the empty input if and only if there is a universal solution for $S_{\mathbb{M}}$ under M_{HALT}. Even stronger, we show that the following four statements are equivalent:

1. $\mathbb{M}$ halts on the empty input.
2. There is a successful chase sequence on $S_{\mathbb{M}}$ and Σ.
3. Each chase sequence on $S_{\mathbb{M}}$ and Σ can be extended to a successful chase sequence on $S_{\mathbb{M}}$ and Σ.
4. There is a universal solution for $S_{\mathbb{M}}$ under M_{HALT}.

In particular, EXISTENCE-OF-UNIVERSAL-SOLUTIONS(M_{HALT}) is undecidable.

We first show by induction on n that:

> For every computation $\gamma = (\gamma_0, \gamma_1, \ldots, \gamma_n)$ of $\mathbb{M}$ on the empty input, there is a finite chase sequence C_γ on $S_{\mathbb{M}}$ and Σ with result $S_{\mathbb{M}} \cup T_\gamma$. (*)

For the case that $n = 0$, (*) follows immediately from $\gamma_0 = (q_0, 0, \square\square)$ and the definition of T_γ.

Assume now that (*) is true for n. Let $\gamma = (\gamma_0, \gamma_1, \ldots, \gamma_{n+1})$ be a computation of $\mathbb{M}$ on the empty input, and let $\gamma' := (\gamma_0, \gamma_1, \ldots, \gamma_n)$. By the induction hypothesis, there is a finite chase sequence $C_{\gamma'}$ on $S_{\mathbb{M}}$ and Σ with result $S_{\mathbb{M}} \cup T_{\gamma'}$. Suppose that $\gamma_n = (q, p, a_{n,0} \cdots a_{n,n+1})$ and $\gamma_{n+1} = (q', p', a'_{n+1,0} \cdots a'_{n+1,n+2})$. Then, we must have $\delta(q, a_{n,p}) = (q', a'_{n+1,p}, d)$ for some $d \in \{\text{L}, \text{R}\}$. Thus, by construction of $T_{\gamma'}$, either the tgd (2.3) (if $d = \text{L}$) or the tgd (2.4) (if $d = \text{R}$) applies to $T_{\gamma'}$ with some assignment α. It follows immediately that T_γ can be obtained from $T_{\gamma'}$ by a finite chase sequence C on $T_{\gamma'}$ and Σ by first applying (2.3) or (2.4), then (2.5) and (2.6) as long as possible (note that this can be done only finitely many times), and finally, (2.7). The concatenation of $C_{\gamma'}$ and C is therefore a finite chase sequence on $S_{\mathbb{M}}$ and Σ with result $S_{\mathbb{M}} \cup T_\gamma$. This completes the induction step.

We are now ready to prove the equivalence of the statements 1–4 above.

1 $\Longrightarrow$ 2: Suppose that $\mathbb{M}$ halts on the empty input. Then there is a computation $\gamma = (\gamma_0, \gamma_1, \ldots, \gamma_n)$ of $\mathbb{M}$ on the empty input, for which γ_n is final. By (*), the instance $S_{\mathbb{M}} \cup T_\gamma$ is the result of a finite chase sequence C on $S_{\mathbb{M}}$ and Σ. Since γ_n is final, it is easy to verify that $T_\gamma \models \Sigma$. Thus, C is successful.

2 $\Longrightarrow$ 3: Consider an arbitrary chase sequence C on $S_{\mathbb{M}}$ and Σ. By induction on the length of C, it is easy to check that the result of C is isomorphic to some

subinstance of $S_{\mathbb{M}} \cup T_\gamma$. In particular, this implies that C can be extended to a successful chase sequence on $S_{\mathbb{M}}$ and Σ.

3 $\Longrightarrow$ 4: Since each chase sequence on $S_{\mathbb{M}}$ and Σ can be extended to a successful chase sequence on $S_{\mathbb{M}}$ and Σ, there is a successful chase sequence C on $S_{\mathbb{M}}$ and Σ. Let I be the result of C. Then, by Theorem 2.18(1), $I \setminus S_{\mathbb{M}}$ is a universal solution for $S_{\mathbb{M}}$ under M_{HALT}.

4 $\Longrightarrow$ 1: Let T be a universal solution for $S_{\mathbb{M}}$ under M_{HALT}. Let n be the largest integer for which there are pairwise distinct values $u'_0, u'_1, \ldots, u'_n$ such that $u'_0 = 0$, and $(u'_s, u'_{s+1}) \in \mathit{Succ}_t^T$ for all $s \in \{0, 1, \ldots, n-1\}$. We show that $\mathbb{M}$ halts on the empty input.

Suppose, for a contradiction, that $\mathbb{M}$ does *not halt* on the empty input. Then there is a computation $\gamma = (\gamma_0, \gamma_1, \ldots, \gamma_{n+1})$ of $\mathbb{M}$ on the empty input, where γ_{n+1} is *not final*. By (*), there is a chase sequence C_γ on $S_{\mathbb{M}}$ and Σ with result $S_{\mathbb{M}} \cup T_\gamma$. By Corollary 2.21, there is a homomorphism h from T_γ to T. Recall that

$$\mathit{Succ}_t^{T_\gamma} = \{(u_s, u_{s+1}) \mid 0 \leq s \leq n\}.$$

Therefore,

$$(h(u_s), h(u_{s+1})) \in \mathit{Succ}_t^T \quad \text{for each } s \in \{0, 1, \ldots, n\}, \tag{2.8}$$

and by the choice of n:

$$\text{There are distinct } k, l \in \{0, 1, \ldots, n+1\} \text{ such that } h(u_k) = h(u_l). \tag{2.9}$$

Now extend T_γ to a solution T' for $S_{\mathbb{M}}$ under M_{HALT} as follows. First, add (u_{n+1}, u_{n+1}) to $\mathit{Succ}_t^{T_\gamma}$. Second, for all $p \in \{0, 1, \ldots, n+2\}$,

- add $(u_{n+1}, v_p, v_{p'})$ to $\mathit{Succ}_p^{T_\gamma}$ for all $p' \in \{0, 1, \ldots, n+2\}$,
- add (u_{n+1}, q, v_p) to $\mathit{State}^{T_\gamma}$ for all $q \in Q$,
- add (u_{n+1}, v_p, a) to Ins^{T_γ} for all $a \in A$,
- add (u_{n+1}, u_{n+1}, v_p) to $\mathit{Copy}_L^{T_\gamma}$ and to $\mathit{Copy}_R^{T_\gamma}$,
- add (u_{n+1}, v_p) to End^{T_γ}.

Since T is a universal solution for $S_{\mathbb{M}}$ under M_{HALT}, there is a homomorphism h' from T to T'. We show by induction on s that

$$h'(h(u_s)) = u_s \quad \text{for all } s \in \{0, 1, \ldots, n+1\}. \tag{2.10}$$

For $s = 0$, we have $h'(h(u_0)) = u_0$, since $u_0 = 0$ is a constant, and homomorphisms are the identity on constants. Now assume that $h'(h(u_s)) = u_s$ for some $s \in \{0, 1, \ldots, n\}$. Recall that, by (2.8), we have $(h(u_s), h(u_{s+1})) \in Succ_t^T$. Since h' is a homomorphism from T to T', we have $(h'(h(u_s)), h'(h(u_{s+1}))) \in Succ_t^{T'}$. Furthermore, since $h'(h(u_s)) = u_s$, and (u_s, u_{s+1}) is the only tuple of the form $(u_s, \cdot)$ in $Succ_t^{T'}$, we conclude that $h'(h(u_{s+1})) = u_{s+1}$.

Since the values $u_0, u_1, \ldots, u_{n+1}$ are pairwise distinct, (2.10) implies that $h(u_0), h(u_1), \ldots, h(u_{n+1})$ are pairwise distinct values. However, this is a contradiction to (2.9). Consequently, $\mathbb{M}$ must halt on the empty input.

Altogether, the proof of Theorem 2.33 is complete. □

From the proof of Theorem 2.33, we immediately obtain:

2.34 Corollary.
There is a schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$, where Σ consists of tgds only, such that the following problems are undecidable:

1. *Given a source instance S for M, is there a successful chase sequence on S and Σ?*

2. *Given a source instance S for M, can every chase sequence on S and Σ be extended to a successful chase sequence on S and Σ?*

Moreover, using the proof of Theorem 2.33, it is not hard to obtain Theorem 1, Theorem 6 and Theorem 14 in Deutsch et al. [2008]. These theorems use the following terminology:

2.35 Definition (model, strong and weak universal model)
Let Σ be a set of tgds and egds, and let I be an instance.

- A *model for I and Σ* is a (possibly infinite) instance J such that there is a homomorphism from I to J and $J \models \Sigma$. In particular, J may have relations R^J of infinite size.

- A *strong universal model* for I and Σ is a *finite* model J for I and Σ such that for every (possibly infinite) model K for I and Σ there is a homomorphism from J to K.

- A *weak universal model* for I and Σ is a *finite* model J for I and Σ such that for every *finite* model K for I and Σ there is a homomorphism from J to K.

Note that, if I is a *ground* instance, then a model for I and a set Σ of tgds and egds is a possibly infinite instance $J \supseteq I$ with $J \models \Sigma$. Note also that, if $M = (\sigma_s, \sigma_t, \Sigma)$ is a schema mapping defined by tgds and egds, S is a source instance for M, and T is a target instance for M, then $S \cup T$ is a weak universal model for S and Σ if and only if T is a universal solution for S under M.

2.36 Theorem (Deutsch et al. [2008]).
There is a schema σ such that the following problems are undecidable, where the input consists of an instance I over σ and a set Σ of tgds and egds over σ.

1. *Decide whether there is some complete chase sequence on I and Σ.*
2. *Decide whether all chase sequences on I and Σ can be extended to a complete chase sequence on I and Σ.*
3. *Decide whether a strong universal model for I and Σ exists.*
4. *Decide whether a weak universal model for I and Σ exists.*

In fact, this is even true if I is restricted to be empty.

Proof. Let Σ be the set of all tgds of the schema mapping M_{HALT} constructed in the proof of Theorem 2.33. Given a deterministic Turing machine $\mathbb{M}$ as in the proof of Theorem 2.33, let $\Sigma_{\mathbb{M}}$ be the union of Σ and the set of all tgds of the form $\rightarrow A$, where A is an atom of $S_{\mathbb{M}}$. Let I be the empty instance, that is, $I = \emptyset$. Then, every chase sequence on I and $\Sigma_{\mathbb{M}}$ with result J can be turned into a chase sequence on $S_{\mathbb{M}}$ and Σ with result $S_{\mathbb{M}} \cup J$. The other way round, every chase sequence on $S_{\mathbb{M}}$ and Σ can be turned into a chase sequence on I and $\Sigma_{\mathbb{M}}$ with the same result. Since Σ contains only tgds, every complete chase sequence on $S_{\mathbb{M}}$ and Σ is successful. Thus, as shown in the proof of Theorem 2.33, it is undecidable whether there is a complete chase sequence on I and $\Sigma_{\mathbb{M}}$. Furthermore, it is undecidable whether every chase sequence on I and $\Sigma_{\mathbb{M}}$ can be extended to a complete chase sequence on I and $\Sigma_{\mathbb{M}}$. This proves 1 and 2.

For proving 3 and 4, observe the following. First, there is a universal solution for $S_{\mathbb{M}}$ under M_{HALT} if and only if there is a weak universal model for I and $\Sigma_{\mathbb{M}}$. Second, each model for $S_{\mathbb{M}}$ and Σ is a model for I and $\Sigma_{\mathbb{M}}$. Third, every weak universal model for I and $\Sigma_{\mathbb{M}}$ is a strong universal model for I and $\Sigma_{\mathbb{M}}$. To see this, let J be a weak universal model for I and $\Sigma_{\mathbb{M}}$. Then there is a universal solution for $S_{\mathbb{M}}$ under M_{HALT}, and, as shown in the proof of Theorem 2.33, there is a successful chase sequence on $S_{\mathbb{M}}$ and Σ. By Corollary 2.21 (which holds as well if K is an "infinite instance"), the result J' of this chase sequence is a strong universal model for I and $\Sigma_{\mathbb{M}}$. Thus, let J'' be a model for I and $\Sigma_{\mathbb{M}}$.

Then there is a homomorphism h' from J' to J''. Since J' is a model for I and $\Sigma_{\mathbb{M}}$, there also exists a homomorphism h from J to J'. In particular, the composition of h and h' is a homomorphism from J to J''. This proves that J is a strongly universal model for I and $\Sigma_{\mathbb{M}}$.

Together with the three observations above, Theorem 2.33 implies that it is undecidable whether a strong universal model for I and $\Sigma_{\mathbb{M}}$, or a weak universal model for I and $\Sigma_{\mathbb{M}}$ exists. This proves 3 and 4. □

Interestingly, the Existence-of-Solutions problem considered in Section 1.1.2 is trivial for the schema mapping $M_{\text{HALT}} = (\sigma_{\text{s}}, \sigma_{\text{t}}, \Sigma)$ constructed in the proof of Theorem 2.33. In fact, for every source instance S for M_{HALT}, the target instance T for M_{HALT}, where $R^T = (\text{dom}(S) \cup \{0, 1, \square\})^{\text{ar}(R)}$ for each relation symbol $R \in \sigma_{\text{t}}$, is a solution for S under M_{HALT}.

On the other hand, we have:

2.37 Theorem (Kolaitis et al. [2006]).
There is a schema mapping $M_{\text{emb}} = (\sigma_s, \sigma_t, \Sigma)$, where Σ consists of tgds and egds, such that Existence-of-Solutions(M_{emb}) *is undecidable.*

The schema mapping $M_{\text{emb}} = (\sigma_{\text{s}}, \sigma_{\text{t}}, \Sigma)$ constructed by Kolaitis et al. [2006] for proving Theorem 2.37 has the property that inputs $p\colon X^2 \to X$ to the embedding problem for finite semigroups (recall the definition from Example 1.14) can be encoded as source instances S_p for M_{emb}, and solutions for S_p under M_{emb} correspond to solutions for p with respect to the embedding problem for finite semigroups. As in Example 1.14, σ_{s} consists of a ternary relation symbol R such that a partial function $p\colon X^2 \to X$ for some finite set X can be represented by an instance S_p over σ_{s} with

$$R^{S_p} \;=\; \{(a, b, c) \in X^3 \mid p(a, b) = c\}.$$

Furthermore, σ_{t} consists of a ternary relation symbol $\tilde{R}$ such that a solution $f\colon Y^2 \to Y$ for p with respect to the embedding problem for finite semigroups can be represented as an instance T_f over σ_{t} with

$$\tilde{R}^{T_f} \;=\; \{(a, b, c) \in Y^3 \mid f(a, b) = c\}.$$

To check that a target instance T indeed represents a solution for p with respect to the embedding problem for finite semigroups, Σ consists of the tgds and egds

$$\forall x \forall y \forall z \Big(R(x, y, z) \to \tilde{R}(x, y, z) \Big), \tag{2.11}$$

$$\forall x \forall y \forall z_1 \forall z_2 \Big(\tilde{R}(x, y, z_1) \wedge \tilde{R}(x, y, z_2) \to z_1 = z_2 \Big), \tag{2.12}$$

$$\forall x \forall y \forall z \forall u \forall v \forall w \Big(\tilde{R}(x, y, u) \wedge \tilde{R}(y, z, v) \wedge \tilde{R}(u, z, w) \to \tilde{R}(x, v, w) \Big), \tag{2.13}$$

and for all $i, j \in \{1, 2, 3\}$, the tgd

$$\forall x_1 \forall x_2 \forall x_3 \forall y_1 \forall y_2 \forall y_3 \Big(\tilde{R}(x_1, x_2, x_3) \wedge \tilde{R}(y_1, y_2, y_3) \rightarrow \exists z\, \tilde{R}(x_i, y_j, z) \Big). \quad (2.14)$$

Here, (2.11) and (2.12) state that T represents the graph of a (potentially partial) function $f\colon Y^2 \to Y$, where Y is a finite set and f extends p, (2.13) states that f is associative, and the tgds (2.14) state that f is total.

The following example shows that the reduction from the embedding problem for finite semigroups to Existence-of-Solutions(M_{emb}) described above does not establish that Existence-of-Universal-Solutions(M_{emb}) is undecidable.

2.38 Example
Consider the partial function p encoded by $S_p = \{R(0, 1, 1)\}$. Clearly, the total function $f\colon Y \times Y \to Y$ with $Y := \{0, 1\}$ and $f(x, y) := x + y \bmod 2$ is associative and extends p. However, we show that there is no universal solution for S_p under M_{emb}.

For a contradiction, assume that there is a universal solution T for S_p under M_{emb}. Let n be the largest integer for which there are pairwise distinct values $u_0, u_1, \ldots, u_n$ with $u_0 = 0$, and $(u_i, 1, u_{i+1}) \in \tilde{R}^T$ for each $i \in \{0, 1, \ldots, n-1\}$. Such an n exists, since T is finite. By the choice of n, and since T satisfies (2.14), there is an integer $k \in \{0, 1, \ldots, n\}$ with $(u_n, 1, u_k) \in \tilde{R}^T$. Thus, the sequence

$$(u_k, 1, u_{k+1}),\ (u_{k+1}, 1, u_{k+2}),\ \ldots,\ (u_{n-1}, 1, u_n),\ (u_n, 1, u_k)$$

forms a "cycle" of length $n - k + 1 \leq n + 1$ in $\tilde{R}^T$.

Now consider the target instance T' for M_{emb} with

$$\tilde{R}^{T'} \ = \ \{(a, b, c) \mid a + b = c \bmod n + 2\}.$$

It is easy to verify that T' is a solution for S_p under M_{emb}. Since T is universal, there is a homomorphism h from T to T'. In particular, we must have

$$(h(u_i), 1, h(u_{i+1})) \in R^{T'} \quad \text{for each } i \in \{0, 1, \ldots, n-1\}. \quad (2.15)$$

We now have

$$h(u_i) = i \quad \text{for each } i \in \{0, 1, \ldots, n-1\}. \quad (2.16)$$

This follows in an analogous way as in the proof of Theorem 2.33. First, note that $h(u_0) = 0$, since $u_0 = 0$ is a constant, and homomorphisms are the identity on constants. Now assume that $h(u_i) = i$ for some $i \in \{0, 1, \ldots, n-1\}$. Since

$(h(u_i), 1, h(u_{i+1})) \in R^{T'}$ by (2.15), $h(u_i) = i$, and $(i, 1, i+1)$ is the only tuple in $\tilde{R}^{T'}$ of the form $(i, 1, \cdot)$, it follows that $h(u_{i+1}) = i + 1$.

By (2.16) and $(u_n, 1, u_k) \in \tilde{R}^T$, we have $(n, 1, k) = (h(u_n), 1, h(u_k)) \in \tilde{R}^{T'}$. However, $(n, 1, n+1)$ is the only tuple in $\tilde{R}^{T'}$ of the form $(n, 1, \cdot)$. We conclude that there is no universal solution for S_p under M_{emb}. □

Even more, we can extend M_{emb} to a schema mapping M'_{emb} such that Existence-of-Solutions(M'_{emb}) is undecidable, but Existence-of-Universal-Solutions(M'_{emb}) is trivial, which demonstrates once more the difference between the two problems Existence-of-Solutions and Existence-of-Universal-Solutions:

2.39 Example
Let $M'_{\text{emb}} = (\sigma_{\text{s}}, \sigma'_{\text{t}}, \Sigma')$ be obtained from $M_{\text{emb}} = (\sigma_{\text{s}}, \sigma_{\text{t}}, \Sigma)$ as follows. First, add a new binary relation symbol E to the target schema σ_{t}. Second, add the tgd $\rightarrow E(0, 1)$ to Σ (which ensures that every solution contains the tuple $(0, 1)$ in E), and third, add the tgd

$$\chi_2 \;=\; \forall x \forall y (E(x, y) \rightarrow \exists z\, E(y, z))$$

from Example 2.12 to Σ. Then for every source instance S for M'_{emb}, we have:

- There is a solution for S under M'_{emb} if and only if there is a solution for S under M_{emb}. This is because χ_2 is independent of the other tgds and egds in Σ, and χ_2 is satisfied in any solution T for S under M'_{emb} with $(1, 0) \in E^T$.
- There is no universal solution for S under M'_{emb}. This follows easily from Example 2.12.

Hence, the problem Existence-of-Solutions(M'_{emb}) is undecidable, while the problem Existence-of-Universal-Solutions(M'_{emb}) is trivial. □

2.4 Queries With Inequalities

We conclude this chapter with a short overview on standard techniques and results on computing the certain answers to monotonic queries that are not preserved under homomorphisms. One particular class of such queries is the class of *unions of conjunctive queries with inequalities*:

2.40 Definition (Unions of conjunctive queries with inequalities)
A *union of conjunctive queries with inequalities* over a schema σ is a FO query over σ of the form

$$\varphi(\bar{x}) \;=\; \exists \bar{y}_1\, \psi_1(\bar{x}, \bar{y}_1) \vee \cdots \vee \exists \bar{y}_m\, \psi_m(\bar{x}, \bar{y}_m), \tag{2.17}$$

where for all $i \in \{1, \ldots, m\}$, ψ_i is a FO formula of the form $A_1 \wedge \cdots \wedge A_n$ for some positive integer n, and each A_j is a relational atomic FO formula over σ or an inequality of the form $\neg z = z'$ for variables z, z' that occur in $\bar{x}$ or $\bar{y}_i$. The subformulas $\exists \bar{y}_i \, \psi_i(\bar{x}, \bar{y}_i)$ are called the *conjuncts of* φ.

A *conjunctive query with inequalities* is a union of conjunctive queries with one conjunct.

To compute the certain answers to queries that are not preserved under homomorphisms, Deutsch et al. [2008] proposed *universal solution sets*.[2] This notion is tightly related to the notion of *universal basis* that appeared earlier in the work of Fuxman et al. [2006], and to the *universal solutions* in Afrati et al. [2008]. As the name suggests, universal solution sets are sets of solutions (of course, with special properties). For each class F of mappings, there is a corresponding notion of *F-universal solution set* that is suitable for computing the certain answers to queries, where F is the class of mappings under which these queries are preserved. Note that monotonic queries are preserved under injective homomorphisms, that is, for all instances I, J, for all injective homomorphisms h from I to J, and for all tuples $\bar{t} \in q(I)$, we have $h(\bar{t}) \in q(J)$ (recall that we assume that queries are generic). Hence, we only consider universal solution sets with respect to the set ihom of all injective homomorphisms (i.e., injective mappings that are the identity on constants). Given a set $\mathcal{I}$ of instances over some schema σ, and an instance J over σ, we write $\mathcal{I} \overset{\text{inj}}{\to} J$ if and only if there is an instance $I \in \mathcal{I}$ and an *injective* homomorphism from I to J.

2.41 Definition (ihom-universal solution set)
Let M be a schema mapping, and let S be a source instance for M. An *ihom-universal solution set* for S under M is a finite set $\mathcal{T}$ of solutions for S under M with the following properties:

- $\mathcal{T} \overset{\text{inj}}{\to} T$ for every solution T for S under M.
- There is no $\mathcal{T}' \subsetneq \mathcal{T}$ with $\mathcal{T}' \overset{\text{inj}}{\to} T$ for every solution T for S under M.

As shown by Deutsch et al. [2008], the certain answers to a monotonic query q on a schema mapping M and a source instance S for M can be computed from an ihom-universal solution set $\mathcal{T}$ for S under M by taking the intersection of the answers to q on the solutions in $\mathcal{T}$, and by removing all tuples with nulls:

2.42 Theorem (Deutsch et al. [2008]).
Let M be a schema mapping, let S be a source instance for M, let $\mathcal{T}$ be an

[2] Deutsch et al. [2008] call universal solution sets *universal model sets*. Here, we prefer to replace *model* by *solution*.

ihom-universal solution set for S under M, and let q be a monotonic query over M's target schema. Then,

$$\mathit{cert}(q, M, S) \;=\; \{\bar{t} \mid \bar{t} \in q(T) \text{ for all } T \in \mathcal{T}, \text{ and } \bar{t} \text{ contains only constants}\}.$$

A drawback of ihom-universal solution sets is that they are in general very large, so that it is not possible to compute such an ihom-universal solution set in polynomial time.

2.43 Example
Consider the schema mapping $M = (\{P\}, \{R\}, \Sigma)$, where P is a unary relation symbol, R is a ternary relation symbol, and Σ consists of the single st-tgd

$$\forall x\Big(P(x) \to \exists y \exists z\, R(x, y, z)\Big).$$

Let n be a positive integer, and let S be a source instance for M with $P^S = \{1, \ldots, n\}$. Then it is not hard to verify that there are $\gg 2^n$ many solutions in every ihom-universal model set for M and S. □

On the other hand, it follows from Mądry [2005] that ihom-universal solution sets must be large:

2.44 Theorem (Mądry [2005]).
There is a schema mapping M defined by st-tgds, and a Boolean conjunctive query q with two inequalities such that the following problem is co-NP*-complete: given a source instance S for M, decide whether the certain answers to q on S and M are nonempty.*

What seems to be more practical is to compute a "small" representation of the set of all relevant solutions such as $\mathrm{Core}(M, S)$, and given a monotonic query q over σ_t, compute a set of solutions that is sufficient for computing the certain answers to q on M and S. For example, Fagin et al. [2005a] have shown:

2.45 Theorem (Fagin et al. [2005a]).
Let M be a weakly acyclic schema mapping, and let q be a Boolean query that is the union of conjunctive queries with at most one inequality per disjunct. Then there is a polynomial time algorithm that, given a universal solution for some source instance S under M, computes the certain answers to q on M and S.

Roughly, the algorithm guaranteed by Theorem 2.45 works as follows. Given a universal solution T for some source instance S under M, it first uses the chase to compute a new solution T' for S under M from T. If the chase fails to compute a solution, then the certain answers to q on M and S are nonempty. Otherwise, the certain answers to q on M and S are $q(T')$.

Arenas et al. [2009a] extended Theorem 2.45 to properly restricted classes of unions of conjunctive queries with more than one inequality per disjunct.

3 Justification-Based Approaches to Query Answering

First semantics for query answering in relational data exchange that take into account implicit information in schema mappings and source instances were proposed by Libkin [2006]. These semantics are based on the concept of *CWA-solutions* (the set of all CWA-solutions for a source instance S intuitively corresponds to the set of all possible outcomes of translating S to the target if implicit information in the formalized sense is taken into account). Originally, Libkin introduced CWA-solutions for the case of schema mappings defined by st-tgds,[1] but the definition was extended by Hernich and Schweikardt [2007] to capture the more general case of schema mappings defined by tgds and egds. This chapter introduces CWA-solutions and the corresponding query answering semantics, and discusses generalizations as well as specializations of CWA-solutions considered by Libkin and Sirangelo [2008], Afrati and Kolaitis [2008], and corresponding query answering semantics.

Sections 3.1 and 3.2 are devoted to the definition and basic properties of CWA-solutions. As the name suggests, CWA-solutions are based on the *closed world assumption (CWA)* [Reiter, 1978].[2] The basic idea is that all data in the target must be *justified* in some sense by the schema mapping and the source instance. To put this into more concrete terms, Libkin identified three (rather informal) requirements that a CWA-solution should have, and then formalized them (in the context of schema mappings defined by st-tgds). One of these requirements is that all atoms in a CWA-solution T for a source instance S under a schema mapping M must be *justified* by M and S. For the case that M is defined by st-tgds, this requirement was formalized by Libkin essentially as: there is a st-tgd $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$ of M, and tuples $\bar{u},\bar{v}$ such that $S \models \varphi(\bar{u},\bar{v})$, the atom is one of the atoms in $\psi[\bar{u},\bar{w}]$ for some tuple $\bar{w}$, and T contains all the atoms of $\psi[\bar{u},\bar{w}]$. Hence, a justification for the atom basically consists of a st-tgd of M, and appropriate tuples $\bar{u}$ and $\bar{v}$. For the case that M is defined by tgds and egds, the formalization is a bit more involved: the atom must be "derivable" from S using the tgds of M as "rules" to "produce" new atoms, in a similar way as in logic programming. Furthermore, all atoms

[1] Strictly speaking, the sentences considered by Libkin [2006] have the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$, which differ from st-tgds only in the fact that the body φ can be a (properly restricted) FO formula rather than a conjunction of relational atomic FO formulas.

[2] In Chapter 4, we will consider the original definition of the CWA by Reiter [1978]. For this chapter, the following requirements given by Libkin suffice.

obtained along the way must belong to T, and egds must be respected.

Note that this first requirement for CWA-solutions is very natural in the following context. For simplicity, consider a schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$, where Σ is a set of st-tgds, and let S be a source instance for M. Imagine an algorithm $\mathbb{A}$ that translates S to σ_t by adding atoms to an initially empty target instance T. Then it seems natural to assume that $\mathbb{A}$ adds only atoms to T that can be obtained from S by "applying" a st-tgd in Σ to S. That is, if $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$ is a st-tgd in Σ, and $\bar{u}, \bar{v}$ are assignments for $\bar{x}, \bar{y}$ with $S \models \varphi(\bar{u},\bar{v})$, then the algorithm may add the atoms of $\psi[\bar{u},\bar{w}]$ to T for some tuple $\bar{w}$. So, atoms added to T are justified as indicated above.

The other two requirements for CWA-solutions say that justifications do not "generate" more atoms than necessary, and that all facts (basically, statements without negation) that are true in a CWA-solution follow from the schema mapping and the source instance. Note that these requirements are natural in the context described above, too. Table 3.1 summarizes the three informal requirements for CWA-solutions.

(R1)	Each atom is justified by the schema mapping and the source instance.
(R2)	Justifications for atoms are not "overused": that is, each justification for atoms does not "generate" more atoms than necessary.
(R3)	Each fact in the target instance follows from the schema mapping and the source instance. That is, CWA-solutions contain no "invented" facts.

Table 3.1: Informal requirements for CWA-solutions

Section 3.1 formalizes the first two requirements in Table 3.1. This results in the definition of *CWA-presolutions*, which are solutions that intuitively satisfy the first two requirements in Table 3.1. Section 3.2 then defines *CWA-solutions* by formalizing the third requirement in Table 3.1, and restricting CWA-presolutions to CWA-presolutions that satisfy this third requirement. In Section 3.2, we also identify basic properties of CWA-solutions. In particular, we characterize CWA-solutions as special universal solutions. Even more, CWA-solutions exist if and only if universal solutions exist, and the core of the universal solutions is the unique "minimal" CWA-solution, if it exists.

Sections 3.3 and 3.4 are devoted to the problem of answering queries us-

ing CWA-solutions. In Section 3.3, we first present several semantics for query answering based on CWA-solutions and illustrate them with examples. Afterwards, in Section 3.4, we investigate the complexity of evaluating FO queries under these semantics. It turns out that for queries preserved under homomorphisms, the CWA-solution-based query answering semantics coincide with the certain answers semantics, so that such queries can be evaluated in polynomial time provided a universal solution can be computed in polynomial time, and the query has polynomial time data complexity. However, for more expressive queries, there are simple schema mappings such that the problem of deciding whether a given tuple belongs to the set of answers to such a query may be co-NP-hard or NP-hard, depending on the semantics.

Section 3.5 takes a closer look at generalizations and specializations of CWA-solutions considered by Libkin and Sirangelo [2008] and Afrati and Kolaitis [2008], and corresponding query answering semantics.

Finally, Section 3.6 discusses some limitations of the justification-based approaches for query answering presented in this chapter.

3.1 Definition and Basic Properties of CWA-Presolutions

In this section, we formalize requirements (R1) and (R2) in Table 3.1. This results in the definition of *CWA-presolutions*, which are solutions that intuitively satisfy the first two requirements in Table 3.1. As a warm-up, Section 3.1.1 reviews Libkin's concept of CWA-presolutions for schema mappings defined by st-tgds. The notation used in Section 3.1.1 differs considerably from Libkin's notation, and is chosen carefully to prepare for Section 3.1.2, where we define CWA-presolutions for the more general case of schema mappings defined by tgds and egds. Section 3.1.3 finally presents a game-based characterization of CWA-presolutions that directly reflects the first two requirements of Table 3.1.

3.1.1 Schema Mappings Defined by St-Tgds

This section reviews the concept of CWA-presolutions from Libkin [2006] for schema mappings defined by st-tgds. It serves as a warm-up for the more involved definition of CWA-presolutions in Section 3.1.2. All concepts and results presented in this section are from Libkin [2006]. However, we use a different notation that will be more convenient in Section 3.1.2.

Let $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$ be a schema mapping, where Σ consists of st-tgds, and let S be a source instance for M. Given a target instance T for M and an atom A of T, we say that A *is justified in* T *under* M *and* S if and only if A can be obtained by "applying" a st-tgd in Σ to S, so that all atoms obtained "along the way" belong to T as well. Formally, this is captured as follows:

3.1 Definition (Justification for atoms)
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of st-tgds, let S be a source instance for M, let T be a target instance for M, and let A be an atom of T.

- A *justification for atoms under M and S* is a triple $(\chi, \bar{u}, \bar{v})$ consisting of
  - a st-tgd $\chi \in \Sigma$ of the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x}, \bar{z}))$, and
  - tuples $\bar{u} \in \mathit{Const}^{|\bar{x}|}$ and $\bar{v} \in \mathit{Const}^{|\bar{y}|}$

  such that $S \models \varphi(\bar{u}, \bar{v})$. If M and S are understood from the context, we call $(\chi, \bar{u}, \bar{v})$ just *justification*.
- Let $\mathcal{J}_{M,S}$ be the set of all justifications for atoms under M and S.
- Given $j = (\chi, \bar{u}, \bar{v}) \in \mathcal{J}_{M,S}$, where χ has the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x}, \bar{z}))$, we say that A *is justified by j in T* if and only if there is a tuple $\bar{w} \in \mathit{Dom}^{|\bar{z}|}$ with $A \in \psi[\bar{u}, \bar{w}]$ and $\psi[\bar{u}, \bar{w}] \subseteq T$.
- We say that A *is justified in T under M and S* if and only if there is some $j \in \mathcal{J}_{M,S}$ such that A is justified by j in T.

3.2 Example (Library database restructuring, continued)
Consider the schema mapping $M' = (\sigma_s, \sigma_t, \{\chi'_1, \chi'_2\})$, where σ_s, σ_t, χ'_1 and χ'_2 are defined as in Example 1.15. Furthermore, let S^* be the source instance for M (resp., M') from Example 1.4. Then,

$$\begin{aligned}\mathcal{J}_{M',S^*} = \big\{ &\big(\chi'_1, (\text{“0-201-53771-0”}, \text{“Foundations of Databases”}), ()\big),\\ &\big(\chi'_2, (\text{“0-201-53771-0”}, \text{“Serge Abiteboul”}), ()\big),\\ &\big(\chi'_2, (\text{“0-201-53771-0”}, \text{“Richard Hull”}), ()\big),\\ &\big(\chi'_2, (\text{“0-201-53771-0”}, \text{“Victor Vianu”}), ()\big),\\ &\big(\chi'_2, (\text{“0-201-53082-1”}, \text{“Christos H. Papadimitriou”}), ()\big)\ \big\}.\end{aligned}$$

Note that the instance T presented in Example 1.4 is a solution for S^* under M. The atom

$$\mathit{Books}(\text{“0-201-53771-0”}, \text{“Foundations of Databases”}), \text{“DB”})$$

of T is justified by $(\chi'_1, (\text{“0-201-53771-0”}, \text{“Foundations of Databases”}), ())$ in T. Moreover, the atoms

$$\mathit{AuthorList}(1, \text{“Serge Abiteboul”}) \quad \text{and} \quad \mathit{WrittenBy}(\text{“0-201-53771-0”}, 1)$$

of T are justified by $(\chi'_2, (\text{“0-201-53771-0”}, \text{“Serge Abiteboul”}), ())$ in T. □

Let T be a solution for S under M. Intuitively, T satisfies requirement (R1) if we can label each atom A in T with at least one element $j \in \mathcal{J}_{M,S}$ such that A is justified by j in T. One possible way to model this labeling is to provide a relation R between elements $(\chi, \bar{u}, \bar{v}) \in \mathcal{J}_{M,S}$, where χ has the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$, and tuples $\bar{w} \in \mathit{Dom}^{|\bar{z}|}$ with $\psi[\bar{u},\bar{w}] \subseteq T$: the element $(\chi, \bar{u}, \bar{v})$ then labels the atoms of $\psi[\bar{u},\bar{w}]$. See Figure 3.1 for an illustration.

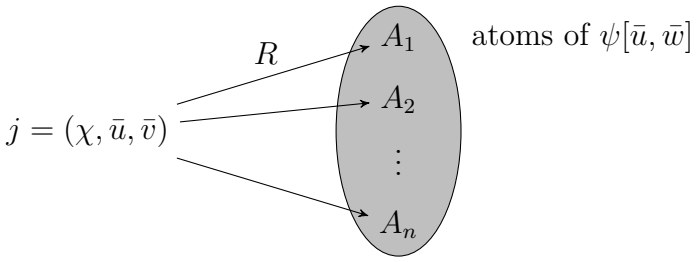


Figure 3.1: Assignment of justifications to atoms

Intuitively, T also satisfies (R2) if R associates each $j \in \mathcal{J}_{M,S}$ with at most one tuple $\bar{w}$. In other words, R is a partial function on $\mathcal{J}_{M,S}$.

Note that we can assume, without loss of generality, that R is a (total) function on $\mathcal{J}_{M,S}$. This is because each justification $j = (\chi, \bar{u}, \bar{v}) \in \mathcal{J}_{M,S}$, where χ has the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$, tells us that $S \models \varphi(\bar{u},\bar{v})$, and thus, since T is a solution for S under M, there must be some tuple $\bar{w}$ with $\psi[\bar{u},\bar{w}] \subseteq T$. In other words, if R does not associate j with some tuple, then we can modify R such that j is associated with $\bar{w}$.

In the following, instead of working with functions that map justifications to tuples of appropriate length, it will be more convenient to work with functions that map pairs in

$$\mathcal{J}^*_{M,S} \;:=\; \{(j,z) \mid j = (\chi,\bar{u},\bar{v}) \in \mathcal{J}_{M,S}, \text{ where } \chi = \forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z})), \text{ and } z \text{ is a variable in } \bar{z}\}$$

to values in Dom. We can think of a mapping $\zeta\colon \mathcal{J}^*_{M,S} \rightarrow \mathit{Dom}$ as an assignment of justifications $j = (\chi, \bar{u}, \bar{v}) \in \mathcal{J}_{M,S}$, where χ has the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$ and $\bar{z} = (z_1,\ldots,z_k)$, to tuples

$$\zeta(j) \;:=\; (\zeta(j,z_1),\ldots,\zeta(j,z_k)),$$

or as a labeling of the atoms in

$$\mathrm{atoms}(\zeta, j) \;:=\; \psi[\bar{u}, \zeta(j)]$$

with j.

We are now ready to give the formal definition of CWA-presolutions:

3.3 Definition (CWA-presolution)
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ is a set of st-tgds, and let S be a source instance for M. A *CWA-presolution* for S under M is a target instance T for S under M such that there is a mapping $\zeta\colon \mathcal{J}^*_{M,S} \to \mathit{Dom}$ with $T = T_{M,S,\zeta}$, where

$$T_{M,S,\zeta} := \bigcup_{j \in \mathcal{J}_{M,S}} \text{atoms}(\zeta, j).$$

An immediate consequence of the definition of CWA-presolutions is that every CWA-presolution for S under M is a solution for S under M.

3.4 Example (A CWA-presolution that is a universal solution)
Recall the schema mapping $M' = (\sigma_s, \sigma_t, \{\chi'_1, \chi'_2\})$ and the source instance S^* for M' from Example 3.2. Let $\zeta\colon \mathcal{J}^*_{M',S^*} \to \mathit{Dom}$ be defined as follows:

justification j	$\zeta(j, z)$
$\big(\chi'_1$, (“0-201-53771-0”, “Foundations of Databases”), $()\big)$	$\bot_1$
$\big(\chi'_2$, (“0-201-53771-0”, “Serge Abiteboul”), $()\big)$	$\bot_4$
$\big(\chi'_2$, (“0-201-53771-0”, “Richard Hull”), $()\big)$	$\bot_5$
$\big(\chi'_2$, (“0-201-53771-0”, “Victor Vianu”), $()\big)$	$\bot_6$
$\big(\chi'_2$, (“0-201-53082-1”, “Christos H. Papadimitriou”), $()\big)$	$\bot_7$

Then we have $T_{M',S^*,\zeta} = T$, where

$$\begin{aligned}
\mathit{BookInfo}^T &= \{(\text{“0-201-53771-0”},\ \text{“Foundations of Databases”},\ \bot_1)\},\\
\mathit{Authors}^T &= \{(\bot_4,\ \text{“Serge Abiteboul”}),\ (\bot_5,\ \text{“Richard Hull”}),\\
&\qquad (\bot_6,\ \text{“Victor Vianu”}),\ (\bot_7,\ \text{“Christos H. Papadimitriou”})\},\\
\mathit{WrittenBy}^T &= \{(\text{“0-201-53771-0”},\ \bot_4),\ (\text{“0-201-53771-0”},\ \bot_5),\\
&\qquad (\text{“0-201-53771-0”},\ \bot_6),\ (\text{“0-201-53082-1”},\ \bot_7)\}.
\end{aligned}$$

□

CWA-presolutions can be characterized in terms of a single universal solution: the *canonical universal solution*. The following definition of the canonical universal solution is equivalent to the definition of the canonical universal solution from Arenas et al. [2004].

3.5 Definition (Canonical universal solution)
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of st-tgds, and let

S be a source instance for M. The *canonical universal solution for S under M* is defined by

$$\mathrm{CanSol}(M, S) \;:=\; T_{M,S,\zeta},$$

where ζ is some injective mapping from $\mathcal{J}^*_{M,S}$ to *Null*.[3]

In Example 3.4, we have $\mathrm{CanSol}(M', S^*) \cong T$. It should be clear from the definition that $\mathrm{CanSol}(M, S)$ is a universal solution for S under M.

3.6 Proposition (Characterization of CWA-presolutions).
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of st-tgds, and let S be a source instance for M. Then for every target instance T for M, the following statements are equivalent.

1. *T is a CWA-presolution for S under M.*
2. *There is a homomorphism h from $\mathrm{CanSol}(M, S)$ to T with the property that $h(\mathrm{CanSol}(M, S)) = T$.*

Proof.

1 $\Longrightarrow$ 2: Suppose that T is a CWA-presolution for S under M. Then there is a mapping $\zeta\colon \mathcal{J}^*_{M,S} \to \mathit{Dom}$ with $T = T_{M,S,\zeta}$. Furthermore, by Definition 3.5, there is an injective mapping $\zeta'\colon \mathcal{J}^*_{M,S} \to \mathit{Null}$ with $\mathrm{CanSol}(M, S) = T_{M,S,\zeta'}$.

Now consider the mapping $h\colon \mathrm{dom}(\mathrm{CanSol}(M, S)) \to \mathrm{dom}(T)$ such that

- $h(\zeta'(j^*)) = \zeta(j^*)$ for all $j^* \in \mathcal{J}^*_{M,S}$, and
- $h(u) = u$ for all $u \in \mathrm{const}(\mathrm{CanSol}(M, S))$.

It is easy to verify that h is a homomorphism from $\mathrm{CanSol}(M, S)$ to T with $h(\mathrm{CanSol}(M, S)) = T$.

2 $\Longrightarrow$ 1: Let h be a homomorphism from $\mathrm{CanSol}(M, S)$ to T with the property that $h(\mathrm{CanSol}(M, S)) = T$. Let $\zeta'\colon \mathcal{J}^*_{M,S} \to \mathit{Null}$ be an injective mapping with $\mathrm{CanSol}(M, S) = T_{M,S,\zeta'}$. Define a mapping $\zeta\colon \mathcal{J}^*_{M,S} \to \mathit{Dom}$ such that for every $j^* \in \mathcal{J}^*_{M,S}$, we have $\zeta(j^*) := h(\zeta'(j^*))$. Then h is a mapping as in the proof of the direction from 1 to 2 above, so that $T = h(\mathrm{CanSol}(M, S)) = T_{M,S,\zeta}$. In particular, T is a CWA-presolution for S under M. □

Proposition 3.6 now makes it easier to verify that a given solution is a CWA-presolution or not.

[3] Note that $\mathrm{CanSol}(M, S)$ is defined only up to isomorphism.

3.7 Example (A CWA-presolution that is no universal solution)
Recall the schema mapping $M' = (\sigma_s, \sigma_t, \{\chi'_1, \chi'_2\})$, the source instance S^* for M', and the solution T for S^* under M' from Example 3.2. Recall that $T = \text{CanSol}(M', S^*)$.

Now consider the solution T' for S^* under M' with

$$\begin{aligned} \mathit{BookInfo}^{T'} &= \{(\text{“0-201-53771-0”}, \text{“Foundations of Databases”}, \text{“DB”})\}, \\ \mathit{Authors}^{T'} &= \{(1, \text{“Serge Abiteboul”}), (2, \text{“Richard Hull”}), \\ &\quad (3, \text{“Victor Vianu”}), (4, \text{“Christos H. Papadimitriou”})\}, \\ \mathit{WrittenBy}^{T'} &= \{(\text{“0-201-53771-0”}, 1), (\text{“0-201-53771-0”}, 2), \\ &\quad (\text{“0-201-53771-0”}, 3), (\text{“0-201-53082-1”}, 4)\}. \end{aligned}$$

It is easy to see that there is a homomorphism h from T to T' with $h(T) = T'$. The homomorphism h just has to map the null $\bot_1$ to “DB”, and the null $\bot_{i+3}$ to i for each $i \in \{1, 2, 3, 4\}$. Thus, by Proposition 3.6, T' is a CWA-presolution for S^* under M'.

Note that T' is *no* universal solution for S^* under M'. For example, there is no homomorphism from T' to T. □

3.8 Example (A universal solution that is no CWA-presolution)
Recall the schema mapping $M' = (\sigma_s, \sigma_t, \{\chi'_1, \chi'_2\})$ and the source instance S^* for M' from Example 3.2. Let $T := \text{CanSol}(M', S^*)$, and let $f\colon \text{dom}(T) \to \mathit{Dom}$ be an injective mapping such that $f(c) = c$ for each constant $c \in \text{const}(T)$, and $f(\bot) \notin \text{dom}(T)$ for each null $\bot \in \text{nulls}(T)$. Then it is easy to see that $T' := T \cup f(T)$ is a universal solution for S^* under M'. However, by Proposition 3.6, T' is no CWA-presolution for S^* under M', since there is no homomorphism h from T to T' such that $h(T) = T'$. □

3.1.2 Schema Mappings Defined by Tgds and Egds

In this section, we extend the definition of CWA-presolutions from Section 3.1.1 to capture schema mappings defined by tgds and egds.

First note that a CWA-presolution for a source instance S under a schema mapping $M' = (\sigma_s, \sigma_t, \Sigma')$, where Σ' consists of st-tgds, is in general no solution for S under a schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$, where Σ is obtained from Σ' by adding t-tgds and egds. In Example 3.4, $T = \text{CanSol}(M', S^*)$ is a CWA-presolution for S^* under M'. However, it is easy to see that T is no solution for S^* under M, since the tgd χ_4 (cf., Example 1.15) is not satisfied. To capture schema mappings with tgds and egds, we thus need to extend the definition of CWA-presolutions.

Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of tgds and egds, and let S be a source instance for M. In a similar way as in Section 3.1.1, we will say that an atom A in a target instance T for M *is justified in* T *under* M *and* S if A can be obtained by "applying" tgds in Σ to S, so that all atoms obtained "along the way" belong to T as well. Note that egds that may be present in Σ are ignored here; they are incorporated later. More precisely (but still informal), A will be called justified in T under M and S if

1. there is a st-tgd $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$ in Σ and tuples $\bar{u},\bar{v}$ such that $S \models \varphi(\bar{u},\bar{v})$, $A \in \psi[\bar{u},\bar{w}]$ for some tuple $\bar{w}$, and $\psi[\bar{u},\bar{w}] \subseteq T$, or

2. there is a t-tgd $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$ in Σ and tuples $\bar{u},\bar{v}$ such that $T \models \varphi(\bar{u},\bar{v})$, $A \in \psi[\bar{u},\bar{w}]$ for some tuple $\bar{w}$, $\psi[\bar{u},\bar{w}] \subseteq T$, and the atoms of $\varphi[\bar{u},\bar{v}]$ are already "justified".

When justifying atoms with t-tgds (according to condition 2 above), we have to take care to avoid "circular justifications": it should not be the case that a t-tgd $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$ and tuples $\bar{u},\bar{v}$ justify a set $\psi[\bar{u},\bar{w}]$ of atoms based on the atoms in $\varphi[\bar{u},\bar{v}]$, while another t-tgd justifies some of the atoms in $\varphi[\bar{u},\bar{v}]$ based on some of the atoms in $\psi[\bar{u},\bar{w}]$. To avoid such "circular justifications", we use a derivation-based approach, adding atoms only if they can be justified based on atoms that have already been justified. We start with a CWA-presolution T_0 for S under $M' = (\sigma_s, \sigma_t, \Sigma')$, where Σ' consists of all st-tgds in Σ, in the sense of Definition 3.3. This derives atoms that can be justified according to condition 1 above. Based on the atoms of T_0 we then derive further atoms that can be justified according to condition 2. This is done by employing a suitably "controlled" version of the chase (cf., Section 2.2.1), which we call ζ-*chase* and which is defined below.

For defining the ζ-chase, we use the following relaxation of the notion of justification (cf., Definition 3.1), which focuses on justifications of atoms in terms of *t-tgds*, according to condition 2 above:

3.9 Definition (potential justification for atoms)
Let Σ be a set of tgds.

- A *potential justification for atoms under* Σ is a triple $(\chi,\bar{u},\bar{v})$ consisting of
  - a tgd $\chi \in \Sigma$ of the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$, and
  - tuples $\bar{u} \in Const^{|\bar{x}|}$ and $\bar{v} \in Const^{|\bar{y}|}$.

  If Σ is understood from the context, we call $(\chi,\bar{u},\bar{v})$ just *potential justification*.

- Let $\mathcal{J}_\Sigma$ be the set of all potential justifications for atoms under Σ.
- Given a schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$ for a set Σ of tgds and egds, we let $\mathcal{J}_M := \mathcal{J}_{\Sigma'}$, where Σ' consists of all t-tgds in Σ.

We will interpret potential justifications $j = (\chi, \bar{u}, \bar{v}) \in \mathcal{J}_M$, where χ has the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x}, \bar{y}) \to \exists\bar{z}\,\psi(\bar{x}, \bar{z}))$, as follows. Suppose that we have already "derived" a target instance T for M (i.e., all atoms of T are already justified) such that $T \models \varphi(\bar{u}, \bar{v})$. Then we can add the atoms of $\psi[\bar{u}, \bar{w}]$ for any tuple $\bar{w} \in Dom^{|\bar{z}|}$ to T, so that all atoms in the resulting instance are justified. In other words, j serves as a *justification* for the atoms of $\psi[\bar{u}, \bar{w}]$ *based on* the atoms of $\varphi[\bar{u}, \bar{v}]$ that *are* already justified.

3.10 Example (Library database restructuring, continued)
Consider the schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$ from Example 1.15. The only t-tgd in Σ is χ_4. Thus,

$$\begin{aligned}\mathcal{J}_M \;=\; \Big\{\; &\big(\chi_4,\, (\text{“0-201-53082-1”}, \bot_7),\, ()\big),\\ &\big(\chi_4,\, (\text{“0-201-53082-1”}, \text{“Christos H. Papadimitriou”}),\, ()\big),\\ &\big(\chi_4,\, (\text{“0-521-30442-3”}, \text{“Wilfrid Hodges”}),\, ()\big),\; \ldots\;\Big\}.\end{aligned}$$

□

Similarly as in Section 3.1.1, we will consider mappings ζ from the set

$$\begin{aligned}\mathcal{J}^*_\Sigma \;:=\; \{(j, z) \mid\; &j = (\chi, \bar{u}, \bar{v}) \in \mathcal{J}_\Sigma, \text{ where } \chi = \forall\bar{x}\forall\bar{y}(\varphi(\bar{x}, \bar{y}) \to \exists\bar{z}\,\psi(\bar{x}, \bar{z})),\\ &\text{and } z \text{ is a variable in } \bar{z}\}\end{aligned}$$

to *Dom* for "labeling atoms with potential justifications". Again, we can view a mapping $\zeta\colon \mathcal{J}^*_\Sigma \to Dom$ as an assignment of potential justifications $j = (\chi, \bar{u}, \bar{v}) \in \mathcal{J}_\Sigma$, where χ has the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x}, \bar{y}) \to \exists\bar{z}\,\psi(\bar{x}, \bar{z}))$ and $\bar{z} = (z_1, \ldots, z_k)$, to tuples

$$\zeta(j) \;:=\; (\zeta(j, z_1), \ldots, \zeta(j, z_k)),$$

or as a labeling of the atoms in

$$\text{atoms}(\zeta, j) \;:=\; \psi[\bar{u}, \zeta(j)]$$

with j.

The ζ-chase is defined for sets of tgds only, since we need it only for the purpose of justifying atoms according to condition 2 mentioned at the beginning of the present Section 3.1.2. Given an instance I over a schema σ, and a set Σ of tgds over σ, the ζ-chase starts—like the chase—with the input instance $I_0 := I$ and proceeds in steps $1, 2, \ldots$. In step $s \geq 1$, an instance I_{s-1} has already been computed. If possible, the ζ-chase then makes a *ζ-tgd chase step*:

3.11 Definition (ζ-tgd chase step)
Let $\zeta\colon \mathcal{J}_\Sigma^* \to \mathit{Dom}$ be defined with respect to a set Σ of tgds over a schema σ. Let I be an instance over σ, let $\chi \in \Sigma$ have the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \to \exists\bar{z}\,\psi(\bar{x},\bar{z}))$, let α be an assignment for φ, and let $j := (\chi, \alpha(\bar{x}), \alpha(\bar{y})) \in \mathcal{J}_\Sigma$.

- We say that χ *applies to* I *with* α *and* ζ if and only if $I \models \varphi(\alpha)$ and $\mathrm{atoms}(\zeta, j) \nsubseteq I$.
- Given an instance J, we write $I \vdash^{\zeta}_{\chi,\alpha} J$ if and only if χ applies to I with α and ζ, and $J = I \cup \mathrm{atoms}(\zeta, j)$.
- We write $I \vdash^{\zeta}_{\Sigma} J$ if and only if there is some $\chi \in \Sigma$ and an appropriate assignment α such that $I \vdash^{\zeta}_{\chi,\alpha} J$.

Otherwise, the ζ-chase stops and outputs I_{s-1}. The possible "runs" of the ζ-chase are formally described by *ζ-chase sequences*:

3.12 Definition (ζ-chase sequence)
Let $\zeta\colon \mathcal{J}_\Sigma^* \to \mathit{Dom}$ be defined with respect to a set Σ of tgds over a schema σ, and let I be an instance over σ.

- A *ζ-chase sequence* on I and Σ is a (finite or infinite) sequence $C = I_0, I_1, \ldots$ of instances over σ such that $I_0 = I$, and for every instance I_s in C with $s \geq 1$ we have $I_{s-1} \vdash^{\zeta}_{\Sigma} I_s$.
- Let $C = (I_0, I_1, \ldots, I_l)$ be a finite ζ-chase sequence on I and Σ. We call I_l the *result* of C, and l the *length* of C.
- A *successful ζ-chase sequence* on I and Σ is a *finite* ζ-chase sequence $C = (I_0, I_1, \ldots, I_l)$ on I and Σ such that there is no instance J over σ with $I_l \vdash^{\zeta}_{\Sigma} J$.

3.13 Example (Library database restructuring, continued)
Consider the schema mapping $M = (\sigma_\mathrm{s}, \sigma_\mathrm{t}, \Sigma)$ and the source instance S^* for M from Example 1.15 and Example 1.4, respectively.

Let T be the target instance for M from Example 3.4, and let $\zeta\colon \mathcal{J}_M^* \to \mathit{Dom}$ be such that on the potential justifications j and the variables z listed in Table 3.2, $\zeta(j, z)$ has the given value.

It is not hard to see that the instance T^* from Example 1.5 is the result of a successful ζ-chase sequence on T and the set $\Sigma' = \{\chi_4\}$ of all t-tgds in Σ. Indeed, we have $T \vdash^{\zeta}_{\chi_4,\alpha} T^*$, where $\alpha(x_1) =$ "0-201-53082-1" and $\alpha(x_2) = \bot_7$. Note also that there is no instance J with $T^* \vdash^{\zeta}_{\Sigma} J$. Therefore, the sequence $C := (T, T^*)$ is a successful ζ-chase sequence on T and Σ'. □

potential justification $j \in \mathcal{J}_M$	z	$\zeta(j, z)$
$(\chi_4, (\text{“0-201-53771-0”}, \bot_4), ())$	z_1	“Foundations of Databases”
$(\chi_4, (\text{“0-201-53771-0”}, \bot_4), ())$	z_2	$\bot_1$
$(\chi_4, (\text{“0-201-53771-0”}, \bot_4), ())$	z_3	“Serge Abiteboul”
$(\chi_4, (\text{“0-201-53771-0”}, \bot_5), ())$	z_1	“Foundations of Databases”
$(\chi_4, (\text{“0-201-53771-0”}, \bot_5), ())$	z_2	$\bot_1$
$(\chi_4, (\text{“0-201-53771-0”}, \bot_5), ())$	z_3	“Richard Hull”
$(\chi_4, (\text{“0-201-53771-0”}, \bot_6), ())$	z_1	“Foundations of Databases”
$(\chi_4, (\text{“0-201-53771-0”}, \bot_6), ())$	z_2	$\bot_1$
$(\chi_4, (\text{“0-201-53771-0”}, \bot_6), ())$	z_3	“Victor Vianu”
$(\chi_4, (\text{“0-201-53082-1”}, \bot_7), ())$	z_1	$\bot_2$
$(\chi_4, (\text{“0-201-53082-1”}, \bot_7), ())$	z_2	$\bot_3$
$(\chi_4, (\text{“0-201-53082-1”}, \bot_7), ())$	z_3	“Christos H. Papadimitriou”

Table 3.2: Definition of ζ for some potential justifications

The following lemma summarizes some basic properties of ζ-chase sequences:

3.14 Lemma.
Let I be an instance over a schema σ, let Σ be a set of tgds over σ, and let $\zeta\colon \mathcal{J}_\Sigma^* \to Dom$.

1. *A successful ζ-chase sequence on I and Σ exists if and only if there is no infinite ζ-chase sequence on I and Σ.*

2. *If C_1, C_2 are successful ζ-chase sequences on I and Σ, then C_1 and C_2 have the same result.*

3. *If J is the result of some ζ-chase sequence on I and Σ with $J \models \Sigma$, then J is the result of some* successful *ζ'-chase sequence on I and Σ, for some $\zeta'\colon \mathcal{J}_\Sigma^* \to Dom$.*

Proof.

Ad 1: If there is no successful ζ-chase sequence on I and Σ, then all ζ-chase sequences on I and Σ must be infinite.

Assume now that $C = (I_0, I_1, \dots, I_l)$ is a successful ζ-chase sequence on I and Σ. For a contradiction, suppose that there is an infinite ζ-chase sequence $C' = (I'_0, I'_1, I'_2, \dots)$ on I and Σ. Note that $I_0 = I'_0$ and that $I_0 \subseteq I_l$. Since in

each ζ-tgd chase step at least one new atom is introduced, we have $I'_0 \subsetneq I'_1 \subsetneq I'_2 \subsetneq \cdots$. Thus, there is a smallest index $i \geq 0$ such that $I'_i \subseteq I_l$ and $I'_{i+1} \nsubseteq I_l$.

Let us assume that $I'_i \vdash^{\zeta}_{\chi,\alpha} I'_{i+1}$ for some $\chi \in \Sigma$, where χ has the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$, and an appropriate assignment α. Then we have $I'_i \models \varphi(\alpha)$ and $\mathrm{atoms}(\zeta, j) \nsubseteq I'_i$, where $j := (\chi, \alpha(\bar{x}), \alpha(\bar{y}))$. Since $I'_i \subseteq I_l$ and $I'_{i+1} = I'_i \cup \mathrm{atoms}(\zeta, j) \nsubseteq I_l$, this implies that $I_l \vdash^{\zeta}_{\chi,\alpha} I_l \cup \mathrm{atoms}(\zeta, j)$. However, this is impossible, because C is successful.

Ad 2: Let $C = (I_0, I_1, \ldots, I_l)$ and $C' = (I'_0, I'_1, \ldots, I'_{l'})$ be successful ζ-chase sequences on I and Σ. For a contradiction, suppose that $I_l \neq I'_{l'}$. Without loss of generality, $I'_{l'} \nsubseteq I_l$. Let $i \geq 0$ be the smallest index such that $I'_{i+1} \nsubseteq I_l$. Then we obtain the desired contradiction in the same way as in part 1 of the proof.

Ad 3: Let $C = (I_0, I_1, \ldots, I_l)$ be a ζ-chase sequence on I and Σ with $I_l = J$. Consider all elements $j = (\chi, \bar{u}, \bar{v}) \in \mathcal{J}_\Sigma$, where χ has the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$, $J \models \varphi(\bar{u},\bar{v})$ and $J \not\models \psi(\bar{u}, \zeta(j))$. For each such element, modify ζ such that $J \models \psi(\bar{u}, \zeta(j))$. This is possible, since $J \models \Sigma$, and therefore, $J \models \psi(\bar{u},\bar{w})$ for some tuple $\bar{w}$. Let ζ' be the resulting mapping. Then C is a successful ζ'-chase sequence on I and Σ with result J . □

We are now ready to give the formal definition of CWA-presolutions.

3.15 Definition (CWA-presolution)
Let $M = (\sigma_\mathrm{s}, \sigma_\mathrm{t}, \Sigma)$ be a schema mapping, where Σ consists of tgds and egds, and let S be a source instance for M. Let Σ_st be the set of all st-tgds in Σ, and let Σ_t be the set of all t-tgds in Σ.

A *CWA-presolution for S under M* is a target instance T for M with the following properties:

1. There is a CWA-presolution T_0 for S under $M' = (\sigma_\mathrm{s}, \sigma_\mathrm{t}, \Sigma_\mathrm{st})$ in the sense of Definition 3.3, and a mapping $\zeta\colon \mathcal{J}^*_{\Sigma_\mathrm{t}} \rightarrow \mathit{Dom}$ such that T is the result of some successful ζ-chase sequence on T_0 and Σ_t.

2. T satisfies the egds in Σ.

Note how condition 1 formalizes the intuitive notion of being derivable from the source instance described at the beginning of the present section. Condition 2 filters out all target instances that can be derived in such a way that egds are respected. In particular, it follows from Definition 3.15 that all CWA-presolutions for S under M are solutions for S under M.

3.16 Example (Library database restructuring, continued)
Consider the schema mapping $M = (\sigma_\mathrm{s}, \sigma_\mathrm{t}, \Sigma)$ and the source instance S^* for M from Example 1.15 and Example 1.4, respectively.

Recall the target instance T for M from Example 3.4, and that it is a CWA-presolution for S^* under the reduced schema mapping M'. In Example 3.13, we have shown that the instance T^* from Example 1.5 is the result of a successful ζ-chase sequence on T and Σ_t, where Σ_t is the set of all t-tgds in Σ, for some mapping $\zeta\colon \mathcal{J}^*_{\Sigma_\mathrm{t}} \to \mathit{Dom}$. Thus, T^* is a CWA-presolution for S^* under M. □

Note that the two notions of CWA-presolutions, from Section 3.1.2 and the present section, coincide on schema mappings with st-tgds. Often, a more convenient "definition" of CWA-presolutions is:

3.17 Proposition.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of tgds and egds, and let S be a source instance for M. Then a CWA-presolution for S under M *is a solution T for S under M such that $S \cup T$ is the result of some successful ζ-chase sequence on S and Σ', where Σ' consists of all tgds in Σ, and $\zeta\colon \mathcal{J}^*_{\Sigma'} \to \mathit{Dom}$.*

Proof. Let Σ_st be the set of all st-tgds in Σ, and let Σ_t be the set of all t-tgds in Σ. Then $\Sigma' = \Sigma_\mathrm{st} \cup \Sigma_\mathrm{t}$.

First assume that T is a CWA-presolution for S under M. Then T is a solution for S under M. Moreover, there is

- a CWA-presolution T_0 for S under $M' = (\sigma_\mathrm{s}, \sigma_\mathrm{t}, \Sigma_\mathrm{st})$, and
- a successful ζ-chase sequence C on T_0 and Σ_t for some mapping $\zeta\colon \mathcal{J}^*_{\Sigma_\mathrm{t}} \to \mathit{Dom}$ such that T is the result of C.

Let $\zeta'\colon \mathcal{J}^*_{M',S} \to \mathit{Dom}$ be such that $T_0 = T_{M',S,\zeta'}$. It is easy to see that $S \cup T$ is the result of a successful ζ''-chase sequence on S and Σ', where $\zeta''\colon \mathcal{J}^*_{\Sigma'} \to \mathit{Dom}$ is such that for all $j \in \mathcal{J}_{M',S}$ we have $\zeta''(j) = \zeta'(j)$, and for all $j \in \mathcal{J}_{\Sigma_\mathrm{t}}$ we have $\zeta''(j) = \zeta(j)$.

Finally assume that T is a solution for S under M, and that $S \cup T$ is the result of a successful ζ-chase sequence on S and Σ' for some $\zeta\colon \mathcal{J}^*_{\Sigma'} \to \mathit{Dom}$. Define $\zeta'\colon \mathcal{J}^*_{M',S} \to \mathit{Dom}$ such that for all $j \in \mathcal{J}_{M',S}$ we have $\zeta'(j) = \zeta(j)$, and define $\zeta''\colon \mathcal{J}^*_{\Sigma_\mathrm{t}} \to \mathit{Dom}$ such that for all $j \in \mathcal{J}_{\Sigma_\mathrm{t}}$ we have $\zeta''(j) = \zeta(j)$. Then it is easy to see that $T_0 := T_{M',S,\zeta'}$ is a CWA-presolution for S under M', and that T is the result of a successful ζ''-chase sequence on T_0 and Σ_t. □

In the following Section 3.1.3, we give an alternative definition of CWA-presolutions in terms of a *game*.

3.1.3 A Game-Based Characterization

CWA-presolutions can be characterized alternatively in terms of a *game* which directly reflects requirements (R1) and (R2) in Table 3.1.

Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of tgds and egds, let S be a source instance for M, and let T be a target instance for M. The game, denoted by $\mathcal{D}(M, S, T)$, is played by two players, the *verifier* and the *falsifier*. The verifier's goal is to show that T satisfies requirements (R1) and (R2) with respect to S and M, whereas the falsifier's goal is to show the converse. The game has at most $|\text{atoms}(T)| + 1$ rounds.

In round 0, the verifier fixes the two sets $\mathcal{A}_1 := \text{atoms}(T)$ and $\mathcal{J}_1 := \mathcal{J}_{\Sigma'}$, where Σ' is the set of all tgds in Σ (recall Definition 3.9), and picks a linear order $\preceq$ on $\mathcal{A}_1$. Intuitively, $\mathcal{A}_1$ is the set of all atoms that need to be justified, and $\mathcal{J}_1$ is the set of all potential justifications that may be used for this purpose. The linear order $\preceq$ determines that for justifying an atom A, only atoms that either belong to S or are smaller than A with respect to $\preceq$ can be used. In particular, this ensures that there are no "circular justifications".

The game proceeds in rounds $1, 2, \ldots$. In each round $i \geq 1$, the falsifier begins by picking an atom A in $\mathcal{A}_i$, or loses if $\mathcal{A}_i$ is empty. Then, the verifier has to justify A by picking a potential justification $(\chi, \bar{u}, \bar{v}) \in \mathcal{J}_i$, where χ has the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists\bar{z}\, \psi(\bar{x}, \bar{z}))$, and a tuple $\bar{w} \in \mathit{Dom}^{|\bar{z}|}$ such that:

- $S \cup T \models \varphi(\bar{u}, \bar{v})$, $T \models \psi(\bar{u}, \bar{w})$ and $A \in \psi[\bar{u}, \bar{w}]$, and
- for each atom A' of $\varphi[\bar{u}, \bar{v}]$, either A' is an atom of S, or $A' \prec A$.

If this is not possible, the verifier loses. Otherwise, the game proceeds with round $i+1$, where $\mathcal{A}_{i+1}$ consists of all atoms in $\mathcal{A}_i$ that do not occur in $\psi[\bar{u}, \bar{w}]$, and $\mathcal{J}_{i+1} := \mathcal{J}_i \setminus \{(\chi, \bar{u}, \bar{v})\}$.

Note that by removing all atoms of $\psi[\bar{u}, \bar{w}]$ from $\mathcal{A}_i$, we ensure that these atoms do not need to be justified again (they are already justified by $(\chi, \bar{u}, \bar{v})$). By removing $(\chi, \bar{u}, \bar{v})$ from the set $\mathcal{J}_i$, we ensure that every potential justification is used at most once (this corresponds to requirement (R2)).

A winning strategy for the verifier (resp., falsifier) in the game $\mathcal{D}(M, S, T)$ is a strategy that allows the verifier (resp., falsifier) to win the game regardless of which moves the falsifier (resp., verifier) makes during the game. Note that either the verifier or the falsifier has a winning strategy in $\mathcal{D}(M, S, T)$.

3.18 Example (Library database restructuring, continued)
Consider the schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$ and the source instance S^* for M from Example 1.15 and Example 1.4, respectively.

Let T^* be the target instance for M from Example 1.5. Then the verifier has a winning strategy in $\mathcal{D}(M, S^*, T^*)$. Indeed, it is not difficult to see that the verifier can win $\mathcal{D}(M, S^*, T^*)$ by picking, in round 0, a linear order $\preceq$ on $\text{atoms}(T^*)$ such that the atom $\mathit{BookInfo}(\text{"0-201-53082-1"}, \bot_2, \bot_3)$ is the largest atom with respect to $\preceq$. Similarly, the verifier has a winning

strategy in $\Im(M, S^*, T)$, where T is the solution for S^* under M from Example 1.4. Here, the verifier can pick a linear order $\preceq$ on atoms(T) such that *BookInfo*("0-201-53082-1", "Computational Complexity", "CC") is the largest atom with respect to $\preceq$.

On the other hand, consider the solutions T_1 and T_2 for S^* under M with

$$\begin{aligned} T_1 &= T^* \cup \{\ \mathit{WrittenBy}(\text{"0-521-30442-3"}, \bot)\ \}, \\ T_2 &= T^* \cup \{\ \mathit{BookInfo}(\text{"0-201-53082-1"}, \bot, \bot')\ \}. \end{aligned}$$

Then the falsifier has a winning strategy in both games, $\Im(M, S^*, T_1)$ and $\Im(M, S^*, T_2)$. To win the game $\Im(M, S^*, T_1)$, the falsifier simply picks the atom *WrittenBy*("0-521-30442-3", $\bot$) in round 1. Then the verifier cannot justify this atom, because the only potential justification that could justify this atom would have to be of the form (χ_2, ("0-521-30442-3", $\cdot$), ()). However, to "use" such a potential justification, $\mathit{Authors}^{S^*}$ would have to contain a tuple of the form ("0-521-30442-3", $\cdot$), which is not the case.

Furthermore, to win the game $\Im(M, S^*, T_2)$, the falsifier can choose the atom $A_1 := \mathit{BookInfo}(\text{"0-201-53082-1"}, \bot_2, \bot_3)$ in round 1, and the atom $A_2 := \mathit{BookInfo}(\text{"0-201-53082-1"}, \bot, \bot')$ in round 2. Both atoms, A_1 and A_2, can only be justified using the potential justification $j := (\chi_4, (\text{"0-201-53082-1"}, \bot_7), ())$. Thus, the verifier would have to choose j in round 1 and round 2, which is impossible. □

3.19 Proposition (Game characterization of CWA-presolutions). *Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of tgds and egds, and let S be a source instance for M. For every target instance T for M, the following statements are equivalent:*

1. *T is a CWA-presolution for S under M.*
2. *T is a solution for S under M, and the verifier has a winning strategy in the game $\Im(M, S, T)$.*

Proof.

1 $\Longrightarrow$ 2: Suppose that T is a CWA-presolution for S under M. In particular, T is a solution for S under M. It remains to show that the verifier has a winning strategy in the game $\Im(M, S, T)$.

Let Σ' be the set of all tgds in Σ. By Proposition 3.17, $S \cup T$ is the result of a successful ζ-chase sequence $C = (I_0, I_1, \ldots, I_l)$ on S and Σ', for some mapping $\zeta\colon \mathcal{J}^*_{\Sigma'} \to \mathit{Dom}$. From C, we obtain a winning strategy for the verifier in the game $\Im(M, S, T)$ as follows.

The strategy will be such that:

> If the verifier picks a justification j and a tuple $\bar{w}$ in some round, then $\bar{w} = \zeta(j)$. (*)

In round 0, the verifier fixes the sets $\mathcal{A}_1$ and $\mathcal{J}_1$ as described above, and picks a linear order $\preceq$ on $\mathcal{A}_1$ such that for all atoms $A, A' \in \mathcal{A}_1$, where $A \in I_{k-1}$ and $A' \in I_k \setminus I_{k-1}$ for some $k \in \{1, \ldots, l\}$, we have $A \prec A'$.

Let $i \geq 1$ be such that $\mathcal{A}_i \neq \emptyset$, and assume that in each of the rounds $i' \in \{1, \ldots, i-1\}$, the verifier played according to (*). Then round i proceeds as follows. Let $A \in \mathcal{A}_i$ be the choice of the falsifier in round i. Let k be the smallest integer in $\{0, 1, \ldots, l\}$ for which $A \in I_k$. Note that $k \geq 1$, since $I_0 = S$, and A is an atom of T.

Assume that $I_{k-1} \vdash^{\zeta}_{\chi,\alpha} I_k$ for some tgd $\chi \in \Sigma'$ of the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists\bar{z}\, \psi(\bar{x}, \bar{z}))$, and an assignment α. For $j := (\chi, \alpha(\bar{x}), \alpha(\bar{y}))$, we then have

- $S \cup T \models \varphi(\alpha)$, $T \models \psi(\alpha(\bar{x}), \zeta(j))$ and $A \in \psi[\alpha(\bar{x}), \zeta(j)]$, and
- for each atom A' of $\varphi[\alpha]$, either A' is an atom of S (if χ is a st-tgd), or $A' \prec A$ (if χ is a t-tgd, by construction of $\preceq$).

Since the verifier plays according to (*), we have $j \in \mathcal{J}_i$ (otherwise, if $j \notin \mathcal{J}_i$, then all atoms of $\text{atoms}(\zeta, j)$ would already be "justified", that is, $A \notin \mathcal{A}_i$). Thus, the verifier can choose j and the tuple $\bar{w} := \zeta(j)$ in round i.

2 $\Longrightarrow$ 1: Assume that T is a solution for S under M, and that the verifier has a winning strategy in $\mathfrak{I}(M, S, T)$. Let Σ' be the set of all tgds in Σ. To show that T is a CWA-presolution for S under M, it suffices to find, by Proposition 3.17, a mapping $\zeta\colon \mathcal{J}^*_{\Sigma'} \rightarrow \mathit{Dom}$ such that $S \cup T$ is the result of a successful ζ-chase sequence on S and Σ'.

A *play* in $\mathfrak{I}(M, S, T)$ is a sequence $p = (p_1, \ldots, p_n)$, where $n \geq 0$ is the number of rounds of the play, and each p_i is a tuple of the form $(A, j, \bar{w})$, where A is the atom chosen by the falsifier in round i, and j and $\bar{w}$ are the justification and the tuple chosen by the verifier in round i. With each play p in $\mathfrak{I}(M, S, T)$, we associate a set $\mathcal{A}(p) \subseteq \text{atoms}(T)$ as follows.

- If $p = ()$, then $\mathcal{A}(p) := \text{atoms}(T)$.
- If $p = (p_1, \ldots, p_{n+1})$, where $p_{n+1} = (A, j, \bar{w})$, $j = (\chi, \bar{u}, \bar{v})$ and χ has the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists\bar{z}\, \psi(\bar{x}, \bar{z}))$, then $\mathcal{A}(p) := A(p_1, \ldots, p_n) \setminus \psi[\bar{u}, \bar{w}]$.

Hence, for each play $p = (p_1, \ldots, p_n)$ in $\mathfrak{I}(M, S, T)$, $\mathcal{A}(p)$ coincides with the set $\mathcal{A}_{n+1}$ of atoms at the beginning of round $n+1$. Note that the verifier wins p if and only if $\mathcal{A}(p) = \emptyset$.

Consider a play $p = (p_1, \ldots, p_n)$ in $\mathfrak{D}(M, S, T)$, where the verifier wins. For every $i \in \{1, \ldots, n\}$, let $p_i = (A_i, j_i, \bar{w}_i)$, and assume that the falsifier plays such that A_i is the minimal atom in $\mathcal{A}(p_1, \ldots, p_{i-1})$ with respect to $\preceq$. Choose a mapping $\zeta\colon \mathcal{J}^*_{\Sigma'} \to \mathit{Dom}$ such that for every $i \in \{1, \ldots, n\}$ we have $\bar{w}_i = \zeta(j_i)$. This is possible, because each potential justification j_i occurs once in p.

We now show that $S \cup T$ is the result of a ζ-chase sequence on S and Σ'. To this end, we prove the following by induction on $i \in \{0, 1, \ldots, n\}$:

$$\text{There is a } \zeta\text{-chase sequence } C_i \text{ on } S \text{ and } \Sigma' \text{ with result } (S \cup T) \setminus \mathcal{A}(p_1, \ldots, p_i). \tag{$*$}$$

For $i = 0$, we can choose $C_0 = (S)$. Now suppose that $(*)$ is true for some $i < n$. That is, there is a ζ-chase sequence $C_i = (I_0, I_1, \ldots, I_l)$ on S and Σ' with $I_l = (S \cup T) \setminus \mathcal{A}(p_1, \ldots, p_i)$. Assume that $j_{i+1} = (\chi, \bar{u}, \bar{v})$, where χ has the form $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$. Then,

- $S \cup T \models \varphi(\bar{u}, \bar{v})$, $T \models \psi(\bar{u}, \bar{w}_{i+1})$ and $A \in \psi[\bar{u}, \bar{w}_{i+1}]$, and
- for each atom A' of $\varphi[\bar{u}, \bar{v}]$, either A' is an atom of S, or $A' \prec A$.

If $\psi[\bar{u}, \bar{w}_{i+1}] \subseteq I_l$, then we can choose $C_{i+1} := C_i$. In the following, we consider the case that $\psi[\bar{u}, \bar{w}_{i+1}] \nsubseteq I_l$.

Note that $\varphi[\bar{u}, \bar{v}] \subseteq I_l$. To see this, let A' be an atom of $\varphi[\bar{u}, \bar{v}]$. If A' is an atom of S, then $A' \in I_l$ since $S \subseteq I_l$. Otherwise, $A' \prec A$. If A' would not be an atom of I_l, then $A' \in \mathcal{A}(p_1, \ldots, p_i)$. But this is impossible, since by the construction of p, A is the minimal atom in $\mathcal{A}(p_1, \ldots, p_i)$ with respect to $\preceq$.

Since $\varphi[\bar{u}, \bar{v}] \subseteq I_l$, $\psi[\bar{u}, \bar{w}_{i+1}] \nsubseteq I_l$, and $\bar{w}_{i+1} = \zeta(j_{i+1})$ by the construction of ζ, the sequence $C_{i+1} := (I_0, I_1, \ldots, I_l, I_l \cup \psi[\bar{u}, \bar{w}_{i+1}])$ is a ζ-chase sequence on S and Σ' with result $I_l \cup \psi[\bar{u}, \bar{w}_{i+1}] = (S \cup T) \setminus \mathcal{A}(p_1, \ldots, p_{i+1})$. This completes the induction step.

To complete the proof, observe that by Lemma 3.14(3), it follows immediately that $S \cup T$ is the result of a *successful* ζ'-chase sequence on S and Σ', for some mapping $\zeta'\colon \mathcal{J}^*_{\Sigma'} \to \mathit{Dom}$. □

3.2 Definition and Basic Properties of CWA-Solutions

Note that a CWA-presolution for a source instance S under a schema mapping M defined by tgds and egds may imply certain "facts" that do not "follow" from S and M. Recall, for instance, the setting in Example 1.4. Here, the solution T for S^* under M is a CWA-presolution for S^* under M, as shown by Example 3.18 and Proposition 3.19. In particular, T tells us that "Christos H. Papadimitriou" is author of the book "Computational Complexity". However,

this fact intuitively does not follow from S^* and M. The third requirement for CWA-solutions, Requirement (R3), ensures that such "invented" facts do not occur in CWA-solutions.

Let us now show how to formalize this third requirement. Formally, a *fact* F over a schema σ is a FO sentence over σ of the form $\exists \bar{z}\, \varphi(\bar{z})$, where φ is a conjunction of relational atomic FO formulas. For example, recall the setting in Example 1.4. Then the fact

$$\begin{aligned} \exists z_1 \exists z_2 \exists z_3 \big(& \mathit{AuthorList}(z_1, \text{“Christos H. Papadimitriou”}) \wedge \\ & \mathit{WrittenBy}(z_2, z_1) \wedge \\ & \mathit{BookInfo}(z_2, \text{“Computational Complexity”}, z_3) \big) \end{aligned} \tag{3.1}$$

tells us that "Christos H. Papadimitriou" is author of the book "Computational Complexity". We say that a fact $F = \exists \bar{z}\, \varphi(\bar{z})$ over a schema σ is true in an instance I over σ if and only if $I \models F$. Libkin [2006] formalized CWA-solutions by requiring that each fact that is true in a CWA-solution is true in the canonical universal solution. For the more general case of schema mappings defined by tgds and egds, a solution like the canonical universal solution may not exist. Here, we formalize CWA-solutions by requiring that each fact that is true in a CWA-solution can be inferred from the source instance and the schema mapping. Note that this directly corresponds to Requirement (R3).

3.20 Definition (CWA-solution)
Let $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$ be a schema mapping, where Σ consists of tgds and egds, and let S be a source instance for M.

- A *CWA-solution* for S under M is a CWA-presolution T for S under M such that each fact that is true in T is also true in every solution for S under M.
- The set of all CWA-solutions for S under M is denoted by $\mathit{sol}_{\mathrm{CWA}}(M, S)$.

The following theorem characterizes CWA-solutions as CWA-presolutions that are universal. Note that in the context of schema mappings defined by st-tgds, if we use the definition of CWA-solutions from Libkin [2006]—that a CWA-solution is a CWA-presolution that has a homomorphism into the canonical universal solution—this follows immediately from the fact that the canonical universal solution is a universal solution.

3.21 Theorem (Characterization of CWA-solutions).
Let M be a schema mapping, where Σ consists of tgds and egds, and let S be a source instance for M. Then, for every target instance T for M, the following statements are equivalent:

1. *T is a CWA-solution for S under M.*

2. *T is a universal solution for S under M, and T is a CWA-presolution for S under M.*

Proof.

1 $\Longrightarrow$ 2: Let T be a CWA-solution for S under M. In particular, T is a CWA-presolution for S under M. To prove 2, it suffices therefore to show that T is a universal solution for S under M.

Let $\bot_1, \ldots, \bot_k$ be an enumeration of all nulls that occur in T (without repetition). Consider the fact $F_T := \exists \bar{z}\, \varphi_T(\bar{z})$, where $\bar{z} = (z_1, \ldots, z_k)$, and $\varphi_T(\bar{z})$ is the conjunction of all atomic formulas $R(\bar{u})$ obtained from an atom $R(\bar{t})$ of T by replacing each null $\bot_i$ in $\bar{t}$ with z_i. Clearly, F_T is true in T. Since T is a CWA-solution for S under M, Definition 3.20 implies that F_T is true in every solution T' for S under M. We use this to show that for every solution T' for S under M, there is a homomorphism from T to T', which proves that T is a universal solution for S under M.

Let T' be an arbitrary solution for S under M. Since F_T is true in T', there is a tuple $\bar{v} = (v_1, \ldots, v_k) \in \mathit{Dom}^k$ such that $T' \models \varphi_T(\bar{v})$. In other words, the mapping $h\colon \mathrm{dom}(T) \to \mathrm{dom}(T')$, where $h(\bot_i) = v_i$ for each $i \in \{1, \ldots, k\}$, and $h(c) = c$ for each constant $c \in \mathrm{const}(T)$, is a homomorphism from T to T'.

2 $\Longrightarrow$ 1: Suppose that T is a universal solution for S under M, and that T is a CWA-presolution for S under M. To show that T is a CWA-solution, it remains to show that each fact that is true in T, is true in every solution T' for S under M.

Let $F = \exists \bar{z}\, \varphi(\bar{z})$ be a fact that is true in T, and let T' be an arbitrary solution for S under M. Then there is a tuple $\bar{v} \in \mathit{Dom}^{|\bar{z}|}$ such that $T \models \varphi(\bar{v})$. Since T is a universal solution for S under M, there is a homomorphism h from T to T'. Furthermore, since φ is preserved under homomorphisms, we have $T' \models \varphi(h(\bar{v}))$. Thus, F is true in T'. □

3.22 Example (Library database restructuring revisited)
Consider the schema mapping $M = (\sigma_\mathrm{s}, \sigma_\mathrm{t}, \Sigma)$ and the source instance S^* for M from Example 1.15 and Example 1.4, respectively.

Recall that the target instance T^* for M from Example 1.5 is the core of the universal solutions for S^* under M. Furthermore, as shown in Example 3.16, T^* is a CWA-presolution for S^* under M. Thus, by Theorem 3.21, T^* is a CWA-solution for S^* under M.

It is not hard to see that the target instance T' for M with

$$\begin{aligned} \mathit{BookInfo}^{T'} &= \mathit{BookInfo}^{T^*} \cup \{\, (\text{“0-201-53771-0”},\ \bot_8,\ \bot_9), \\ &\qquad\qquad (\text{“0-201-53771-0”},\ \bot_{10},\ \bot_{11}), \\ &\qquad\qquad (\text{“0-201-53771-0”},\ \bot_{12},\ \bot_{13})\,\}, \\ \mathit{AuthorList}^{T'} &= \mathit{AuthorList}^{T^*}, \quad \text{and} \\ \mathit{WrittenBy}^{T'} &= \mathit{WrittenBy}^{T^*} \end{aligned}$$

is another CWA-solution for S^* under M. Moreover, it is not hard to see that for every CWA-solution T'' for S^* under M there is a homomorphism h from T' to T'' such that $h(T') = T''$.

Consider now the instance T from Example 1.4, which, as shown by Example 3.18 and Proposition 3.19, is a CWA-presolution for S^* under M. However, T is no CWA-solution for S^* under M, since (3.1) is a fact that is true in T, but not in T^*.

Next, consider the instance T_2 from Example 3.18. It is easy to see that T_2 is a universal solution for S^* under M: there is a homomorphism from T_2 to T^* that simply maps $\bot$ to $\bot_2$, and $\bot'$ to $\bot_3$. However, it follows from Example 3.18 and Proposition 3.19 that T_2 is no CWA-presolution (and therefore no CWA-solution) for S^* under M. □

By Theorem 3.21, CWA-solutions are particular universal solutions. The following theorem shows that the “smallest” universal solution—the core of the universal solutions—is one of these CWA-solutions. In particular, it shows that the core of the universal solutions is the “smallest” CWA-solution. The corresponding result for CWA-solutions in the context of schema mappings defined by st-tgds has been obtained by Libkin [2006].

3.23 Theorem.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of tgds and egds, and let S be a source instance for M such that $\mathrm{Core}(M, S)$ *exists. Then,*

1. $\mathrm{Core}(M, S)$ *is a CWA-solution for S under M.*

2. *If T is a CWA-solution for S under M, then* $\mathrm{Core}(M, S)$ *is isomorphic to a subinstance of T.*

Proof. It suffices to prove 1. Then 2 follows immediately from Theorem 2.6(3) and the fact that, by Theorem 3.21, every CWA-solution for S under M is a universal solution for S under M. Since $\mathrm{Core}(M, S)$ is a universal solution for S under M, it remains to show, by Theorem 3.21 and Proposition 3.17, that

$S \cup \mathrm{Core}(M, S)$ is the result of a successful ζ-chase sequence C on S and Σ', where Σ' is the set of all tgds in Σ, and $\zeta\colon \mathcal{J}^*_{\Sigma'} \to \mathit{Dom}$.

For $i = 0, 1, \ldots$, let us inductively construct mappings $\zeta_i\colon \mathcal{J}^*_{\Sigma'} \to \mathit{Dom}$ and ζ_i-chase sequences C_i on S and Σ' such that the result of C_i is a subinstance of $S \cup \mathrm{Core}(M, S)$. Let $\zeta_0\colon \mathcal{J}^*_{\Sigma'} \to \mathit{Dom}$ be arbitrary, and $C_0 := (S)$. Assume now that $C_i = (I_0, I_1, \ldots, I_l)$ is a ζ_i-chase sequence on S and Σ' with $I_l \subseteq S \cup \mathrm{Core}(M, S)$. If $I_l \models \Sigma$, then stop. Otherwise, if $I_l \not\models \Sigma$, there is some $\chi \in \Sigma$ with $I_l \not\models \chi$. Note that χ is not an egd, since $I_l \subseteq S \cup \mathrm{Core}(M, S)$, and $S \cup \mathrm{Core}(M, S)$ satisfies all egds in Σ. Thus, χ is a tgd.

Let χ be of the form $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$. Since $I_l \not\models \chi$, there are tuples $\bar{u}$ and $\bar{v}$ such that $I_l \models \varphi(\bar{u}, \bar{v})$, and there is no tuple $\bar{w}$ with $I_l \models \psi(\bar{u}, \bar{w})$. On the other hand, we have $S \cup \mathrm{Core}(M, S) \models \chi$, so that there is a tuple $\bar{w}$ with $S \cup \mathrm{Core}(M, S) \models \psi(\bar{u}, \bar{w})$. Let $\zeta_{i+1}\colon \mathcal{J}^*_{\Sigma'} \to \mathit{Dom}$ be the mapping such that for $j := (\chi, \bar{u}, \bar{v})$ we have $\zeta_{i+1}(j) = \bar{w}$, and for all $j' \in \mathcal{J}_{\Sigma'}$ with $j' \neq j$ we have $\zeta_{i+1}(j') = \zeta_i(j')$. It is easy to verify that C_i is a ζ_{i+1}-chase sequence on S and Σ'. Moreover, we have $I_l \vdash^{\zeta_{i+1}}_{\chi, \bar{u}, \bar{v}} I_{l+1}$ for

$$I_{l+1} \;:=\; I_l \cup \psi[\bar{u}, \bar{w}] \;\subseteq\; S \cup \mathrm{Core}(M, S).$$

Consequently, $C_{i+1} := (I_0, I_1, \ldots, I_{l+1})$ is a ζ_{i+1}-chase sequence on S and Σ' with result $I_{l+1} \subseteq S \cup \mathrm{Core}(M, S)$. This completes the induction step.

Since $S \cup \mathrm{Core}(M, S)$ is finite, and each step in a ζ-chase sequence on S and Σ' introduces at least one new atom, there must be some integer n such that the result J of C_n satisfies Σ. By Lemma 3.14(3), J is the result of a *successful* ζ-chase sequence on S and Σ', for some $\zeta\colon \mathcal{J}^*_{\Sigma'} \to \mathit{Dom}$. Note also that $T := J \setminus S$ is a solution for S under M. Even more, T is a universal solution for S under M, since $T \subseteq \mathrm{Core}(M, S)$. Thus, we must have $T = \mathrm{Core}(M, S)$. □

In particular, this implies:

3.24 Corollary.
For every schema mapping M defined by tgds and egds, and every source instance S for M, the following statements are equivalent:

1. *There exists a CWA-solution for S under M.*
2. *There exists a universal solution for S under M.*
3. $\mathrm{Core}(M, S)$ *exists.*

Proof. If there is a CWA-solution for S under M, then this CWA-solution is a universal solution for S under M by Theorem 3.21. If there is a universal solution for S under M, then $\mathrm{Core}(M, S)$ exists. Finally, if $\mathrm{Core}(M, S)$ exists, then $\mathrm{Core}(M, S)$ is a CWA-solution for S under M by Theorem 3.23(1). □

Together with Theorem 3.23(1) and Theorem 2.32, this yields:

3.25 Corollary.
Let M be a weakly acyclic schema mapping. Then there is a polynomial time algorithm that takes a source instance S for M as input, and either outputs a CWA-solution for S under M if such a solution exists, or tells us that such a solution does not exist.

Furthermore, by Theorem 2.33, we have:

3.26 Corollary.
There is a schema mapping M defined by tgds such that the following problem is undecidable: given a source instance S for M, is there a CWA-solution for S under M?

In some cases, we even have a CWA-solution T that is *maximal* in the sense that for every CWA-solution T' for S under M there is a homomorphism h from T to T' with $h(T) = T'$. Under schema mappings defined by st-tgds, we have:

3.27 Proposition (Libkin [2006]).
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of st-tgds, and let S be a source instance for M. Then $\mathrm{CanSol}(M, S)$ is the unique maximal CWA-solution for S under M (up to isomorphism).

Proof. Clearly, $\mathrm{CanSol}(M, S)$ is both, a universal solution for S under M, and a CWA-presolution for S under M. Theorem 3.21 thus implies that $\mathrm{CanSol}(M, S)$ is a CWA-solution for S under M. Furthermore, Proposition 3.6 implies that $\mathrm{CanSol}(M, S)$ is the unique maximal CWA-solution for S under M (up to isomorphism). □

Furthermore, Proposition 3.27 can be extended to a slightly larger class of schema mappings:

3.28 Proposition.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of st-tgds and egds, or all tgds in Σ are full. Let S be a source instance for M such that there is a CWA-solution for S under M. Then there is a unique maximal CWA-solution T for S under M (up to isomorphism).

Proof. First consider the case that all tgds of Σ are full. Then there is a unique CWA-solution T for S under $M' = (\sigma_\mathrm{s}, \sigma_\mathrm{t}, \Sigma')$, where Σ' is the set of all tgds in Σ. Since there exists a CWA-solution for S under M, we have $T \models \Sigma$. Therefore, T is the unique maximal CWA-solution for S under M.

In the following, we consider the case that Σ consist of st-tgds and egds. Let $M' = (\sigma_s, \sigma_t, \Sigma_{st})$, where Σ_{st} is the set of all st-tgds in Σ, and let $T_0 := \mathrm{CanSol}(M', S)$. Consider a complete chase sequence $C = (T_0, T_1, \ldots, T_l)$ on T_0 and $\Sigma \setminus \Sigma_{st}$. Then C is successful: otherwise there would be no solution for S under M, contradicting the proposition's hypothesis. Let $T := T_l$. Note that T is a CWA-solution for S under M. Moreover, T is unique up to isomorphism (see, e.g., Beeri and Vardi [1984]). It remains to prove T is maximal.

To this end, let T' be a CWA-solution for S under M. We show by induction on i that for every $i \in \{0, 1, \ldots, l\}$, there is a homomorphism h from T_i to T' with $h(T_i) = T'$. Note that T' is a CWA-presolution for S under M'. Thus, by Proposition 3.6(2), there is a homomorphism h from T_0 to T' with $h(T_0) = T'$.

Suppose that for some $i \in \{0, \ldots, l-1\}$, there is a homomorphism h from T_i to T' with $h(T_i) = T'$. Let χ be an egd in Σ, and let $\bar{u}$ be a tuple such that $T_i \vdash_{\chi,\bar{u}} T_{i+1}$. Say, χ has the form

$$\forall x_1 \cdots \forall x_k \Big(\varphi(x_1, \ldots, x_k) \to x_i = x_j \Big),$$

and $\bar{u} = (u_1, \ldots, u_k)$. Then $T_i \models \varphi(\bar{u})$ and $u_i \neq u_j$. Say, $T_{i+1} = g(T_i)$, where for all $v \in \mathrm{dom}(T_i)$,

$$g(v) \;=\; \begin{cases} u_i, & \text{if } v = u_j \\ v, & \text{otherwise.} \end{cases} \tag{3.2}$$

Since $T_i \models \varphi(\bar{u})$, $h(T_i) = T'$, and φ is preserved under homomorphisms, we have $T' \models \varphi(h(\bar{u}))$. This implies that $h(u_i) = h(u_j)$, since T' is a CWA-solution for S under M, and hence $T' \models \Sigma$. But then,

$$h(g(u_j)) \overset{(3.2)}{=} h(u_i) \;=\; h(u_j)$$

and

$$h(g(v)) \overset{(3.2)}{=} h(v) \quad \text{for all } v \in \mathrm{dom}(T_i) \setminus \{u_j\}.$$

Consequently, $h(T_{i+1}) = h(g(T_i)) = h(T_i) = T'$, which completes the induction step. □

The unique maximal CWA-solution guaranteed by Proposition 3.28 will be called *canonical universal solution*:

3.29 Definition (canonical universal solution)
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of st-tgds and egds, or all tgds in Σ are full, and let S be a source instance for M.

Then $\mathrm{CanSol}(M, S)$ is the unique maximal CWA-solution for S under M if S has a CWA-solution under M, and undefined otherwise.

Thus, if $M = (\sigma_s, \sigma_t, \Sigma)$ is a schema mapping, where Σ consists of st-tgds and egds, or all tgds in Σ are full, then the set of all CWA-solutions for S under M has two unique extreme points (see Figure 3.2): $\mathrm{Core}(M, S)$, which is the unique minimal CWA-solution for S under M in the sense that every CWA-solution for S under M contains a subinstance that is isomorphic to $\mathrm{Core}(M, S)$, and $\mathrm{CanSol}(M, S)$, which is the unique maximal CWA-solution for S under M (i.e., for every CWA-solution T for S under M there is a homomorphism h from $\mathrm{CanSol}(M, S)$ to T with $h(\mathrm{CanSol}(M, S)) = T$).

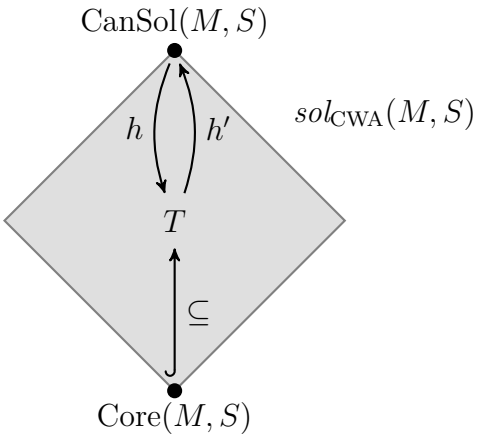


Figure 3.2: Structure of the set of all CWA-solutions for S under M in the case that M is a schema mappings defined by st-tgds and egds, or full tgds and egds.

3.30 Remark (Libkin [2006])
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of st-tgds, and let S be a source instance for M. While every CWA-solution for S under M contains a subinstance isomorphic to the minimal CWA-solution $\mathrm{Core}(M, S)$, there can be CWA-solutions T for S under M such that the maximal CWA-solution $\mathrm{CanSol}(M, S)$ contains no subinstance isomorphic to T.

For example, let $M = (\{R\}, \{R'\}, \Sigma)$, where Σ consists of the single st-tgd

$$\forall x \forall y \Big(R(x, y) \rightarrow \exists z \exists z' R'(x, z, z') \Big).$$

Furthermore, let S be the source instance for M with $R^S = \{(a, b), (a, c)\}$. Then,

$$(R')^{\mathrm{CanSol}(M,S)} = \{(a, \bot_1, \bot_2), (a, \bot_3, \bot_4)\}.$$

However, the target instance T for M with

$$(R')^T = \{(a, \bot_1, \bot), (a, \bot_3, \bot)\}$$

is a CWA-solution for S under M, and $\mathrm{CanSol}(M, S)$ contains no subinstance that is isomorphic to T. □

In general, $\mathrm{Core}(M, S)$ is still the unique minimal CWA-solution for S under M. However, there may be no maximal CWA-solution. The following example presents a very simple schema mapping M, a source instance S for M, and two CWA-solutions T_1, T_2 under M such that $\{T_1, T_2\}$ is a *maximal CWA-solution set* for S under M. Here, a set $\mathcal{T} \subseteq sol_{\mathrm{CWA}}(M, S)$ is called *maximal CWA-solution set for S under M* if and only if for every CWA-solution T' for S under M, there is some $T \in \mathcal{T}$ and a homomorphism h from T to T' with $h(T) = T'$, and there is no set $\mathcal{T}' \subsetneq \mathcal{T}$ with this property. The example also shows that for every positive integer n, there is a source instance for M that has a maximal CWA-solution set under M of size 2^n.

3.31 Example
We consider a slight extension of the schema mapping M from Remark 3.30. Let $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$ be such that $\sigma_{\mathrm{s}} = \{R\}$, $\sigma_{\mathrm{t}} = \{R', R''\}$, and Σ consists of the following tgds:

$$\chi_1 \;:=\; \forall x \forall y \Big(R(x, y) \rightarrow \exists z \exists z'\, R'(x, z, z') \Big),$$
$$\chi_2 \;:=\; \forall x \forall x_1 \forall x_2 \forall y \Big(R'(x, x_1, y) \wedge R'(x, x_2, y) \rightarrow R''(x, x_1, x_2) \Big).$$

For every positive integer n, let S_n be the source instance for M with $R^{S_n} = \{(i, j) \mid 1 \leq i \leq n,\, 1 \leq j \leq 2\}$.

Let us first consider the case $n = 1$. It is not hard to see that

$$\begin{aligned} T_1 \;&:=\; \{R'(1, \bot_1, \bot_2), R'(1, \bot_3, \bot_4), R''(1, \bot_1, \bot_1), R''(1, \bot_3, \bot_3)\} \\ T_2 \;&:=\; \{R'(1, \bot_1, \bot_2), R'(1, \bot_3, \bot_2), R''(1, \bot_1, \bot_1), R''(1, \bot_3, \bot_3), \\ &\qquad R''(1, \bot_1, \bot_3), R''(1, \bot_3, \bot_1)\} \end{aligned}$$

are CWA-solutions for S_1 under M. Note that for every CWA-solution T for S_1 under M there is some $i \in \{1, 2\}$ and a homomorphism h from T_i to T with $h(T_i) = T$. Indeed, let T be a CWA-solution for S_1 under M. Then there are (not necessarily distinct) nulls $\bot'_1, \bot'_2, \bot'_3, \bot'_4$ such that

$$(R')^T \;=\; \{(1, \bot'_1, \bot'_2), (1, \bot'_3, \bot'_4)\}.$$

If $\bot'_2 \neq \bot'_4$, we have

$$(R'')^T \;=\; \{(1, \bot'_1, \bot'_1), (1, \bot'_3, \bot'_3)\},$$

so that $T = h_1(T_1)$, where $h_1(1) = 1$, and $h_1(\bot_i) = \bot'_i$ for every $i \in \{1, 2, 3, 4\}$. Furthermore, if $\bot'_2 = \bot'_4$, we have

$$(R'')^T \;=\; \{(1, \bot'_1, \bot'_1), (1, \bot'_3, \bot'_3), (1, \bot'_1, \bot'_3), (1, \bot'_3, \bot'_1)\},$$

so that $T = h_2(T_2)$, where $h_2(a) = a$, and $h_2(\bot_i) = \bot'_i$ for every $i \in \{1, 2, 3, 4\}$. Furthermore, it is easy to see that there is no homomorphism h from T_1 to T_2 with $h(T_1) = T_2$, and vice versa. Thus, $\{T_1, T_2\}$ is a maximal CWA-solution set for S_1 under M.

Finally, let $n \geq 1$. Then for every set $\mathcal{I} \subseteq \{1, \ldots, n\}$, the target instance

$$T_{\mathcal{I}} := \bigcup_{i \in \mathcal{I}} \{R'(i, \bot_1^i, \bot_2^i), R'(i, \bot_3^i, \bot_4^i), R''(i, \bot_1^i, \bot_1^i), R''(i, \bot_3^i, \bot_3^i)\} \cup \bigcup_{i \in \{1,\ldots,n\} \setminus \mathcal{I}} \Big(\{R'(i, \bot_1^i, \bot_2^i), R'(i, \bot_3^i, \bot_2^i)\} \cup \{R''(i, \bot_k^i, \bot_l^i) \mid k, l \in \{1, 3\}\}\Big)$$

is a CWA-solution for S_n under M such that for every CWA-solution T for S_n under M there is some $\mathcal{I} \subseteq \{1, \ldots, n\}$ and a homomorphism h from $T_{\mathcal{I}}$ to T with $h(T_{\mathcal{I}}) = T$. Furthermore, for distinct $\mathcal{I}, \mathcal{I}' \subseteq \{1, \ldots, n\}$, there is no homomorphism h from $T_{\mathcal{I}}$ to $T_{\mathcal{I}'}$ with $h(T_{\mathcal{I}}) = T_{\mathcal{I}'}$. Thus, $\{T_{\mathcal{I}} \mid \mathcal{I} \subseteq \{1, \ldots, n\}\}$ is a maximal CWA-solution set for S_n under M of size 2^n. □

Thus, in general, the set of all CWA-solutions is as shown in Figure 3.3.

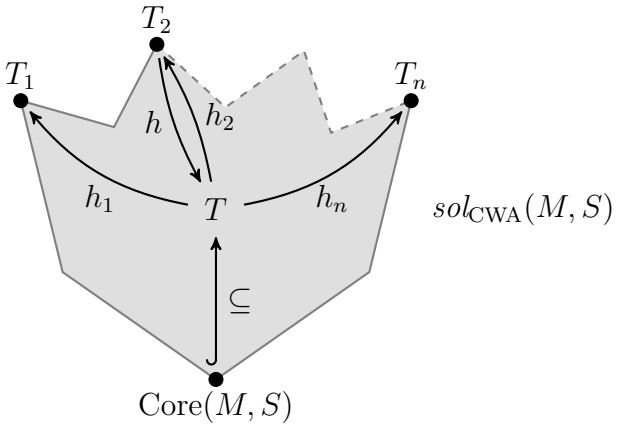


Figure 3.3: Structure of the set of all CWA-solutions for S under M in the case that M is a schema mappings defined by tgds and egds.

3.3 Semantics for Query Answering Using CWA-Solutions

Libkin [2006] defined various query answering semantics that are based on CWA-solutions. In this section, we review these semantics, adapted to schema mappings defined by tgds and egds.

Given a schema mapping $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$ defined by tgds and egds, a source instance S for M, and a query q over σ_{t}, the basic idea for these semantics is

- to evaluate q on each individual CWA-solution for S under M, and
- to combine the answers to a single answer.

Note that CWA-solutions are in general incomplete instances (cf., Section 1.3.1). For evaluating q on an individual CWA-solution, Libkin therefore uses techniques developed for answering queries on incomplete instances. As mentioned in Section 1.3.1, there are several ways of evaluating a query q on an incomplete instance $\mathcal{I}$. One of them is to take the *certain answers* to q on $\mathcal{I}$, introduced in Section 1.3.1, which can be seen as a "lower approximation" to the unknown query result (i.e., the result of q on the unknown instance represented by $\mathcal{I}$). Another one considered by Libkin is to take the *maybe answers* to q on $\mathcal{I}$ [van der Meyden, 1998], denoted by $\mathit{maybe}(q, \mathcal{I})$, which are defined as the set of all tuples that occur in $q(I)$ for some instance $I \in \mathcal{I}$. That is, the maybe answers can be seen as an "upper approximation" to the unknown query result.

Let $M = (\sigma_\mathrm{s}, \sigma_\mathrm{t}, \Sigma)$ be a schema mapping, where Σ is a set of tgds and egds, and let S be a source instance for M. When we answer a query q on a target instance T for M, we also have to take into account the t-tgds and egds of M (see also van der Meyden [1998] for answering queries on incomplete instances with respect to a set of constraints). That is, instead of taking the certain answers or the maybe answers on $\mathit{poss}(T)$, we take the certain answers or the maybe answers on

$$\mathit{poss}_M(T) \;:=\; \{\hat{T} \in \mathit{poss}(T) \mid \hat{T} \text{ satisfies all t-tgds and egds in } \Sigma\}.$$

In particular, this makes sense if T is a solution for some source instance under M. Then T represents an unknown solution with constants. Given a query q, we let

$$\Box_M q(T) \;:=\; \mathit{cert}(q, \mathit{poss}_M(T))$$

and

$$\Diamond_M q(T) \;:=\; \mathit{maybe}(q, \mathit{poss}_M(T)).$$

For combining answers to the individual CWA-solutions to a single answer we can either look for tuples which are answers on all CWA-solutions, or for tuples which are answers on some CWA-solution. By combining the different possibilities of evaluating queries on individual CWA-solutions, and combining answers to a single answer, one obtains the following four semantics:

3.32 Definition
Let $M = (\sigma_\mathrm{s}, \sigma_\mathrm{t}, \Sigma)$ be a schema mapping defined by tgds and egds, let S be a source instance for M, and let q be a query over σ_t. We define the following semantics for answering q with respect to S and M:

- *The certain CWA-answers semantics.* The certain CWA-answers to q on M under S, denoted by $\mathit{cert}_{\Box}(q, M, S)$, consist of all $\mathrm{ar}(q)$-tuples that occur in $\Box_M q(T)$ for every CWA-solution T for S under M. That is,

$$\mathit{cert}_{\Box}(q, M, S) \;=\; \bigcap_{T \in \mathit{sol}_{\mathrm{CWA}}(M,S)} \; \bigcap_{\hat{T} \in \mathit{poss}_M(T)} q(\hat{T}).$$

- *The potential certain CWA-answers semantics.* The potential certain CWA-answers to q on M and S, denoted by $\mathit{cert}_{\Diamond}(q, M, S)$, consist of all $\mathrm{ar}(q)$-tuples that occur in $\Box_M q(T)$ for some CWA-solution T for S under M. That is,

$$\mathit{cert}_{\Diamond}(q, M, S) \;=\; \bigcup_{T \in \mathit{sol}_{\mathrm{CWA}}(M,S)} \; \bigcap_{\hat{T} \in \mathit{poss}_M(T)} q(\hat{T}).$$

- *The persistent maybe CWA-answers semantics.* The persistent maybe CWA-answers to q on M and S, denoted by $\mathit{maybe}_{\Box}(q, M, S)$, consist of all $\mathrm{ar}(q)$-tuples that occur in $\Diamond_M q(T)$ for every CWA-solution T for S under M. That is,

$$\mathit{maybe}_{\Box}(q, M, S) \;=\; \bigcap_{T \in \mathit{sol}_{\mathrm{CWA}}(M,S)} \; \bigcup_{\hat{T} \in \mathit{poss}_M(T)} q(\hat{T}).$$

- *The maybe CWA-answers semantics.* The maybe CWA-answers to q on M and S, denoted by $\mathit{maybe}_{\Diamond}(q, M, S)$, consist of all $\mathrm{ar}(q)$-tuples that occur in $\Diamond_M q(T)$ for some CWA-solution T for S under M. That is,

$$\mathit{maybe}_{\Diamond}(q, M, S) \;=\; \bigcup_{T \in \mathit{sol}_{\mathrm{CWA}}(M,S)} \; \bigcup_{\hat{T} \in \mathit{poss}_M(T)} q(\hat{T}).$$

Let us now come back to the examples in Section 1.3.2. Each of the above-mentioned semantics leads to the expected query result in Example 1.25 under the CWA-based interpretation:

3.33 Example (Copying schema mappings)
Consider again the schema mapping M from Example 1.25. Let S be a source instance for M. Then the target instance T_S for M with $(R')^{T_S} = R^S$ is the unique CWA-solution for S under M, and therefore,

$$\mathit{cert}_{\Box}(q, M, S) \;=\; \mathit{cert}_{\Diamond}(q, M, S) \;=\; \mathit{maybe}_{\Box}(q, M, S) \;=\; \mathit{maybe}_{\Diamond}(q, M, S) \;=\; q(T_S),$$

as intuitively expected. □

More generally, if M is a schema mapping defined by full tgds and egds, and S is a source instance for M, then there is at most one CWA-solution T_S for S under M. This CWA-solution, if it exists, intuitively corresponds to the expected result of translating S to the target, so that the answer to a query q on M and S should be expected to be $q(T_S)$. Indeed, if T_S exists, we have

$$\mathit{cert}_\Box(q, M, S) \;=\; \mathit{cert}_\Diamond(q, M, S) \;=\; \mathit{maybe}_\Box(q, M, S) \;=\; \mathit{maybe}_\Diamond(q, M, S) \;=\; q(T_S).$$

Note also that $\mathit{cert}_\Box$, $\mathit{cert}_\Diamond$, $\mathit{maybe}_\Box$ and $\mathit{maybe}_\Diamond$ do not exhibit the behavior of the certain answers semantics described in Example 1.26. Furthermore, the certain CWA-answers semantics yields the expected answer to the query in Example 1.27:

3.34 Example
Recall the setting from Example 1.27. Then,

$$\mathit{cert}_\Box(q_3, M, S^*) \;=\; \mathit{cert}_\Diamond(q_3, M, S^*) \;=\; \{\text{0-201-53082-1}\},$$

as intuitively expected. To see this, we show that

$$\Box_M q_3(T) \;=\; \{\text{0-201-53082-1}\} \quad \text{for each } T \in \mathit{sol}_{\mathrm{CWA}}(M, S^*). \tag{3.3}$$

Indeed, if T' is the CWA-solution for S^* under M from Example 3.22, then $\Box_M q_3(T') = \{\text{0-201-53082-1}\}$, since for each valuation f of T', where $f(T')$ satisfies the egd χ_3 from Example 1.15, we have $f(\bot_4) \neq f(\bot_7)$. This implies (3.3) as follows. Let T be an arbitrary CWA-solution for S^* under M. As pointed out in Example 3.22, there is a homomorphism h from T' to T with $h(T') = T$. Thus, for each valuation f of T, where $f(T)$ satisfies the egd χ_3 from Example 1.15, the mapping $f \circ h$ is a valuation of T' such that $(f \circ h)(T') = f(T)$ satisfies the egd χ_3. Since $\Box_M q_3(T') = \{\text{0-201-53082-1}\}$, we therefore have $\text{0-201-53082-1} \in q_3(f(T))$. Consequently, (3.3) holds.

It is now easy to see that $\mathit{maybe}_\Diamond(q_3, M, S^*) = \{\text{0-201-53082-1}\}$, and furthermore, $\mathit{maybe}_\Box(q_3, M, S^*) = \{\text{0-201-53082-1}\}$. □

Let M be a schema mapping defined by tgds and egds, let S be a source instance for M, and let q be a query over M's target schema. Theorem 3.36 below characterizes $\mathit{cert}_\Diamond(q, M, S)$ as the certain answers to q on $\mathit{poss}_M(\mathrm{Core}(M, S))$, and $\mathit{maybe}_\Box(q, M, S)$ as the maybe answers to q on $\mathit{poss}_M(\mathrm{Core}(M, S))$. Similar characterizations, with $\mathrm{Core}(M, S)$ replaced by the canonical universal solution for S under M, hold for $\mathit{cert}_\Box(q, M, S)$ and $\mathit{maybe}_\Diamond(q, M, S)$ with respect to restricted schema mappings. Hence, to evaluate a query under one of these semantics, it often suffices to construct $T = \mathrm{Core}(M, S)$ or $T = \mathrm{CanSol}(M, S)$, and to compute the certain answers or the maybe answers on $\mathit{poss}_M(T)$. For stating Theorem 3.36, we need the following proposition:

3.35 Proposition.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of tgds and egds, and let S be a source instance for M. Then for every CWA-solution T for S under M, we have

$$poss_M(\mathrm{Core}(M,S)) \subseteq poss_M(T).$$

Moreover, if Σ consists of st-tgds and egds, or all tgds in Σ are full, then for every CWA-solution T for S under M, we have

$$poss_M(T) \subseteq poss_M(\mathrm{CanSol}(M,S)).$$

Proof. Let T be a CWA-solution for S under M.

Step 1: $poss_M(\mathrm{Core}(M,S)) \subseteq poss_M(T)$.
Let $\hat{T} \in poss_M(\mathrm{Core}(M,S))$. Then there is a valuation f of $\mathrm{Core}(M,S)$ with $f(\mathrm{Core}(M,S)) = \hat{T}$. On the other hand, T is a universal solution for S under M by Theorem 3.21. Thus, by Theorem 2.6(3), there is a homomorphism h from T to $\mathrm{Core}(M,S)$ with $h(T) = \mathrm{Core}(M,S)$. It follows that the composition $f' := f \circ h$ of h and f is a valuation of T with $f'(T) = f(\mathrm{Core}(M,S)) = \hat{T}$. Hence, $\hat{T} \in poss_M(T)$.

Step 2: $poss_M(T) \subseteq poss_M(\mathrm{CanSol}(M,S))$ if Σ consists of st-tgds and egds, or all tgds in Σ are full.
This can be proved in an analogous way as in step 1. Instead of Theorem 2.6(3), use Proposition 3.28 to obtain a homomorphism h from $\mathrm{CanSol}(M,S)$ to T with $h(\mathrm{CanSol}(M,S)) = T$. □

The following theorem generalizes the corresponding result from Libkin [2006] for schema mappings defined by st-tgds:

3.36 Theorem (Characterization of CWA-answers semantics).
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of tgds and egds, let S be a source instance for M that has a CWA-solution under M, and let q be a query over σ_t. Then we have:

1. *$cert_{\Diamond}(q,M,S) = \Box_M q(\mathrm{Core}(M,S))$, and*
2. *$maybe_{\Box}(q,M,S) = \Diamond_M q(\mathrm{Core}(M,S))$.*

Moreover, suppose that T^* is a CWA-solution for S under M such that for every CWA-solution T for S under M we have $poss_M(T) \subseteq poss_M(T^*)$. Then:

3. *$cert_{\Box}(q,M,S) = \Box_M q(T^*)$, and*

4. $maybe_{\Diamond}(q, M, S) = \Diamond_M q(T^*)$.

Proof. The proof is analogous to the corresponding proof by Libkin [2006]. We first prove 1 and 2. By Proposition 3.35, each CWA-solution T for S under M satisfies $poss_M(\mathrm{Core}(M,S)) \subseteq poss_M(T)$, and therefore

$$\Box_{\Sigma_t} q(T) = \bigcap_{\hat{T} \in poss_M(T)} q(\hat{T}) \subseteq \bigcap_{\hat{T} \in poss_M(\mathrm{Core}(M,S))} q(\hat{T}) = \Box_M q(\mathrm{Core}(M,S)), \tag{3.4}$$

$$\Diamond_M q(T) = \bigcup_{\hat{T} \in poss_M(T)} q(\hat{T}) \supseteq \bigcup_{\hat{T} \in poss_M(\mathrm{Core}(M,S))} q(\hat{T}) = \Diamond_M q(\mathrm{Core}(M,S)). \tag{3.5}$$

Consequently,

$$cert_{\Diamond}(q, M, S) = \bigcup_{T \in sol_{\mathrm{CWA}}(M,S)} \Box_M q(T) \overset{(3.4)}{=} \Box_M q(\mathrm{Core}(M,S)),$$

$$maybe_{\Box}(q, M, S) = \bigcap_{T \in sol_{\mathrm{CWA}}(M,S)} \Diamond_M q(T) \overset{(3.5)}{=} \Diamond_M q(\mathrm{Core}(M,S)).$$

For proving 3 and 4, let T^* be a CWA-solution for S under M such that for every CWA-solution T for S under M we have $poss_M(T) \subseteq poss_M(T^*)$. Then, in a similar way as above,

$$\Box_M q(T) = \bigcap_{\hat{T} \in poss_M(T)} q(\hat{T}) \supseteq \bigcap_{\hat{T} \in poss_M(T^*)} q(\hat{T}) = \Box_M q(T^*), \tag{3.6}$$

$$\Diamond_M q(T) = \bigcup_{\hat{T} \in poss_M(T)} q(\hat{T}) \subseteq \bigcup_{\hat{T} \in poss_M(T^*)} q(\hat{T}) = \Diamond_M q(T^*), \tag{3.7}$$

and consequently,

$$cert_{\Box}(q, M, S) = \bigcap_{T \in sol_{\mathrm{CWA}}(M,S)} \Box_M q(T) \overset{(3.6)}{=} \Box_M q(T^*),$$

$$maybe_{\Diamond}(q, M, S) = \bigcup_{T \in sol_{\mathrm{CWA}}(M,S)} \Diamond_M q(T) \overset{(3.7)}{=} \Diamond_M q(T^*). \qquad \square$$

Note that together with Proposition 3.35, Theorem 3.36 directly leads to:

3.37 Corollary.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping such that Σ consists of st-tgds and egds, or all tgds in Σ are full. Let S be a source instance for M that has a CWA-solution under M, and let q be a query over σ_t. Then:

1. $cert_{\Box}(q, M, S) = \Box_M q(\mathrm{CanSol}(M, S))$
2. $cert_{\Diamond}(q, M, S) = \Box_M q(\mathrm{Core}(M, S))$
3. $maybe_{\Box}(q, M, S) = \Diamond_M q(\mathrm{Core}(M, S))$
4. $maybe_{\Diamond}(q, M, S) = \Diamond_M q(\mathrm{CanSol}(M, S))$

Furthermore, one can use Theorem 3.36 to establish the following relationships between the new semantics. These relationships were pointed out by Libkin [2006] for the case of schema mappings defined by st-tgds.

3.38 Corollary.
For each schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$ defined by tgds and egds, each source instance S for M, and each query q over σ_t, we have:

$$cert_{\Box}(q, M, S) \subseteq cert_{\Diamond}(q, M, S) \subseteq maybe_{\Box}(q, M, S) \subseteq maybe_{\Diamond}(q, M, S).$$

Proof. The inclusions $cert_{\Box}(q, M, S) \subseteq cert_{\Diamond}(q, M, S)$ and $maybe_{\Box}(q, M, S) \subseteq maybe_{\Diamond}(q, M, S)$ follow directly from the definitions. To prove $cert_{\Diamond}(q, M, S) \subseteq maybe_{\Box}(q, M, S)$, observe that for all CWA-solutions T for S under M we have

$$\Box_M q(T) \subseteq \Diamond_M q(T).$$

Therefore, by Theorem 3.36, we have

$$\begin{aligned} cert_{\Diamond}(q, M, S) &= \Box_M q(\mathrm{Core}(M, S)) \\ &\subseteq \Diamond_M q(\mathrm{Core}(M, S)) = maybe_{\Box}(q, M, S). \end{aligned}$$

□

Notice also that for each schema mapping M, each source instance S for M, and each query q over M's target schema, $cert(q, M, S) \subseteq cert_{\Box}(q, M, S)$.

3.4 Complexity of Query Answering Using CWA-Solutions

We now turn to the problem of answering queries under the four semantics from Definition 3.32. We focus on the *data complexity*, which assumes that the schema mapping and the query is fixed (i.e., they do not belong to the input). Since in data exchange, queries must usually be answered based on some "materialized" solution, given a schema mapping $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$ defined by tgds and egds, and a query language L, we seek for algorithms $\mathbb{A}_1, \mathbb{A}_2$ with the following properties:

- $\mathbb{A}_1$ takes a source instance S for M as input and computes a solution T for S under M, and

- $\mathbb{A}_2$ takes a solution T computed by $\mathbb{A}_1$ and a query $q \in L$ over σ_t as input and computes the answers to q on M and S under one of the semantics from Definition 3.32.

In particular, $\mathbb{A}_1$ takes care of the actual data exchange, while $\mathbb{A}_2$ answers queries based on some materialized solution, independent of M and S. At best, both $\mathbb{A}_1$, and $\mathbb{A}_2$ for fixed $q \in L$, run in polynomial time.

In Section 3.4.1, we will show that the certain CWA-answers semantics and the potential certain CWA-answers semantics coincide with the certain answers semantics on queries preserved under homomorphisms and source instances that have a CWA-solution. Hence, if L is the class of all homomorphism-preserved queries with polynomial time data complexity, and M is a weakly acyclic schema mapping, say, then polynomial time algorithms $\mathbb{A}_1$ and $\mathbb{A}_2$ as above exist: $\mathbb{A}_1$ just has to compute an arbitrary universal solution T for S under M, while $\mathbb{A}_2$ evaluates a query $q \in L$ on T and removes all tuples with nulls from the answer (see Chapter 2).

Section 3.4.2 then considers FO queries in general. In particular, we will see that beyond the class of queries preserved under homomorphisms, polynomial time algorithms $\mathbb{A}_1$ and $\mathbb{A}_2$ as above may not exist. To show this, we consider, for given semantics $ans \in \{cert_\Box, cert_\Diamond, maybe_\Box, maybe_\Diamond\}$, schema mappings $M = (\sigma_s, \sigma_t, \Sigma)$ defined by tgds and egds, and queries q over σ_t, the problem

$\textsc{Eval}_{ans}(M, q)$	
Input:	a source instance S for M, and a tuple $\bar{t} \in Dom^{\mathrm{ar}(q)}$
Question:	Is $\bar{t} \in ans(q, M, S)$?

The complexity of this problem is a lower bound on the joint complexity of algorithms $\mathbb{A}_1$ and $\mathbb{A}_2$ as above: if, for example, $\textsc{Eval}_{ans}(M, q)$ is co-NP-complete, then finding T is intractable, or computing $ans(q, M, S)$ from T is intractable.

3.4.1 Queries Preserved Under Homomorphisms

In the following, we show that the certain CWA-answers semantics and the potential certain CWA-answers semantics coincide with the certain answers semantics on queries preserved under homomorphisms and source instances that have a CWA-solution. Recall from Section 2.1 that, given a schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$, a source instance S for M, a query q over σ_t, and an arbitrary universal solution T for S under M, the certain answers to q on M and S are characterized as

$$q(T)_\downarrow \;:=\; \{\bar{t} \in q(T) \mid \bar{t} \text{ contains only constants}\}.$$

We now have:

3.39 Lemma.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping defined by tgds and egds, let S be a source instance for M, and let q be a query over σ_t that is preserved under homomorphisms. Then, for every CWA-solution T for S under M, we have

$$cert_{\Box}(q, M, S) \;=\; cert_{\Diamond}(q, M, S) \;=\; q(T)_{\downarrow}.$$

Proof. Let Σ' be the set of all t-tgds and egds in Σ. We first show the following intermediate claim:

> If T and T' are homomorphically equivalent target instances for M and $T' \models \Sigma'$, then $\Box_M q(T') = q(T)_{\downarrow}$. $\quad(\star)$

Let T and T' be target instances for M such that $T' \models \Sigma'$. Furthermore, let h be a homomorphism from T' to T, and let h' be a homomorphism from T to T'. We first show that $\Box_M q(T') \subseteq q(T)_{\downarrow}$.

Let $\bar{t} \in \Box_M q(T')$. Then for all $\hat{T} \in poss_M(T')$ we have $\bar{t} \in q(\hat{T})$. This implies that $\bar{t} \in q(T')$ (here we need that $T' \models \Sigma'$), and that $\bar{t}$ consists of constants. Since h is a homomorphism from T' to T, and q is preserved under homomorphisms, this leads to $h(\bar{t}) \in q(T)$. Since $\bar{t}$ consists of constants, and h is the identity on constants, we conclude that $\bar{t} \in q(T)_{\downarrow}$.

The proof for $q(T)_{\downarrow} \subseteq \Box_M q(T')$ is almost identical: If $\bar{t} \in q(T)_{\downarrow}$, then $\bar{t} \in q(T)$, and $\bar{t}$ contains only constants. The remaining part of the proof is then the same, except that T and T' must be interchanged, and h must be replaced with h'.

Now, by Theorem 3.21, every two CWA-solution for S under M are homomorphically equivalent, and satisfy Σ'. Therefore, for every CWA-solution T for S under M we have

$$cert_{\Box}(q, M, S) \;=\; \bigcap_{T' \in sol_{\mathrm{CWA}}(M,S)} \Box_M q(T') \;\overset{(\star)}{=}\; \bigcap_{T' \in sol_{\mathrm{CWA}}(M,S)} q(T)_{\downarrow} \;=\; q(T)_{\downarrow}$$

and

$$cert_{\Diamond}(q, M, S) \;\overset{\text{Thm. 3.36}}{=}\; \Box_M q(\mathrm{Core}(M, S)) \;\overset{(\star)}{=}\; q(T)_{\downarrow}. \qquad \Box$$

3.40 Corollary.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping defined by tgds and egds, let S a source instance for M that has a CWA-solution under M, and let q be a query over σ_t that is preserved under homomorphisms. Then,

$$cert_{\Box}(q, M, S) \;=\; cert_{\Diamond}(q, M, S) \;=\; cert(q, M, S).$$

Proof. Let T be a CWA-solution for S under M. By Lemma 3.39, we have $\mathit{cert}_{\Box}(q, M, S) = \mathit{cert}_{\Diamond}(q, M, S) = q(T)_{\downarrow}$. Furthermore, by Theorem 3.21 and Theorem 2.10, we have $q(T)_{\downarrow} = \mathit{cert}(q, M, S)$. □

Note that Corollary 3.40 and Theorem 2.26(2) imply that the certain CWA-answers semantics and the potential certain CWA-answers semantics coincide with the certain answers semantics on weakly acyclic schema mappings and queries preserved under homomorphisms. Thus, given a weakly acyclic schema mapping $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$, and a query q over σ_{t} that is preserved under homomorphisms and has polynomial time data complexity, there are polynomial time algorithms $\mathbb{A}_1$ and $\mathbb{A}_2$ as described at the beginning of the present Section 3.4: $\mathbb{A}_1$ just has to compute an arbitrary universal solution T for S under M, while $\mathbb{A}_2$ evaluates a query $q \in L$ on T and removes all tuples with nulls from the answer (cf., Chapter 2).

3.4.2 First-Order Queries

Next we consider the problem of answering FO queries under one of the semantics from Definition 3.32.

As Proposition 3.41 below shows, the $\textsc{Eval}_{\mathit{ans}}$-problem may be undecidable for $\mathit{ans} \in \{\mathit{cert}_{\Box}, \mathit{maybe}_{\Diamond}\}$ and FO queries, even with respect to a weakly acyclic schema mapping.

3.41 Proposition.
There is a weakly acyclic schema mapping M and a FO query q over M's target schema such that $\textsc{Eval}_{\mathit{cert}_{\Box}}(M, \neg q)$ and $\textsc{Eval}_{\mathit{maybe}_{\Diamond}}(M, q)$ are undecidable.

Proof. We will construct M and q in such a way that the (undecidable) embedding problem for finite semigroups (see Example 1.14) can be reduced to $\textsc{Eval}_{\mathit{cert}_{\Box}}(M, \neg q)$ and $\textsc{Eval}_{\mathit{maybe}_{\Diamond}}(M, q)$. The source schema σ_{s} and the target schema σ_{t} of M consist of ternary relation symbols R and $\tilde{R}$, respectively, so that inputs $p\colon X^2 \to X$ for the embedding problem for finite semigroups, and solutions $f\colon Y^2 \to Y$ to p with respect to this problem will be encoded in exactly the same way by instances S_p and T_f as in Example 1.14.

Let $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$, where Σ consists of

$$\forall x \forall y \forall z \Big(R(x, y, z) \to \tilde{R}(x, y, z) \Big),$$
$$\forall x \forall y \forall z \Big(\tilde{R}(x, y, z) \to \exists x' \exists y' \exists z'\, \tilde{R}(x', y', z') \Big).$$

Furthermore, let q be the FO query

$$\forall x\forall y\forall z_1\forall z_2\Big(\tilde{R}(x,y,z_1)\wedge\tilde{R}(x,y,z_2)\rightarrow z_1=z_2\Big)\wedge$$
$$\forall x\forall y\forall z\forall u\forall v\forall w\Big(\tilde{R}(x,y,u)\wedge\tilde{R}(y,z,v)\wedge\tilde{R}(u,z,w)\rightarrow\tilde{R}(x,v,w)\Big)\wedge$$
$$\forall x_1\forall x_2\forall x_3\forall y_1\forall y_2\forall y_3\Big(\tilde{R}(x_1,x_2,x_3)\wedge\tilde{R}(y_1,y_2,y_3)\rightarrow\bigwedge_{1\leq i,j\leq 3}\exists z\tilde{R}(x_i,y_j,z)\Big).$$

Note that q is equivalent to a conjunction of the egd (2.12), and the tgds (2.13) and (2.14) used by Kolaitis et al. [2006] in the proof of Theorem 2.37 (see the end of Section 2.3). Note also that q is satisfied in a target instance T for M if and only if $\tilde{R}^T$ encodes the graph of a total associative function $f\colon Y^2\rightarrow Y$ for some finite set Y.

We show that $\textsc{Eval}_{\mathit{maybe}_\Diamond}(M,q)$ is undecidable. Since $\mathit{maybe}_\Diamond$ and $\mathit{cert}_\Box$ are dual to each other (i.e., $\mathit{maybe}_\Diamond(q,M,S)\neq\emptyset$ if and only if $\mathit{cert}_\Box(\neg q,M,S)=\emptyset$), this also implies that $\textsc{Eval}_{\mathit{cert}_\Box}(M,\neg q)$ is undecidable.

To show that $\textsc{Eval}_{\mathit{maybe}_\Diamond}(M,q)$ is undecidable, we give a reduction from the embedding problem for finite semigroups (see Example 1.14). Given an associative partial function $p\colon X^2\rightarrow X$, we map p to the source instance S_p described in Example 1.14, and the empty tuple. It remains to show that $\mathit{maybe}_\Diamond(q,M,S_p)\neq\emptyset$ precisely if p is a "yes"-instance for the embedding problem for finite semigroups.

If $\mathit{maybe}_\Diamond(q,M,S_p)\neq\emptyset$, then there is a CWA-solution T for S_p under M and $\hat{T}\in\mathit{poss}_M(T)$ with $\hat{T}\models q$. So, $\tilde{R}^{\hat{T}}$ represents the graph of a finite total associative function f that extends p. Consequently, p is a "yes"-instance for the embedding problem for finite semigroups.

On the other hand, suppose that p is a "yes"-instance for the embedding problem for finite semigroups. That is, there is a total associative function $f\colon Y^2\rightarrow Y$ for some finite set $Y\supseteq X$ such that f extends p. Consider an enumeration $(a_1^1,a_2^1,a_3^1),\ldots,(a_1^k,a_2^k,a_3^k)$ of all tuples $(a_1,a_2,a_3)\in Y^3$ with $f(a_1,a_2)=a_3$ for which $p(a_1,a_2)$ is undefined. Furthermore, pick pairwise distinct nulls $\bot_1^1,\bot_2^1,\bot_3^1,\bot_1^2,\ldots,\bot_3^k$. Then the instance T with

$$\tilde{R}^T \;=\; R^{S_p}\cup\{(\bot_1^i,\bot_2^i,\bot_3^i)\mid 1\leq i\leq k\}$$

is a CWA-solution for S_p under M: $(\bot_1^1,\bot_2^1,\bot_3^1)$ can be generated from an arbitrary tuple in R^{S_p}, and $(\bot_1^{i+1},\bot_2^{i+1},\bot_3^{i+1})$ can be generated from $(\bot_1^i,\bot_2^i,\bot_3^i)$. Finally, consider the instance $\hat{T}=v(T)$, where v maps each $\bot_j^i$ to a_j^i. Then $\hat{T}\models q$, which implies that $\mathit{maybe}_\Diamond(q,M,S_p)\neq\emptyset$. □

In the following, we restrict attention to particular weakly acyclic schema mappings, called *richly acyclic schema mappings*. Note that Proposition 3.41

depends entirely on the fact that, given a schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$ defined by tgds and egds, a source instance S for M, a mapping $\zeta\colon \mathcal{J}^*_{\Sigma'} \to \mathit{Dom}$, where Σ' is the set of all tgds in Σ, and a tuple $j^* = (\chi, \bar{u}, \bar{v}, z) \in \mathcal{J}^*_{\Sigma'}$, the value of $\zeta(j^*)$ does not only depend on χ, $\bar{u}$ and z, but also on $\bar{v}$. This makes it possible to cascade the creation of nulls even though the schema mapping is weakly acyclic. *Richly acyclic schema mappings* prohibit this.

3.42 Definition (Richly acyclic schema mapping)
Let Σ be a set of tgds.

- The *extended dependency graph* of Σ is obtained from the dependency graph of Σ (see Definition 2.23) as follows: for every tgd $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \to \exists\bar{z}\,\psi(\bar{x},\bar{z}))$ in Σ, every variable y in $\bar{y}$, and every position (R,p) at which y appears in φ, add an existential edge from (R,p) to every position at which some variable from $\bar{z}$ appears in ψ.
- Σ is *richly acyclic* if and only if no cycle in the extended dependency graph of Σ contains an existential edge.
- A schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$ is called *richly acyclic* if and only if Σ is the union of a richly acyclic set of tgds, and a set of egds.

Note that every richly acyclic schema mapping is weakly acyclic, but not vice versa. Furthermore, by Theorem 3.44 below, the Eval_{ans}-problem is decidable for FO queries with respect to richly acyclic schema mappings. The crucial property for proving Theorem 3.44 is:

3.43 Lemma.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a richly acyclic schema mapping, and let $\zeta\colon \mathcal{J}^*_{\Sigma'} \to \mathit{Dom}$, where Σ' is the set of all tgds in Σ. Then there is a polynomial p_M such that, given a source instance S for M, each ζ-chase sequence on S and Σ' has length at most $p_M(\|S\|)$.

The lemma can be proved in the same way as Theorem 2.26(1). In fact, the proof given by Fagin et al. [2005a] for proving Theorem 2.26(1) works without modification.

3.44 Theorem.
For every schema mapping M defined by tgds and egds, and each FO query q over M's target schema, we have:

1. *If the schema mapping M is weakly acyclic, then $\text{Eval}_{cert_\Diamond}(M,q) \in$ co-NP and $\text{Eval}_{maybe_\Box}(M,q) \in$ NP.*

2. *If the schema mapping M is richly acyclic, then* $\text{EVAL}_{cert_\Box}(M, q) \in$ co-NP *and* $\text{EVAL}_{maybe_\Diamond}(M, q) \in$ NP.

Proof. Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping defined by tgds and egds, and let q be a FO query over σ_t.

Ad 1: Since $cert_\Diamond$ and $maybe_\Box$ are dual to each other (i.e., for every tuple $\bar{t}$ we have $\bar{t} \in cert_\Diamond(q, M, S)$ precisely if $\bar{t} \notin maybe_\Box(\neg q, M, S)$), it suffices to show that $\text{EVAL}_{maybe_\Box}(M, q) \in$ NP.

Let S be a source instance for M, and let $\bar{t} \in Dom^{\text{ar}(q)}$. By Theorem 3.36, we have

$$maybe_\Box(q, M, S) \;=\; \Diamond_M q(\text{Core}(M, S))$$

if CWA-solutions for S under M exist. Note that if no CWA-solution for S under M exist, then by Corollary 3.24, $\text{Core}(M, S)$ does not exist. In particular, the algorithm guaranteed by Theorem 2.32 will indicate that $\text{Core}(M, S)$ does not exist, so that we can output $maybe_\Box(q, M, S) = \emptyset$. So in the following we assume that CWA-solutions for S under M exist.

By Theorem 2.32, we can compute $T_0 := \text{Core}(M, S)$ in time polynomial in the size of S. Thus, it remains to show that we can nondeterministically check whether $\bar{t} \in \Diamond_M q(T_0)$ in time polynomial in the size of T_0.

Note that $\bar{t} \in \Diamond_M q(T_0)$ if and only if there is an instance $\hat{T} \in poss_M(T_0)$ such that $\bar{t} \in q(\hat{T})$. Furthermore, if there is an instance $\hat{T} \in poss_M(T)$ with $\bar{t} \in q(\hat{T})$, then there is such an instance $\hat{T}$ with $\text{dom}(\hat{T}) \subseteq C \cup f(\text{nulls}(T))$, where C is the set of all constants that occur in T, q and $\bar{t}$, and $f\colon \text{nulls}(T) \to Const \setminus C$ is an injective function. Thus, we can check $\bar{t} \in \Diamond_M q(T_0)$ by the following nondeterministic procedure:

1. "Guess" a valuation $v\colon \text{dom}(T) \to C \cup f(\text{nulls}(T))$ of T.
2. Check whether $\hat{T} := v(T_0)$ satisfies all t-tgds and egds of Σ. If not, reject the input.
3. If $\bar{t} \in q(\hat{T})$, accept the input. Otherwise reject it.

Clearly, this procedure runs in time polynomial in the size of T_0, which completes the proof of 1.

Ad 2: Since $cert_\Box$ and $maybe_\Diamond$ are dual to each other (i.e., for every tuple $\bar{t}$ we have $\bar{t} \in cert_\Box(q, M, S)$ precisely if $\bar{t} \notin maybe_\Diamond(\neg q, M, S)$), it suffices to show that $\text{EVAL}_{maybe_\Diamond}(M, q) \in$ NP.

Let S be a source instance for M, and let $\bar{t} \in Dom^{\text{ar}(q)}$. Then, $\bar{t} \in maybe_\Diamond(q, M, S)$ if and only if there are a CWA-solution T for S under M

with $\bar{t} \in \Diamond_M q(T)$. Therefore, the following nondeterministic algorithm decides whether $\bar{t}$ belongs to $\mathit{maybe}_\Diamond(q, M, S)$:

1. Compute $T_0 := \mathrm{Core}(M, S)$.
2. Generate a successful ζ-chase sequence C on S and Σ', where Σ' is the set of all tgds in Σ, "guessing" the relevant values for $\zeta\colon \mathcal{J}^*_{\Sigma'} \to \mathit{Dom}$ "along the way".[4] Let $S \cup T$ be the result of C.
3. If T does not satisfy the egds of Σ, reject the input.
4. Check whether there is a homomorphism from T to $\mathrm{Core}(M, S)$. If not, reject the input.
5. If $\bar{t} \in \Diamond_M q(T)$, then accept the input. Otherwise reject the input.

Note that step 2–4 guarantee that T is a CWA-solution for S under M: step 2 and 3 ensure that T is a CWA-presolution for S under M, and step 4 ensures that T is a universal solution for S under M. Thus, the algorithm indeed checks whether $\bar{t} \in \mathit{maybe}_\Diamond(q, M, S)$.

Furthermore, the algorithm runs in time polynomial in the size of S: By Theorem 2.32, we can accomplish step 1 in time polynomial in $\|S\|$. By Lemma 3.43, step 2 can be accomplished in time polynomial in $\|S\|$ as well. It is also easy to see that steps 3 and 4 can be accomplished in time polynomial in $\|S\|$. Finally, we have shown in part 1 of the proof that step 5 can be accomplished in time polynomial in $\|S\|$. □

On the other hand, there are richly acyclic schema mappings M such that for each $\mathit{ans} \in \{\mathit{cert}_\Box, \mathit{cert}_\Diamond, \mathit{maybe}_\Box, \mathit{maybe}_\Diamond\}$ there is a FO query q over M's target schema such that $\textsc{Eval}_{\mathit{ans}}(M, q)$ is co-NP-complete if $\mathit{ans} \in \{\mathit{cert}_\Box, \mathit{cert}_\Diamond\}$, and NP-complete if $\mathit{ans} \in \{\mathit{maybe}_\Box, \mathit{maybe}_\Diamond\}$. Indeed, it is not hard to see that Mądry's proof of Theorem 2.44 carries over to the semantics $\mathit{cert}_\Box$ and $\mathit{cert}_\Diamond$. Since $\mathit{maybe}_\Diamond$ and $\mathit{cert}_\Box$ are dual and $\mathit{maybe}_\Box$ and $\mathit{cert}_\Diamond$ are dual, we get the completeness results for $\mathit{maybe}_\Box$ and $\mathit{maybe}_\Diamond$. The following theorem improves Theorem 2.44 in the sense that q can be chosen to be a Boolean conjunctive query with *only one* inequality.

3.45 Theorem.
There is a richly acyclic schema mapping M and a conjunctive query q over M's target schema with one inequality such that:

[4] Note that we can restrict attention to mappings $\zeta\colon \mathcal{J}^*_{\Sigma'} \to \mathrm{dom}(S) \cup C \cup \mathit{Null}$, where C is the set of all constants that occur in Σ—all other choices do not lead to CWA-solutions for S under M.

1. $\textsc{Eval}_{cert_\Box}(M, q)$ *and* $\textsc{Eval}_{cert_\Diamond}(M, q)$ *are* co-NP*-complete.*

2. $\textsc{Eval}_{maybe_\Box}(M, \neg q)$ *and* $\textsc{Eval}_{maybe_\Diamond}(M, \neg q)$ *are* NP*-complete.*

Proof. We prove only 1, since 2 follows by duality. Let $M = (\sigma_\mathrm{s}, \sigma_\mathrm{t}, \Sigma)$, where $\sigma_\mathrm{s} = \{R, C, L\}$, $\sigma_\mathrm{t} = \{R', C', L'\}$, and Σ consists of

- $\forall i \forall j \forall p \Big(R(i,j,p) \to R'(i,j,p) \Big)$,
- $\forall i \Big(C(i) \to \exists t\, C'(i,t) \Big)$,
- $\forall j \forall p \Big(L(j,p) \to \exists t\, L'(j,p,t) \Big)$,
- $\forall i \Big(C'(i,1) \to \exists j \exists p (R'(i,j,p) \wedge L'(j,p,1)) \Big)$,
- $\forall j \forall p \forall p' \Big(L'(j,p,1) \wedge L'(j,p',1) \to p = p' \Big)$.

Finally, let

$$q \;:=\; \exists i \exists t \Big(C'(i,t) \wedge t \neq 1 \Big).$$

Note that M is richly acyclic. Hence, $\textsc{Eval}_{cert_\Box}(M, q)$ and $\textsc{Eval}_{cert_\Diamond}(M, q)$ are in co-NP by Theorem 3.44. To prove co-NP-hardness, we give a reduction from the complement of the NP-complete Sat, the satisfiability problem for propositional formulas in conjunctive normal form (see, e.g., Papadimitriou [1994]).

The reduction is carried out as follows. On input of a propositional formula

$$\varphi(x_1, x_2, \ldots, x_n) \;:=\; C_1 \wedge C_2 \wedge \cdots \wedge C_m$$

in conjunctive normal form, we construct the source instance

$$\begin{aligned} S_\varphi \;:=\; & \{R(i,j,1) \mid x_j \text{ occurs in } C_i\} \;\cup\; \{R(i,j,0) \mid \neg x_j \text{ occurs in } C_i\} \\ & \cup\, \{C(i) \mid 1 \leq i \leq m\} \;\cup\; \{L(j,b) \mid 1 \leq j \leq n \text{ and } b \in \{0,1\}\}. \end{aligned}$$

Note that there is exactly one CWA-solution for S_φ (up to isomorphism), denoted by T_φ. It consists of a copy of R, contains for each $i \in \{1, 2, \ldots, m\}$ an atom $C'(i, \bot_i)$, and for each $j \in \{1, \ldots, n\}$ and $b \in \{0, 1\}$ an atom $L'(j, b, \bot_{j,b})$, where the nulls $\bot_i, \bot_{j,b}$ introduced for each i, j and b are pairwise distinct. Therefore, $cert_\Box(q, M, S_\varphi) = cert_\Diamond(q, M, S_\varphi) = \Box_M q(T_\varphi)$. We claim that φ is satisfiable if and only if $\Box_M q(T_\varphi) = \emptyset$.

"Only if": Let $\alpha\colon \{x_1, x_2, \ldots, x_n\} \to \{0, 1\}$ be a satisfying truth assignment for φ. We extend α to negated variables by $\alpha(\neg x_i) = 1 - \alpha(x_i)$. Then, for every

$i \in \{1, \ldots, m\}$ there is a literal ℓ in C_i such that $\alpha(\ell) = 1$. Define a valuation v of T_φ such that for each $\bot \in \text{nulls}(T_\varphi)$,

$$v(\bot) = \begin{cases} 1 & \text{if } \bot = \bot_i \\ \alpha(x_j) & \text{if } \bot = \bot_{j,1} \\ \alpha(\neg x_j) & \text{if } \bot = \bot_{j,0}. \end{cases}$$

Then, $v(T_\varphi)$ belongs to $poss_M(T_\varphi)$, but $v(T_\varphi)$ does not satisfy q. Therefore, $\Box_M q(T_\varphi) = \emptyset$.

"If": Assume now that $\Box_M q(T_\varphi) = \emptyset$. Then $poss_M(T_\varphi)$ contains an instance $\hat{T}$ that does not satisfy q. Define a truth assignment $\alpha\colon \{x_1, \ldots, x_n\} \to \{0, 1\}$ such that for each $j \in \{1, \ldots, n\}$,

$$\alpha(x_j) = \begin{cases} 1 & \text{if } L'(j, 1, 1) \in \hat{T} \\ 0 & \text{otherwise,} \end{cases}$$

and extend it to negated variables as above. We claim that α satisfies φ; that is, for each $i \in \{1, \ldots, m\}$ there is a literal ℓ in C_i with $\alpha(\ell) = 1$.

Let $i \in \{1, \ldots, m\}$. Then $C(i, 1) \in \hat{T}$, because $\hat{T} \not\models q$. Since $\hat{T} \in poss_M(T_\varphi)$, there are $j \in \{1, \ldots, n\}$ and $p \in \{0, 1\}$ such that $R'(i, j, p)$ and $L'(j, p, 1)$ are in $\hat{T}$, and $L'(j, 1 - p, 1) \notin \hat{T}$. If $p = 0$, then we have $R'(i, j, 0) \in \hat{T}$ and $L'(j, 1, 1) \notin \hat{T}$. $R'(i, j, 0) \in \hat{T}$ indicates that the literal $\neg x_j$ occurs in C_i, and $L'(j, 1, 1) \notin \hat{T}$ indicates that $\alpha(\neg x_j) = 1 - \alpha(x_j) = 1$. So, if $p = 0$, then C_i is satisfied under α. It remains therefore to show that C_i is satisfied under α if $p = 1$. If $p = 1$, then $R'(i, j, 1)$ and $L'(j, 1, 1)$ are in $\hat{T}$. $R'(i, j, 1) \in \hat{T}$ indicates that the literal x_j occurs in C_i, and $L'(j, 1, 1) \in \hat{T}$ indicates that $\alpha(x_j) = 1$. Consequently, C_i is satisfied under α. □

Contrast this with Theorem 2.45, which tells us that for every weakly acyclic schema mapping M and for each union q of conjunctive queries with at most one inequality per disjunct, there is a polynomial time algorithm that computes the certain answers to q on M and S, given a source instance S for M as input. It is not hard to see that this algorithm can be used to compute $cert_\Box$ and $cert_\Diamond$ for schema mappings $M = (\sigma_\text{s}, \sigma_\text{t}, \Sigma)$, where Σ consists of st-tgds and egds, or all tgds in Σ are full:

3.46 Proposition.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of st-tgds and egds, or all tgds in Σ are full, and let q be a Boolean query that is the union of conjunctive queries with at most one inequality per disjunct. Then there is a polynomial time algorithm that, given a source instance S for M, computes $cert_\Box(q, M, S)$ and $cert_\Diamond(q, M, S)$, respectively.

Proof sketch. Let us consider the case of computing $cert_{\Box}(q, M, S)$ (the proof for the case of computing $cert_{\Diamond}(q, M, S)$ is the same, except that $\text{CanSol}(M, S)$ must be replaced by $\text{Core}(M, S)$). Given a source instance S for M, first compute $T_0 := \text{CanSol}(M, S)$. If T_0 does not exist, then S has no CWA-solution under M, so that $cert_{\Box}(q, M, S) = \emptyset$. Otherwise, Corollary 3.37 tells us that $cert_{\Box}(q, M, S) = \Box_M q(T_0)$.

Let $\mathbb{A}$ be the algorithm given by Fagin et al. [2005a] for proving Theorem 2.45. This algorithm starts by computing a complete chase sequence C on some universal solution T for S under M and a set Σ' that consists of all t-tgds and egds in Σ and additional egds obtained from q. Here, we let $T = T_0$. If C is failing, then by the construction of $\mathbb{A}$, we have $cert(q, M, S) \neq \emptyset$. In particular, $cert_{\Box}(q, M, S) \neq \emptyset$.

Assume that C is successful, and let T' be the result of C. If $T' \models q$, then by the construction of $\mathbb{A}$, we have $cert(q, M, S) \neq \emptyset$, and thus, $cert_{\Box}(q, M, S) \neq \emptyset$.

Otherwise, if $T' \not\models q$, we claim that $cert_{\Box}(q, M, S) = \emptyset$. To see this, note that since Σ' consists of egds only, T' is a CWA-presolution for S under M, so that there is a homomorphism from T_0 to T'. In particular, there is an injective valuation v of T' such that $q(T') = q(v(T'))$, and by the properties of $T_0 = \text{CanSol}(M, S)$, we have $v(T') \in poss_M(T_0)$. Consequently, there is some $\hat{T} \in poss_M(T_0)$ such that $\hat{T} \not\models q$. This yields $cert_{\Box}(q, M, S) = \Box_M q(T_0) = \emptyset$. $\Box$

Finally, it is easy to see that for every schema mapping M defined by full tgds and egds, and each FO query q over M's target schema, there is a polynomial time algorithm that takes a source instance S for M as input, and outputs $cert_{\Box}(q, M, S) = cert_{\Diamond}(q, M, S) = maybe_{\Box}(q, M, S) = maybe_{\Diamond}(q, M, S)$. Instead of a source instance S, we could also evaluate the query on a CWA-solution for S under M (note that there is at most one such CWA-solution for each source instance S under M).

Table 3.3 summarizes the present section's results on the complexity of $\text{EVAL}_{cert_{\Box}}(M, q)$ for various restrictions of the schema mapping M and the query language from which q is chosen. In Table 3.3, UCQ is the class of all unions of conjunctive queries.

3.5 Query Answering Based on Variants of CWA-Solutions

Sometimes, the assumptions on which CWA-solutions, and thus, the CWA-answers semantics, are based on are too strong or too weak. For this reason, Libkin and Sirangelo [2008] and Afrati and Kolaitis [2008] respectively proposed generalizations and specializations of CWA-solutions and corresponding query answering semantics that seem to work better in some situations. This section gives a short overview of these semantics. We will just give the definitions.

restriction of schema mapping \ query language	UCQ	UCQs, at most one ineq. per disjunct	FO
weakly acyclic		co-NP-hard (Thm. 3.45)	undecidable (Prop. 3.41)
richly acyclic		co-NP-complete (Thm. 3.44 & 3.45)	
only st-tgds & egds	PTIME (Section 3.4.1)		co-NP-complete (Thm. 3.44 & 2.44)
only st-tgds		PTIME (Prop. 3.46)	
only full tgds & egds			PTIME

Table 3.3: Complexity of $\textsc{Eval}_{cert_\square}(M, q)$ for certain restrictions of the schema mapping M and the query q.

For motivations and further results, I refer the interested reader to Libkin and Sirangelo [2008] and Afrati and Kolaitis [2008].

Let us first consider the query answering semantics proposed by Libkin and Sirangelo [2008]. These are based on a generalization of CWA-solutions, which we call *mixed world-solutions*. Mixed world-solutions were defined for the case of schema mappings defined by st-tgds only.

Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping defined by st-tgds, and let S be a source instance for M. Mixed world-solutions for S under M are still required to satisfy the first two requirements in Table 3.1. However, we *annotate* each position in such a solution as either *open* or *closed*, and we relax the third requirement in Table 3.1. When answering a query on a mixed world-solution T for S under M, the open and closed positions of T are taken into account as follows. Instead of restricting the set of all possible worlds of T to be the set of all images of T under some valuation of T, we take into account all ground target instances T' for M that contain the image of T under some valuation of T and for which each atom of $T' \setminus T$ coincides with some atom of T on all closed positions. In other words, atoms can be replicated arbitrarily often by instantiating open positions with new values.

Let us now explain how this is formally captured. Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ is a set of st-tgds. An *annotation for a st-tgd* χ in

Σ of the form

$$\forall\bar{x}\forall\bar{y}\Big(\varphi(\bar{x},\bar{y}) \to \exists\bar{z} \bigwedge_{1\leq i\leq n} R_i(u_{i,1},\ldots,u_{i,\mathrm{ar}(R_i)})\Big)$$

is a mapping $\lambda\colon P_\chi \to \{cl, op\}$, where $P_\chi := \{(i,j) \mid 1 \leq i \leq n,\ 1 \leq j \leq \mathrm{ar}(R_i)\}$ is the set of all positions in ψ. That is, each occurrence of a constant or variable in ψ is annotated with either *closed* (*cl*) or *open* (*op*). In the following, we write annotations as superscripts of the corresponding positions in ψ.

3.47 Example (Library database restructuring, continued)
Consider the st-tgd χ_2' from Example 1.15. An annotated variant of χ_2' is

$$\forall x_1\forall x_2\Big(\mathit{Authors}(x_1,x_2) \to \exists z(\mathit{AuthorList}(z^{cl}, x_2^{op}) \wedge \mathit{WrittenBy}(x_1^{cl}, z^{op}))\Big).$$

Here, the occurrence of z in the first atom and the occurrence of x_1 in the second atom are annotated by *cl*. Furthermore, the occurrence of z in the second atom and the occurrence of x_2 in the first atom are annotated by *op*. □

An *annotation for* M is a mapping Λ_M that assigns to each st-tgd χ in Σ an annotation $\Lambda_M(\chi)\colon P_\chi \to \{cl, op\}$ for χ. An *annotated schema mapping* is a pair (M, Λ_M) consisting of a schema mapping M defined by st-tgds, and an annotation Λ_M for M.

Annotated schema mappings give rise to *annotated target instances*. Here, an *annotation for an instance* I is a mapping Λ_I that assigns to each atom $A = R(t_1,\ldots,t_{\mathrm{ar}(R)})$ of I a finite set $\Lambda_I(A)$ of mappings $\lambda\colon \{1,\ldots,\mathrm{ar}(R)\} \to \{cl, op\}$, which are called *annotations of the positions* of A. Strictly speaking, each mapping $\lambda \in \Lambda_I(A)$ represents an *annotated atom* $(R(t_1,\ldots,t_{\mathrm{ar}(R)}), \lambda)$. In particular, Λ_I annotates each occurrence of a value in I with *closed* (*cl*) or *open* (*op*) or both. In a similar way as for annotations of st-tgds, we will denote annotations for instances as superscripts at the corresponding positions. An *annotated instance* over a schema σ is a pair (I, Λ_I) consisting of an instance I over σ, and an annotation Λ_I for I. An *annotated target instance* for an annotated schema mapping (M, Λ_M) is an annotated instance (T, Λ_T) over M's target schema.

Annotated schema mappings produce the following annotation for the canonical universal solution.

3.48 Definition (Annotation for the canonical universal solution)
Let (M, Λ_M) be an annotated schema mapping, and let S be a source instance for M. We define the following annotation $\Lambda_{M,\Lambda_M,S}$ for $\mathrm{CanSol}(M,S)$.

Recall that $\mathrm{CanSol}(M,S) = \bigcup_{j\in\mathcal{J}_{M,S}} \mathrm{atoms}(\zeta, j)$ for some injective mapping ζ from $\mathcal{J}^*_{M,S}$ to *Null* (cf., Definition 3.5). Let $j = (\chi, \bar{u}, \bar{v}) \in \mathcal{J}_{M,S}$, where χ has

the form

$$\forall\bar{x}\forall\bar{y}\Big(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z} \bigwedge_{1\leq i\leq n} R_i(u_{i,1},\ldots,u_{i,\mathrm{ar}(R_i)})\Big),$$

and let α be a mapping that is the identity on constants, and that maps each variable w that occurs in $\bar{x}$ or $\bar{z}$ to the corresponding value assigned to w by $\bar{u}$ and $\zeta(j)$, respectively. For each $i \in \{1,\ldots,n\}$ and each $p \in \{1,\ldots,\mathrm{ar}(R_i)\}$, let $\Lambda_{M,\Lambda_M,S}$ annotate the p-th position of the atom $A := R_i(\alpha(u_{i,1}),\ldots,\alpha(u_{i,\mathrm{ar}(R_i)}))$ by $\Lambda_M(\chi)(i,p)$

3.49 Definition (Library database restructuring, continued)
Consider the schema mapping $M' = (\sigma_\mathrm{s}, \sigma_\mathrm{t}, \Sigma')$ from Example 3.2. An annotation $\Lambda_{M'}$ for M' could be as follows:

$$\forall x_1\forall x_2\Big(\mathit{Books}(x_1,x_2) \rightarrow \exists z\,\mathit{BookInfo}(x_1^{cl}, x_2^{op}, z^{cl})\Big),$$
$$\forall x_1\forall x_2\Big(\mathit{Authors}(x_1,x_2) \rightarrow \exists z(\mathit{AuthorList}(z^{cl}, x_2^{op}) \wedge \mathit{WrittenBy}(x_1^{cl}, z^{op}))\Big).$$

Given the source instance S^* from Example 1.4, $T := \mathrm{CanSol}(M', S^*)$ is then annotated by $\Lambda_{M',\Lambda_{M'},S^*}$ as follows:

$$\begin{aligned}
\mathit{BookInfo}^T &= \{(\text{“0-201-53771-0”}^{cl}, \text{“Foundations of Databases”}^{op}, \bot_1^{cl})\},\\
\mathit{AuthorList}^T &= \{(\bot_4^{cl}, \text{“Serge Abiteboul”}^{op}), (\bot_5^{cl}, \text{“Richard Hull”}^{op}),\\
&\qquad (\bot_6^{cl}, \text{“Victor Vianu”}^{op}), (\bot_7^{cl}, \text{“Christos H. Papadimitriou”}^{op})\},\\
\mathit{WrittenBy}^T &= \{(\text{“0-201-53771-0”}^{cl}, \bot_4^{op}), (\text{“0-201-53771-0”}^{cl}, \bot_5^{op}),\\
&\qquad (\text{“0-201-53771-0”}^{cl}, \bot_6^{op}), (\text{“0-201-53082-1”}^{cl}, \bot_7^{op})\}.
\end{aligned}$$

Let (M,Λ_M) be an annotated schema mapping, and let S be a source instance for M. Analogous to CWA-presolutions in Section 3.1.1, a *mixed world-presolution* for S under (M,Λ_M) is an annotated target instance (T,Λ_T) for (M,Λ_M) such that there is a homomorphism h from $\mathrm{CanSol}(M,S)$ to T with the property that $h(\mathrm{CanSol}(M,S)) = T$, and *the annotation $\Lambda_{M,\Lambda_M,S}$ is preserved*. To make this more precise, let (I_1,Λ_1) and (I_2,Λ_2) be annotated instances. Then a *homomorphism from (I_1,Λ_1) to (I_2,Λ_2)* is a homomorphism h from I_1 to I_2 such that for all atoms $R(\bar{t})$ of I_1, we have $\Lambda_1(R(\bar{t})) = \Lambda_2(R(h(\bar{t})))$. A mixed world-presolution for S under (M,Λ_M) is then an annotated target instance (T,Λ_T) for (M,Λ_M) such that there is a homomorphism h from $(\mathrm{CanSol}(M,S),\Lambda_{M,\Lambda_M,S})$ to (T,Λ_T) with $h(\mathrm{CanSol}(M,S)) = T$.

As mentioned above, mixed world-solutions relax the third requirement in Table 3.1. For the formalization of this requirement, I refer the interested reader to Libkin and Sirangelo [2008]. Here, we will use the following characterization, given by Libkin and Sirangelo [2008]. Namely, to satisfy the third requirement

in Table 3.1, a mixed world-presolution for S under (M, Λ_M) must have a homomorphism into an *expansion* of $(\text{CanSol}(M, S), \Lambda_{M,\Lambda_M,S})$. Formally, we call an annotated instance (I_2, Λ_2) an *expansion of an annotated instance* (I_1, Λ_1) if and only if $I_1 \subseteq I_2$, and all atoms of $I_2 \setminus I_1$ coincide with some atom of I_1 on all closed positions. That is, for all atoms $R(t'_1, \ldots, t'_{\text{ar}(R)}) \in I_2 \setminus I_1$, there is an atom $A := R(t_1, \ldots, t_{\text{ar}(R)}) \in I_1$ and some $\lambda \in \Lambda_1(A)$ such that for all $p \in \{1, \ldots, \text{ar}(R)\}$ with $\lambda(A, p) = cl$ we have $t'_p = t_p$.

Altogether, mixed world-solutions are defined as follows:

3.50 Definition (Mixed world-solution)
Let (M, Λ_M) be an annotated schema mapping, and let S be a source instance for M. A *mixed world-solution* for S under (M, Λ_M) is an annotated target instance (T, Λ_T) for (M, Λ_M) such that

- there is a homomorphism h from $(\text{CanSol}(M, S), \Lambda_{M,\Lambda_M,S})$ to (T, Λ_T) with $h(\text{CanSol}(M, S)) = T$, and
- there is a homomorphism from (T, Λ_T) to an expansion of $(\text{CanSol}(M, S), \Lambda_{M,\Lambda_M,S})$.

Let $sol_{\text{mixed}}(M, \Lambda_M, S)$ be the set of all mixed world-solutions for S under (M, Λ_M).

Given an annotated schema mapping (M, Λ_M) with $M = (\sigma_s, \sigma_t, \Sigma)$, a source instance S for M, and a query q over σ_t, we can now answer q on M and S by the certain answers to q on the set

$$\bigcup_{(T,\Lambda_T) \in sol_{\text{mixed}}(M,\Lambda_M,S)} poss(T, \Lambda_T),$$

where for each annotated instance (T, Λ_T) over σ_t, we let

$$\begin{aligned} poss(T, \Lambda_T) := \{\hat{T} \in \text{inst}(\sigma_t) \mid\ & \hat{T} \text{ is ground, } v(T) \subseteq \hat{T} \text{ for some valuation } v \\ & \text{of } T, \text{ for each } R(t'_1, \ldots, t'_{\text{ar}(R)}) \in \hat{T} \text{ there are} \\ & A = R(t_1, \ldots, t_{\text{ar}(R)}) \in T \text{ and } \lambda \in \Lambda_T(A) \\ & \text{such that for all positions } p \in \{1, \ldots, \text{ar}(R)\} \\ & \text{with } \lambda(p) = cl \text{ we have } t'_p = t_p\}. \end{aligned}$$

Finally, let us consider the query answering semantics proposed by Afrati and Kolaitis [2008], which we call *endomorphic images semantics*. Like the mixed world-solution based semantics, the endomorphic images semantics have been defined for the case of schema mappings defined by st-tgds only. Furthermore, it has been designed with the goal of answering *aggregate queries*,

where Afrati and Kolaitis [2008] argued that one needs to restrict the notion of CWA-solutions in order to obtain "good" answers. However, aggregate queries are not the focus of this thesis; for more on the topic of answering aggregate queries, see Afrati and Kolaitis [2008]. Under the endomorphic images semantics, queries are answered by the certain answers on the endomorphic images of the canonical universal solution. Here, an *endomorphism* of an instance I is a homomorphism from I to I. An *endomorphic image* of I is an instance J for which there is an endomorphism h of I with $h(I) = J$. Thus, given a schema mapping $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$ defined by st-tgds, a source instance S for M, and a query q over σ_{t}, q is answered under the endomorphic images semantics by the certain answers to q on the set of all endomorphic images of $\mathrm{CanSol}(M, S)$.

3.6 Limitations to the Justification-Based Approach

The requirements for CWA-solutions (recall Table 3.1) reflect the operational point of view on tgds and egds. In many situations, this yields query results that are intuitively expected (recall the examples given in Section 3.3). On the other hand, if one is used to the standard semantics of FO quantifiers (e.g., existential quantifiers express that there are one, two, three or more elements that satisfy the given property), these semantics may also yield results that intuitively do not seem to be accurate.

3.51 Example
Let $M = (\{E\}, \{F, G\}, \Sigma)$ be the schema mapping, where E, F, G are binary relation symbols, and Σ consists of the st-tgd

$$\chi \;:=\; \forall x_1 \forall x_2 \Big(E(x_1, x_2) \rightarrow \exists z (F(x_1, z) \wedge G(z, x_2)) \Big).$$

Let S be the source instance for M with $E^S = \{(a, b)\}$. Then $T = \mathrm{CanSol}(M, S)$ with $F^T = \{(a, \bot)\}$ and $G^T = \{(\bot, b)\}$ is the unique CWA-solution for S under M, up to isomorphism. For the query

$$q(x) \;:=\; \exists z \Big(F(x, z) \wedge \forall z' (F(x, z') \rightarrow z' = z) \Big),$$

it follows therefore that $\mathit{cert}_{\square}(q, M, S) = \{a\}$. That is, $\mathit{cert}_{\square}(q, M, S)$ excludes the possibility that there is more than one value z with $F(a, z)$. However, this is inconsistent with χ and S, which, taking the usual semantics of existential quantification, tell us that there are *one or more* z with $F(a, z)$ and $G(z, b)$. In particular, χ and S explicitly state that it is possible that there is more than one z with $F(a, z)$.

This example can be modified to show that also the other CWA-solution-based semantics lead to answers that intuitively do not seem to be right. For

$\mathit{cert}_\Diamond$, we only have to replace $\mathit{cert}_\Box$ with $\mathit{cert}_\Diamond$. For $\mathit{maybe}_\Box$ and $\mathit{maybe}_\Diamond$, we take the query $\neg q$ instead of q. Then $\mathit{maybe}_\Diamond(q, M, S) = \mathit{maybe}_\Box(q, M, S) = \emptyset$. Thus, $\mathit{maybe}_\Diamond(q, M, S)$ and $\mathit{maybe}_\Box(q, M, S)$ tell us that it is *not* possible that there are two distinct z with $F(a, z)$, which, as above, is intuitively inconsistent with M and S.

By choosing another query q, we can also show that the mixed world-based semantics can lead to answers that intuitively do not seem to be right. □

Furthermore, the semantics based on CWA-solutions and its variants do not respect logical equivalence of schema mappings. Here, given schema mappings $M_1 = (\sigma_\mathrm{s}, \sigma_\mathrm{t}, \Sigma_1)$ and $M_2 = (\sigma_\mathrm{s}, \sigma_\mathrm{t}, \Sigma_2)$ over the same source schema σ_s and target schema σ_t, we call M_1 and M_2 *logically equivalent* if and only if Σ_1 and Σ_2 are logically equivalent, that is, if each instance I over $\sigma_\mathrm{s} \cup \sigma_\mathrm{t}$ satisfies

$$I \models \Sigma_1 \iff I \models \Sigma_2.$$

In particular, given an instance S over σ_s, each solution for S under M_1 is a solution for S under M_2 and vice versa. Logical equivalence of schema mappings is respected if the answer to a query is the same on logically equivalent schema mappings. The next example shows that the semantics based on CWA-solutions and its variants do not respect logical equivalence of schema mappings:

3.52 Example
Let $M_1 = (\sigma_\mathrm{s}, \sigma_\mathrm{t}, \Sigma_1)$ and $M_2 = (\sigma_\mathrm{s}, \sigma_\mathrm{t}, \Sigma_2)$ be schema mappings, where σ_s contains a unary relation symbol P, σ_t contains a binary relation symbol E,

$$\Sigma_1 \;:=\; \Big\{ \forall x \Big(P(x) \to E(x, x) \Big) \Big\},$$

and

$$\Sigma_2 \;:=\; \Sigma_1 \cup \Big\{ \forall x \Big(P(x) \to \exists z\, E(x, z) \Big) \Big\}.$$

Then M_1 and M_2 are logically equivalent.

Now let S be an instance over σ_s with $P^S = \{a\}$. Furthermore, let T_1 and T_2 be instances over σ_t with $E^{T_1} = \{(a, a)\}$ and $E^{T_2} = \{(a, a), (a, \bot)\}$. Note that T_1 is the unique CWA-solution for S under M_1, and that $T_2 = \mathrm{CanSol}(M_2, S)$ is a CWA-solution for S under M_2. Thus, for the query

$$q(x) \;:=\; \exists z \Big(E(x, z) \wedge \forall z' (E(x, z') \to z' = z) \Big),$$

we obtain different answers $\mathit{cert}_\Box(q, M_1, S) = \{a\}$ and $\mathit{cert}_\Box(q, M_2, S) = \emptyset$ to the same query q on logically equivalent schema mappings M_1 and M_2.

Using M_1, M_2 and the query q in Example 3.52, one can also show that the mixed world-based semantics, and the semantics based on endomorphic images of the canonical universal solution do not respect logical equivalence of schema mappings either. □

Note that, given logically equivalent schema mappings $M_1 = (\sigma_s, \sigma_t, \Sigma_1)$ and $M_2 = (\sigma_s, \sigma_t, \Sigma_2)$, the sets Σ_1 and Σ_2 intuitively describe one and the same transformation. If two descriptions of transformations are logically equivalent, it should intuitively not matter if a query is answered with respect to the one description or the other one.

The goal in the following two chapters is to identify semantics such that

1. implicit information in schema mappings and source instances are taken into account,
2. logical equivalence of schema mappings is respected, and
3. the standard semantics of FO quantifiers is reflected.

4 Deductive Databases & Relational Data Exchange

As mentioned in Chapter 3, the concept of CWA-solutions is inspired by the *closed world assumption (CWA)*. Originally, the CWA has been introduced by Reiter [1978] for answering non-monotonic queries on *deductive databases*, where a deductive database is a set of universally quantified disjunctions of relational atomic FO formulas and negations of relational atomic FO formulas (see, e.g., Gallaire et al. [1984]). A large body of work in the area of deductive databases deals with semantics for answering non-monotonic queries. Interestingly, the definitions of many of these semantics apply as well, or can be easily extended, to more general sets of logical sentences, such as sets

$$\Sigma \cup \{R(\bar{c}) \mid R \in \sigma_{\mathrm{s}},\, \bar{c} \in R^S\} \cup \{\neg R(\bar{c}) \mid R \in \sigma_{\mathrm{s}},\, \bar{c} \in \mathit{Const}^{\mathrm{ar}(R)} \setminus R^S\},$$

where Σ is the set of sentences of a schema mapping $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$, and S is a source instance for M. It is therefore tempting to use these semantics for query answering in relational data exchange.

In this chapter, we translate various semantics for answering non-monotonic queries on deductive databases into the framework of relational data exchange. Furthermore, we study these semantics with respect to the following question: which of these semantics is appropriate for answering non-monotonic queries in relational data exchange, in the sense that logical equivalence of schema mappings is respected, and the standard semantics of FO quantifiers is reflected? As it turns out, none of these semantics seems to be appropriate. For each of these semantics we show that it is either too weak, too strong, or not preserved under logical equivalence. Nevertheless, the ideas presented in the present chapter are the starting point of—and may help to better understand—the semantics developed in Chapter 5.

Section 4.1 begins by defining deductive databases, and by explaining their relationship to relational data exchange. The subsequent sections 4.2 to 4.4 then study the semantics for query answering on deductive databases in the context of relational data exchange.

4.1 Definition of Deductive Databases

In this section, we define deductive databases, and explain their relationship to relational data exchange.

4.1 Definition (Deductive database; see, e.g., Gallaire et al. [1984])
A *deductive database* over a schema σ is a set of *clauses* over σ, which are FO sentences over σ of the form

$$\forall \bar{x}\Big(\neg R_1(\bar{u}_1) \vee \cdots \vee \neg R_m(\bar{u}_m) \vee R'_1(\bar{v}_1) \vee \cdots \vee R'_n(\bar{v}_n)\Big),$$

where m and n are nonnegative integers with $m+n \geq 1$. A *model* of a deductive database D over σ is a *ground* instance I over σ with $I \models D$.

Given a deductive database D, we may view the subset of D that consists of all clauses of the form $R(\bar{c})$ with $\bar{c} \in \mathit{Const}^{\mathrm{ar}(R)}$ (i.e., clauses with $m = 0$, $n = 1$, and no variables) as a ground instance, namely the instance I with $R^I = \{\bar{c} \mid R(\bar{c}) \in D\}$ for all $R \in \sigma$. A deductive database thus consists of *extensional data*, which corresponds to a ground instance, and *intensional data*, from which new data can be deduced.

4.2 Example (Ground instances)
Let I be a ground instance over a schema σ. Then I can be represented by the following deductive database D_I over σ:

$$D_I \;:=\; \{R(\bar{c}) \mid R \in \sigma,\, \bar{c} \in R^I\} \cup \{\neg R(\bar{c}) \mid R \in \sigma,\, \bar{c} \in \mathit{Const}^{\mathrm{ar}(R)} \setminus R^I\}.$$

Clearly, I is the only model of D_I. □

4.3 Example (Schema mappings defined by full tgds)
Let $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$, where Σ is a set of full tgds, and let S be a source instance for M. Note that every full tgd

$$\chi \;=\; \forall \bar{x}\Big(R_1(\bar{u}_1) \wedge \cdots \wedge R_m(\bar{u}_m) \rightarrow R'(\bar{v})\Big)$$

in Σ can be rewritten as a clause

$$\chi' \;:=\; \forall \bar{x}\Big(\neg R_1(\bar{u}_1) \vee \cdots \vee \neg R_m(\bar{u}_m) \vee R'(\bar{v})\Big).$$

Thus, M and S can be represented by the following deductive database $D'_{M,S}$ over $\sigma_{\mathrm{s}} \cup \sigma_{\mathrm{t}}$:

$$D'_{M,S} \;:=\; D_S \cup \{\chi' \mid \chi \in \Sigma\},$$

where D_S, as defined in Example 4.2, is the deductive database that corresponds to S. Note that for every ground instance I over $\sigma_{\mathrm{s}} \cup \sigma_{\mathrm{t}}$, I is a model of $D'_{M,S}$ if and only if there is a ground solution T for S under M with $I = S \cup T$. □

Given a deductive database D over a schema σ, a query q over σ is usually answered on D by $\mathit{cert}(q, \mathcal{I})$, where $\mathcal{I}$ is a set of models of D that depends on the particular query answering semantics. Various semantics for answering queries on deductive databases exist, among them the above-mentioned CWA-based semantics [Reiter, 1978], the semantics based on the *generalized CWA (GCWA)* [Minker, 1982], the semantics based on the *extended GCWA (EGCWA)* [Yahya and Henschen, 1985], and the *possible worlds semantics (PWS)* [Chan, 1993]. What makes these semantics interesting for relational data exchange is that the definitions of these semantics apply as well, or can be extended, to more general sets of logical sentences, such as

$$D_{M,S} \;:=\; D_S \cup \Sigma,$$

where Σ is the set of logical sentences of a schema mapping $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$, S is a source instance for M, and D_S is defined as in Example 4.2. As shown in Example 4.3, if Σ consists only of full tgds, it is even the case that $D_{M,S}$ is logically equivalent to a deductive database.

In the following, we study the CWA, the GCWA, the EGCWA and the PWS in more detail in the context of relational data exchange.

4.2 The Closed World Assumption (CWA)

The *closed world assumption (CWA)*, first formalized by Reiter [1978], assumes that every atomic formula $R(\bar{c})$ with $\bar{c} \in \mathit{Const}^{\mathrm{ar}(R)}$ that is not implied by a database is false. This is a common assumption for relational databases.

Reiter formalized the CWA, and defined a query answering semantics for deductive databases based on the CWA, as follows. For a set Φ of logical formulas and a formula ψ, we write $\Phi \models \psi$ if and only if ψ *logically follows from* Φ, that is, if for all instances I with $I \models \Phi$ we have $I \models \psi$. Let D be a deductive database over a schema σ. Under the CWA, all atomic formulas in the set

$$\overline{D} \;:=\; \{\neg R(\bar{c}) \mid R \in \sigma,\, \bar{c} \in \mathit{Const}^{\mathrm{ar}(R)},\, D \not\models R(\bar{c})\}$$

are assumed to be false. The models of $D \cup \overline{D}$ are then called the *CWA-models* of D, and a query q over σ is answered by $\mathit{cert}(q, \mathcal{I})$, where $\mathcal{I}$ is the set of all CWA-models of D.

4.4 Example (Ground instances)
Let I be a ground instance over a schema σ, and consider the deductive database

$$D \;:=\; \{R(\bar{c}) \mid R \in \sigma,\, \bar{c} \in R^I\}.$$

The models of D are all the ground instances J over σ with $I \subseteq J$. However, the only CWA-model of D is I. In particular, we have $D \cup \overline{D} = D_I$. □

Translating CWA models and the CWA-based query answering semantics into the framework of relational data exchange, we obtain:

4.5 Definition (RCWA-solution, RCWA-answers)
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, let S be a source instance for M, and let q be a query over σ_t.

1. A *RCWA-solution* for S under M is a ground target instance T for M such that $S \cup T$ is a CWA-model of $D_{M,S}$.

2. The *RCWA-answers to q on M and S* are defined as $cert_{\mathrm{RCWA}}(q, M, S) := cert(q, \mathcal{I})$, where $\mathcal{I}$ is the set of all RCWA-solutions for S under M.

The RCWA-answers semantics coincides with the certain CWA-answers semantics on schema mappings defined by full tgds and egds:

4.6 Proposition.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of full tgds and egds. Then for all source instances S for M, and all queries q over σ_t, we have $cert_{RCWA}(q, M, S) = cert_{\Box}(q, M, S)$.

Proof. If there is no solution for S under M, we have $cert_{\mathrm{RCWA}}(q, M, S) = cert_{\Box}(q, M, S) = \emptyset$. So assume that there is a solution for S under M. Since Σ consists of full tgds and egds, it is easy to see that there is a unique $\subseteq$-minimal solution T_0 for S under M. This means that T_0 is a solution for S under M, and there is no solution T_0' for S under M with $T_0' \subsetneq T_0$. Note that T_0 is ground.

Furthermore, T_0 is the unique RCWA-solution for S under M. Indeed, for every ground solution $T \supseteq T_0$ for S under M, every $R \in \sigma_t$ and every $\bar{t} \in R^T \setminus R^{T_0}$, we have $D_{M,S} \not\models R(\bar{t})$, because $S \cup T_0 \models D_{M,S}$ and $S \cup T_0 \not\models R(\bar{t})$. Thus, $cert_{\mathrm{RCWA}}(q, M, S) = q(T_0)$.

Finally, T_0 is the unique CWA-solution for S under M, and consequently, $cert_{\mathrm{RCWA}}(q, M, S) = cert_{\Box}(q, M, S) = q(T_0)$. □

However, for schema mappings that contain non-full tgds, $cert_{\mathrm{RCWA}}$ may lead to answers that are intuitively inconsistent with M and S. This is illustrated by the following example, which is based on Example 8 in Reiter [1978].

4.7 Example
Let $M = (\{P\}, \{E\}, \Sigma)$, where Σ consists of the st-tgd $\forall x(P(x) \to \exists z E(x, z))$,

and let S be the source instance for M with $P^S = \{a\}$. Then there is no RCWA-solution for S under M, because for all $b, c \in \mathit{Const}$, we have

$$D_{M,S} \;=\; D_S \cup \{\forall x(P(x) \rightarrow \exists z E(x,z))\} \;\not\models\; E(b,c).$$

This is due to the fact that the instance I with $P^I = \{a\}$ and $E^I = \{(a,c')\}$, where $c' \in \mathit{Const} \setminus \{c\}$, satisfies $D_{M,S}$, but not $E(b,c)$. Therefore,

$$\overline{D_{M,S}} \;=\; \{\neg P(b) \mid b \in \mathit{Const},\, b \neq a\} \cup \{\neg E(b,c) \mid b, c \in \mathit{Const}\}.$$

For the query

$$q(x) \;:=\; \exists z\, E(x,z),$$

we thus have $\mathit{cert}_{\mathrm{RCWA}}(q, M, S) = \emptyset$. In other words, $\mathit{cert}_{\mathrm{RCWA}}(q, M, S)$ tells us that there is no value z such that $E(a,z)$ holds. This is clearly inconsistent with M and S, since M and S tell us that there is a value z such that $E(a,z)$ holds. Thus, intuitively, the set of answers should be $\{a\}$. □

4.3 The Generalized Closed World Assumption (GCWA)

Minker [1982] extended the CWA to the *generalized closed world assumption (GCWA)* as follows.

We say that an instance I is $\subseteq$-*minimal* in a set $\mathcal{I}$ of instances if and only if $I \in \mathcal{I}$, and there is no $I' \in \mathcal{I}$ with $I' \subsetneq I$. If D is a deductive database over a schema σ, then a $\subseteq$-*minimal model* of D is a $\subseteq$-minimal instance in the set of all models of D. Moreover, if M is a schema mapping and S is a source instance for M, then a $\subseteq$-*minimal solution* for S under M is a $\subseteq$-minimal instance in $\mathit{sol}(M,S)$.

Let D be a deductive database over a schema σ. Under the GCWA, all atomic formulas in the set

$$\begin{aligned}\overline{\overline{D}} \;:=\; \{\neg R(\bar{c}) \mid\; & R \in \sigma,\, \bar{c} \in \mathit{Const}^{\mathrm{ar}(R)}, \text{ and for all } \subseteq\text{-minimal models } I \text{ of } D \\ & \text{we have } \bar{c} \notin R^I\}\end{aligned}$$

are assumed to be false. The models of $D \cup \overline{\overline{D}}$ are called *GCWA-models* of D, and a query q over σ is answered by $\mathit{cert}(q, \mathcal{I})$, where $\mathcal{I}$ is the set of all GCWA-models of D.

The intuition behind the above definitions is that each ground atom in some $\subseteq$-minimal model of D is in some sense an atom that D "speaks" about. For ground atoms that do not occur in any $\subseteq$-minimal model of D, this means that they are merely "invented", and can therefore safely be assumed to be false.

Translating the GCWA and the GCWA-based query answering semantics into the framework of relational data exchange, we obtain:

4.8 Definition (GCWA-solution, GCWA-answers)
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, let S be a source instance for M, and let q be a query over σ_t.

- A *GCWA-solution* for S under M is a ground target instance T for M such that $S \cup T$ is a GCWA-model of $D_{M,S}$.
- The *GCWA-answers to q on M and S* are defined as $cert_{\mathrm{GCWA}}(q, M, S) := cert(q, \mathcal{I})$, where $\mathcal{I}$ is the set of all GCWA-solutions for S under M.

As for the RCWA-answers semantics, the GCWA-answers semantics coincides with the certain CWA-answers semantics on schema mappings defined by full tgds and egds.

4.9 Proposition.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of full tgds and egds. Then for all source instances S for M, and all queries q over σ_t, we have $cert_{GCWA}(q, M, S) = cert_{\Box}(q, M, S)$.

Proof. If there is no solution for S under M, we have $cert_{\mathrm{GCWA}}(q, M, S) = cert_{\Box}(q, M, S) = \emptyset$. Assume that there is a solution for S under M. As shown in the proof of Proposition 4.6, there is a unique $\subseteq$-minimal solution T_0 for S under M, which is ground. It follows immediately from Definition 4.8 that T_0 is the unique GCWA-solution for S under M. Since T_0 is the unique CWA-solution for S under M, we have $cert_{\mathrm{GCWA}}(q, M, S) = cert_{\Box}(q, M, S) = q(T_0)$. □

Moreover, the GCWA-answers semantics yields the expected answer to the query in Example 4.7:

4.10 Example
Recall the schema mapping M, the source instance S, and the query q from Example 4.7. We now have

$$\overline{\overline{D_{M,S}}} \;=\; \{\neg P(b) \mid b \in \mathit{Const},\, b \neq a\} \cup \{\neg E(b,c) \mid b, c \in \mathit{Const},\, b \neq a\},$$

because for all $c \in \mathit{Const}$ there is a $\subseteq$-minimal model I of $D_{M,S}$ with $(a, c) \in E^I$, and for all $b, c \in \mathit{Const}$ with $b \neq a$ and all $\subseteq$-minimal models I of $D_{M,S}$, we have $(b, c) \notin E^I$. Therefore, the GCWA-solutions for S under M are precisely the target instances T for M for which there is a finite nonempty set $C \subseteq \mathit{Const}$ with $T = T_C$, where $E^{T_C} = \{(a, b) \mid b \in C\}$. It follows that $cert_{\mathrm{GCWA}}(q, M, S) = \{a\}$, as desired. □

Nevertheless, there are cases where the GCWA still seems to be too weak in the sense that it removes not enough solutions from the set of all solutions.

4.11 Example
Consider a slight extension of the schema mapping from Example 4.7, namely the schema mapping $M = (\{P\}, \{E, F\}, \Sigma)$, where Σ consists of the st-tgd

$$\chi \;:=\; \forall x\Big(P(x) \to \exists z_1 \exists z_2 (E(x, z_1) \wedge F(z_1, z_2))\Big).$$

Let S be the source instance for M with $P^S = \{a\}$. Then,

$$\overline{\overline{D_{M,S}}} \;=\; \{\neg P(b) \mid b \in \mathit{Const},\, b \neq a\} \cup \{\neg E(b, c) \mid b, c \in \mathit{Const},\, b \neq a\}.$$

Note that for all $b, c \in \mathit{Const}$ we have $\neg F(b, c) \notin \overline{\overline{D_{M,S}}}$, since the instance I with $P^I = \{a\}$, $E^I = \{(a, b)\}$ and $F^I = \{(b, c)\}$ is a $\subseteq$-minimal model of $D_{M,S}$. So, the GCWA-solutions for S under M are the target instances T for M for which there is a finite nonempty set $C \subseteq \mathit{Const}$ with the following properties:

- $E^T = \{(a, b) \mid b \in C\}$, and
- for at least one $b \in C$ there is some $c \in \mathit{Const}$ with $(b, c) \in F^T$.

In particular, the target instance T^* for M with $E^{T^*} = \{(a, b)\}$ and $F^{T^*} = \{(b, c), (d, e)\}$ is a GCWA-solution for S under M. For the query

$$q \;:=\; \forall z_1 \forall z_2 \Big(F(z_1, z_2) \to \exists x\, E(x, z_1)\Big)$$

we thus have $\mathit{cert}_{\mathrm{GCWA}}(q, M, S) = \emptyset$.

So, $\mathit{cert}_{\mathrm{GCWA}}(q, M, S)$ tells us that it is possible that there is a tuple (b, c) in F for which (a, b) is not in E. However, χ and S intuitively do not "mention" this possibility. In particular, χ and S only tell us that there are one or more pairs $(b, c) \in \mathit{Const}^2$ such that $E(a, b)$ and $F(b, c)$ occur together in a solution. Thus, whenever $E(a, b)$ is present for some $b \in \mathit{Const}$, then $F(b, c)$ should be present for some $c \in \mathit{Const}$. Similarly, whenever $F(b, c)$ is present for some $b, c \in \mathit{Const}$, then $E(a, b)$ should be present. □

4.4 Concepts Related to the GCWA

Various extensions of the GCWA have been proposed in the literature. One of these extensions is the *extended GCWA (EGCWA)* by Yahya and Henschen [1985], which restricts the set of models of a deductive database D to the $\subseteq$-minimal models of D. So, given a schema mapping $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$ and a source instance S for M, an *EGCWA-solution* for S under M can be defined as a ground $\subseteq$-minimal solution for S under M, and given a query q over σ_{t} we can define

$$\mathit{cert}_{\mathrm{EGCWA}}(q, M, S) \;:=\; \mathit{cert}(q, \mathcal{I}),$$

where $\mathcal{I}$ is the set of all EGCWA-solutions for S under M. Then, for the schema mapping M, the source instance S for M, and the query q in Example 4.10, $cert_{\text{EGCWA}}(q, M, S) = cert_{\text{GCWA}}(q, M, S)$, and for the schema mapping M, the source instance S for M, and the query q in Example 4.11, $cert_{\text{EGCWA}}(q, M, S) \neq \emptyset$, as desired. However, the EGCWA-based semantics seems to be too strong in the sense that it removes too many solutions from the set of all solutions. This is illustrated by the following example.

4.12 Example (Range queries)
Let $M = (\{P\}, \{E\}, \Sigma)$ be a schema mapping, where Σ consists of

$$\chi \ := \ \forall x(P(x) \rightarrow \exists^{[2,3]} z E(x, z)),$$

where $\exists^{[2,3]} z\, E(x, z)$ is an abbreviation for "there exist two or three z such that $E(x, z)$ holds". Let S be the source instance for M with $P^S = \{a\}$. Then the $\subseteq$-minimal solutions for S under M have the form $\{E(a, b_1), E(a, b_2)\}$, where b_1, b_2 are distinct constants. Thus, for the query

$$q(x) \ := \ \exists z_1 \exists z_2 \Big(E(x, z_1) \wedge E(x, z_2) \wedge \forall z_3 \Big(E(x, z_3) \rightarrow (z_3 = z_1 \vee z_3 = z_2) \Big) \Big),$$

we have $cert_{\text{EGCWA}}(q, M, S) \neq \emptyset$. In other words, the answer $cert_{\text{EGCWA}}(q, M, S)$ excludes the possibility that there are three distinct values b_1, b_2, b_3 such that $E(a, b_i)$ holds for each $i \in \{1, 2, 3\}$. But χ and S explicitly mention this possibility. Thus, intuitively, $cert_{\text{EGCWA}}$ is inconsistent with M and S. □

A semantics that seems to get rid of the above-mentioned problems is the *possible worlds semantics (PWS)* by Chan [1993]. An obvious translation of the PWS into the framework of relational data exchange for the case of schema mappings defined by st-tgds is as follows: Let $M = (\sigma_{\text{s}}, \sigma_{\text{t}}, \Sigma)$ be a schema mapping, where Σ is a set of st-tgds, and let S be a source instance for M. The definition of a PWS-solution for S under M can then be given in terms of *justifications* as defined in Section 3.1. More precisely, a PWS-solution for S under M is a ground solution T for S under M such that all atoms of T are justified in T under M and S. For a query q over σ_{t}, we let

$$cert_{\text{PWS}}(q, M, S) \ := \ cert(q, \mathcal{I}),$$

where $\mathcal{I}$ is the set of all PWS-solutions for S under M. However, $cert_{\text{PWS}}$ does not respect logical equivalence of schema mappings as can be easily verified using the schema mapping, the source instance, and the query in Example 3.52:

4.13 Example ($cert_{\text{PWS}}$ does not respect logical equivalence)
Let $M_1 = (\sigma_{\text{s}}, \sigma_{\text{t}}, \Sigma_1)$ and $M_2 = (\sigma_{\text{s}}, \sigma_{\text{t}}, \Sigma_2)$ be the logically equivalent schema

mappings from Example 3.52. Furthermore, let S be the instance over σ_{s} from Example 3.52. It is not hard to verify that the instance T_1 over σ_{t} with $E^{T_1} = \{(a, a)\}$ is the only PWS-solution for S under M_1. Moreover, it is not hard to verify that the instance T_2 over σ_{t} with $E^{T_2} = \{(a, a), (a, b)\}$, $a \neq b$, is a PWS-solution for S under M_2. Thus, for the query

$$q(x) \;:=\; \exists z\Big(E(x, z) \wedge \forall z'(E(x, z') \rightarrow z' = z)\Big),$$

we obtain different answers $\mathit{cert}_{\mathrm{PWS}}(q, M_1, S) = \{a\}$ and $\mathit{cert}_{\mathrm{PWS}}(q, M_2, S) = \emptyset$ to the same query q on logically equivalent schema mappings M_1 and M_2. $\square$

5 The GCWA*-Answers Semantics

In this chapter, we introduce the concept of GCWA*-solutions and the corresponding GCWA*-answers semantics. GCWA*-solutions are inspired by GCWA-solutions and EGCWA-solutions, introduced in the previous chapter. GCWA*-solutions, and thus GCWA*-answers, are defined for general schema mappings. For schema mappings defined by st-tgds and egds, GCWA*-solutions are ground solutions that are unions of $\subseteq$-minimal solutions. We argue that the GCWA*-answers semantics intuitively meets the three goals identified at the end of Section 3.6:

1. Implicit information in schema mappings and source instances are taken into account.
2. Logical equivalence of schema mappings is respected.
3. The standard semantics of FO quantifiers is reflected.

Thus, it seems that the GCWA*-answers semantics is well-suited for query answering in relational data exchange.

A drawback is that it is in general hard to evaluate GCWA*-answers, even for simple schema mappings and existential queries with only one negated relational atomic formula (Proposition 5.11 and Proposition 5.12). However, we identify a certain kind of st-tgds, called *packed st-tgds*, such that the GCWA*-answers to *universal queries* (FO queries of the form $\forall \bar{x}\, \varphi$, where φ is quantifier-free) can be evaluated in polynomial time on schema mappings defined by packed st-tgds. More precisely, we show (see Theorem 5.15) that for each schema mapping M defined by packed st-tgds and each universal query q, there is a polynomial time algorithm that, given an instance that is isomorphic to $\mathrm{Core}(M, S)$ for some source instance S for M, computes the GCWA*-answers to q on M and S. This is the technically most challenging result of this thesis. Note that, by Theorem 2.28, this result implies that for each schema mapping M defined by packed st-tgds and each universal query q, there is a polynomial time algorithm that takes a source instance S for M and outputs the GCWA*-answers to q on M and S.

This chapter is structured as follows. In Section 5.1, we define *GCWA*-solutions* and the *GCWA*-answers* semantics, and argue that the GCWA*-answers semantics intuitively meets the three goals mentioned above. In Section 5.2, we then give a characterization of GCWA*-solutions for schema map-

pings defined by st-tgds and egds in the spirit of the definition of GCWA-solutions. Section 5.3 presents first results towards understanding the complexity of computing GCWA*-answers. Finally, Section 5.4 deals with the problem of computing GCWA*-answers to universal queries. This section comprises the main result of this chapter.

Most of the results of this chapter were published in Hernich [2010].

5.1 Definition of GCWA*-Solutions and GCWA*-Answers

In this section, we define *GCWA*-solutions* and the *GCWA*-answers* semantics, and argue that the GCWA*-answers semantics intuitively meets the three goals mentioned at the beginning of this chapter.

As a motivating example, let us consider the schema mapping M and the source instance S for M from Example 4.7. Let $\mathcal{T}$ be the set of all GCWA-solutions for S under M. As shown in Example 4.10, $\mathcal{T}$ consists of all target instances T for M such that there is a nonempty finite set $C \subseteq \mathit{Const}$ with $T = T_C$, where

$$E^{T_C} \;:=\; \{(a, b) \mid b \in C\}.$$

Intuitively, $\mathcal{T}$ precisely reflects the explicit and the implicit information in M and S: taking the standard semantics of existential quantification, M and S just tell us that there is one $b \in \mathit{Const}$ such that $E(a, b)$ holds, or there are two distinct $b_1, b_2 \in \mathit{Const}$ such that $E(a, b_1)$ and $E(a, b_2)$ hold, or there are three distinct $b_1, b_2, b_3 \in \mathit{Const}$ such that $E(a, b_1)$, $E(a, b_2)$ and $E(a, b_3)$ hold, and so on. The case that there are n distinct constants $b_1, \ldots, b_n$ such that $E(a, b_i)$ holds for each $i \in \{1, \ldots, n\}$ is captured by T_C, where $C := \{b_1, \ldots, b_n\}$. Since M and S specify a translation of S from source to target, it seems to be natural to assume that the result of the translation is one of the instances in $\mathcal{T}$. Thus, intuitively, the certain answers to a query on $\mathcal{T}$ take into account explicit and implicit information in M and S in such a way that the standard semantics of FO quantifiers is reflected.

Note that $\mathcal{T}$ consists of all ground target instances for M that are unions of $\subseteq$-minimal solutions for S under M. On the other hand, consider the schema mapping M, the source instance S for M, and the GCWA-solution T^* for S under M from Example 4.11. Then T^* is *not* a union of $\subseteq$-minimal solutions for S under M. However, let $\mathcal{T}$ be the set of all ground target instances for M that are unions of $\subseteq$-minimal solutions for S under M. That is, let $\mathcal{T}$ be the set of all ground target instances T for M such that

$$E^T \;=\; \{(a, b) \mid (b, c) \in F^T \text{ for some } c \in \mathit{Const}\} \quad \text{and} \quad F^T \neq \emptyset.$$

Then, intuitively, $\mathcal{T}$ precisely reflects the explicit and the implicit information in M and S as well as the standard semantics of FO quantifiers: taking once again the standard semantics of existential quantification, M and S tell us that there is one pair $(b, c) \in \mathit{Const}^2$ such that $E(a, b)$ and $F(b, c)$ hold, or there are two pairs $(b_1, c_1), (b_2, c_2) \in \mathit{Const}^2$ such that $E(a, b_i)$ and $F(b_i, c_i)$ hold for each $i \in \{1, 2\}$, and so on. Note that the certain answers to the query q from Example 4.11 on $\mathcal{T}$ are nonempty, as desired.

Now let M be an arbitrary schema mapping, let S be a source instance for M, and let q be a query over M's target schema. The preceding two examples suggest that it might be a good idea to answer q on M and S by $\mathit{cert}(q, \mathcal{T})$, where $\mathcal{T}$ is the set of all ground solutions for S under M that are unions of $\subseteq$-minimal solutions for S under M. For the moment, let us call such solutions *GCWA*-solutions*:

5.1 Definition (GCWA*-solutions, preliminary version)
Let M be a schema mapping, and let S be a source instance for M. A *GCWA*-solution* for S under M is a ground solution for S under M that is the union of one or more $\subseteq$-minimal solutions for S under M.

Note that logically equivalent schema mappings have the same GCWA*-solutions for a given source instance. In particular, the certain answers on GCWA*-solutions respect logical equivalence of schema mappings.

Consider a schema mapping $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$, where Σ is a set of st-tgds, and a source instance S for M. Intuitively, each ground $\subseteq$-minimal solution for S under M corresponds to a possible interpretation of the existentially quantified variables of the st-tgds in Σ by concrete constants. For example, if Σ contains the st-tgd

$$\chi \;:=\; \forall x \forall y \Big(E(x, y) \rightarrow \exists z \exists z' (F(x, z) \wedge F(y, z')) \Big),$$

where $E \in \sigma_{\mathrm{s}}$ and $F \in \sigma_{\mathrm{t}}$, and if $E^S = \{(a, b)\}$, then for each ground $\subseteq$-minimal solution T for S under M, there are constants $c, c' \in \mathit{Const}$ such that $F^T = \{(a, c), (b, c')\}$. In particular, we can view the set of all pairs (c, c') of constants associated with some T as the set of all possible interpretations of the variables z and z' of χ. Note that M and S intuitively tell us that one of these interpretations is possible, or two of these interpretations are possible, or three of these interpretations are possible, and so on. This corresponds to the union of one or more ground $\subseteq$-minimal solution for S under M. Thus, the set of all instances that are unions of one or more ground $\subseteq$-minimal solutions for S under M intuitively reflects the explicit and the implicit information contained in M and S as well as the standard semantics of FO quantifiers.

In the following we argue for a more general class of schema mappings, including schema mappings defined by st-tgds, that with respect to such schema mappings M, the set of all GCWA*-solutions for a source instance S under M intuitively reflects the explicit and the implicit information in M and S in such a way that the standard semantics of FO quantifiers is respected.

Namely, we consider schema mappings defined by a certain kind of $L_{\infty\omega}$ sentences. As indicated in Section 1.1.2, $L_{\infty\omega}$ logic extends FO logic by allowing disjunctions and conjunctions to range over infinitely many formulas. $L_{\infty\omega}$ formulas over a schema σ are defined like FO formulas over σ, except that we add *infinitary disjunctions* $\bigvee \Phi$ and *infinitary conjunctions* $\bigwedge \Phi$, where Φ is an arbitrary set of $L_{\infty\omega}$ formulas. The set of the free variables of $\bigvee \Phi$ or $\bigwedge \Phi$ is the set of all variables that occur free in some formula of Φ. The notions of *assignment* and *sentence* easily carry over to $L_{\infty\omega}$ formulas. The semantics of $L_{\infty\omega}$ formulas is a straightforward extension of the semantics of FO formulas. In particular, given an instance I and an assignment α for $\psi := \bigvee \Phi$, we have $I \models \psi(\alpha)$ if and only if $I \models \varphi(\alpha)$ for some $\varphi \in \Phi$. Furthermore, given an instance I and an assignment α for $\psi := \bigwedge \Phi$, we have $I \models \psi(\alpha)$ if and only if $I \models \varphi(\alpha)$ for all $\varphi \in \Phi$.

Let $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$ be a schema mapping, where Σ consists of *right-monotonic $L_{\infty\omega}$-st-tgds*, which are $L_{\infty\omega}$ sentences of the form

$$\chi \;:=\; \forall \bar{x}(\varphi(\bar{x}) \rightarrow \psi(\bar{x})),$$

where φ is a $L_{\infty\omega}$ formula over σ_{s}, and ψ is a *monotonic* $L_{\infty\omega}$ formula over σ_{t}. We assume that for all instances S over σ_{s}, and all instances T over σ_{t}, we have $S \cup T \models \chi$ if and only if for all $\bar{u} \in (\mathrm{dom}(S) \cup \mathrm{dom}(\chi))^{|\bar{x}|}$, where $\mathrm{dom}(\chi)$ is the set of all constants that occur in χ, $S \models \varphi(\bar{u})$ implies $T \models \psi(\bar{u})$. This can be enforced, for example, by relativizing the universal quantifiers, and the quantifiers in φ to the active domain over σ_{s}, and by relativizing the quantifiers in ψ to the active domain over σ_{t}. Note that right-monotonic $L_{\infty\omega}$-st-tgds in particular capture st-tgds. Now, given a source instance S for M, let

$$\begin{aligned}\Psi_{M,S} \;:=\; \{\psi(\bar{u}) \mid\ & \text{there exists } \forall \bar{x}(\varphi(\bar{x}) \rightarrow \psi(\bar{x})) \in \Sigma \\ & \text{and } \bar{u} \in \mathit{Const}^{|\bar{x}|} \text{ with } S \models \varphi(\bar{u})\}.\end{aligned}$$

Then for each ground target instance T for M, it holds that T is a solution for S under M if and only if T satisfies all sentences in $\Psi_{M,S}$. Since all sentences in $\Psi_{M,S}$ are monotonic, $\Psi_{M,S}$ is logically equivalent (on the set of all ground instances over σ_{t}) to the sentence

$$\psi_{M,S} \;:=\; \bigvee_{T_0 \in \mathcal{T}_0} \bigwedge_{R(\bar{t}) \in T_0} R(\bar{t}),$$

where $\mathcal{T}_0$ is the set of all $\subseteq$-minimal ground solutions for S under M (i.e., for all ground instances T over σ_{t}, we have $T \models \psi_{M,S}$ if and only if T satisfies all sentences in $\Psi_{M,S}$). Note that $\psi_{M,S}$ tells us that the target contains one ground $\subseteq$-minimal solution for S under M, or the target contains two distinct ground $\subseteq$-minimal solutions for S under M, and so on. So, intuitively, the information contained in $\psi_{M,S}$ (and thus in M and S) is reflected by the set of all instances T over σ_{t} that are the union of one or more instances in $\mathcal{T}_0$. This set corresponds to the set of all GCWA*-solutions for S under M.

The following example shows that the certain answers on GCWA*-solutions may be appropriate beyond schema mappings defined by right-monotonic $L_{\infty\omega}$-st-tgds.

5.2 Example
Consider once again the schema mapping M, the source instance S for M, and the query q from Example 4.12. For each ground target instance T for M that is the union of $\subseteq$-minimal solutions for S under M, there exists a nonempty finite set $C \subseteq \mathit{Const}$ with $E^T = \{(a, b) \mid b \in C\}$. Due to the constraint χ in M, T is a GCWA*-solution for S under M if and only if $2 \leq |C| \leq 3$. Thus, the set of all GCWA*-solutions for S under M intuitively reflects the explicit and the implicit information in M and S. Note that the certain answers to q on the set of all GCWA*-solutions for S under M are empty, as desired. □

Let $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$ be a schema mapping, where Σ does *not* entirely consist of right-monotonic $L_{\infty\omega}$-st-tgds, and let S be a source instance. Then the set of the GCWA*-solutions for S under M may suppress information that should intuitively be taken into account when answering queries:

5.3 Example
Consider the schema mapping $M = (\{P\}, \{E, F\}, \{\chi_1, \chi_2\})$, where

$$\chi_1 \;:=\; \forall x\Big(P(x) \to \exists z \exists z'\, E(z, z')\Big),$$
$$\chi_2 \;:=\; \forall x \forall y\Big(E(x, y) \wedge E(x', y) \to F(x, x')\Big),$$

and let S be the source instance for M with $P^S = \{a\}$. Furthermore, let c_1, c_2, c be constants with $c_1 \neq c_2$. Then the target instance T for M with $E^T = \{(c_1, c), (c_2, c)\}$ and $F^T = \{(c_i, c_j) \mid 1 \leq i, j \leq 2\}$ is a solution for S under M, but no GCWA*-solution for S under M. Nevertheless, it seems to be reasonable to take into account T when answering queries. The reason is that M and S intuitively tell us that it is possible that E contains both (c_1, c) and (c_2, c). So we should consider also all information implied by M and S for the case that E contains (c_1, c) and (c_2, c): namely, that F contains (c_i, c_j) for each $i, j \in \{1, 2\}$. Thus, for query answering, we should take into account not only

ground solutions that are unions of $\subseteq$-minimal solutions for S under M, but also all ground solutions that are unions of solutions in $\mathcal{T}$, where $\mathcal{T}$ consists of

- all $\subseteq$-minimal solutions for S under M, and
- all solutions that are $\subseteq$-minimal in the set of all solutions T_0 for S under M with $\{(c_1', c'), (c_2', c')\} \subseteq E^{T_0}$, for all $c_1', c_2', c' \in \mathit{Const}$ with $c_1' \neq c_2'$.

Note that the instance T considered above is the unique $\subseteq$-minimal solution among all solutions T_0 for S under M with $\{(c_1, c), (c_2, c)\} \subseteq E^{T_0}$. □

To resolve this issue, we start with the set

$$\mathcal{T}^0_{M,S} \;:=\; \{T \mid T \text{ is a ground } \subseteq\text{-minimal solution for } S \text{ under } M\},$$

and inductively add to this set solutions for S under M that are in some sense minimal consequences of certain facts mentioned by M and S. For query answering, we then take into account all solutions that are unions of one or more solutions in the resulting set.

For each set $\mathcal{I}$ of instances, let

$$\langle \mathcal{I} \rangle \;:=\; \{\bigcup \mathcal{I}' \mid \mathcal{I}' \text{ is a nonempty finite subset of } \mathcal{I}\},$$

where for a set X, $\bigcup X$ is an abbreviation for $\bigcup_{x \in X} x$. Furthermore, for each integer $i \geq 0$, let

$$\begin{aligned}\mathcal{T}^{i+1}_{M,S} \;:= \mathcal{T}^i_{M,S} \cup \{T \mid\; & T \notin \langle T^i_{M,S} \rangle, \text{ and there is some } T_0 \in \langle \mathcal{T}^i_{M,S} \rangle \text{ such that} \\ & T \text{ is } \subseteq\text{-minimal among all ground solutions } T' \\ & \text{for } S \text{ under } M \text{ with } T_0 \subseteq T'\}.\end{aligned}$$

In some sense, each instance $T \in \mathcal{T}^{i+1}_{M,S} \setminus \mathcal{T}^i_{M,S}$ is a minimal consequence of some fact that is mentioned by M and S. For instance, in Example 5.3, the solution T for S under M belongs to $\mathcal{T}^1_{M,S} \setminus \mathcal{T}^0_{M,S}$, and is a minimal consequence of the fact that it is possible that the relation E contains both (c_1, c) and (c_2, c).

Note that if Σ contains only st-tgds, we have $\mathcal{T}^0_{M,S} = \mathcal{T}^i_{M,S}$ for all $i \geq 0$, and $\langle \mathcal{T}^0_{M,S} \rangle$ is precisely the set of all GCWA*-solutions for S under M (cf., Proposition 5.5). In Example 5.3, it is intuitively clear that the set of all solutions for S under M that are contained in $\langle \mathcal{T}^1_{M,S} \rangle$ reflects the explicit and the implicit information in M and S in such a way that the standard semantics of FO quantifiers is reflected; moreover, it is not hard to see that $\langle \mathcal{T}^1_{M,S} \rangle = \langle \mathcal{T}^i_{M,S} \rangle$ for all $i \geq 1$. In general, we take into account all solutions for S under M that are unions of one or more instances in

$$\mathcal{T}^*_{M,S} \;:=\; \bigcup_{i \geq 0} \mathcal{T}^i_{M,S}.$$

We are ready to give the final definition of GCWA*-solutions and GCWA*-answers.

5.4 Definition (GCWA*-solution, GCWA*-answers, final version)
Let M be a schema mapping, let S be a source instance for M, and let q be a query over M's target schema.

- A *GCWA*-solution* for S under M is a solution for S under M that is a union of one or more instances in $\mathcal{T}^*_{M,S}$. The set of all GCWA*-solutions for S under M is denoted by $sol_{\text{GCWA}^*}(M, S)$.
- The *GCWA*-answers to q on M and S* are defined as

$$cert_{\text{GCWA}^*}(q, M, S) \;:=\; cert(q, sol_{\text{GCWA}^*}(M, S)).$$

The following proposition shows that for schema mappings defined by st-tgds and egds, GCWA*-solutions in the sense of 5.4, and GCWA*-solutions in the sense of 5.1, coincide.

5.5 Proposition.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ is a set of st-tgds and egds, and let S be a source instance for M. Then a target instance T for M is a GCWA-solution for S under M if and only if T is the union of one or more ground $\subseteq$-minimal solutions for S under M, and T satisfies the egds in Σ.*

Proof. It suffices to show that $\mathcal{T}^1_{M,S} = \mathcal{T}^0_{M,S}$. From the definition, it is clear that $\mathcal{T}^0_{M,S} \subseteq \mathcal{T}^1_{M,S}$. So let $T \in \mathcal{T}^1_{M,S}$. Then,

1. $T \in \mathcal{T}^0_{M,S}$, or
2. $T \notin \langle T^0_{M,S} \rangle$, and there is some $T_0 \in \langle \mathcal{T}^0_{M,S} \rangle$ such that T is $\subseteq$-minimal among all ground solutions T' for S under M with $T_0 \subseteq T'$.

Note that the second case is impossible due to the following. First note that T_0 is no solution for S under M, since otherwise, $T = T_0 \in \langle T^0_{M,S} \rangle$, which is not the case. Thus, T_0 does not satisfy the egds in Σ. In particular, since $T_0 \subseteq T$, T does not satisfy the egds in Σ, and therefore, $T \notin \mathcal{T}^1_{M,S}$. It follows that $T \in \mathcal{T}^0_{M,S}$, as desired. □

Moreover, an immediate consequence of Definition 5.4 is that the GCWA*-answers semantics respects logical equivalence of schema mappings:

5.6 Proposition (Preservation under logical equivalence).
Let $M_1 = (\sigma_s, \sigma_t, \Sigma_1)$ and $M_2 = (\sigma_s, \sigma_t, \Sigma_2)$ be logically equivalent schema mappings. Then for each instance S over σ_s, and each query q over σ_t, we have $cert_{GCWA^*}(q, M_1, S) = cert_{GCWA^*}(q, M_2, S)$.

Proof. Let S be an instance over σ_s. Since M_1 and M_2 are logically equivalent, we have $sol(M_1, S) = sol(M_2, S)$. Thus, by induction on i we have $\mathcal{T}^i_{M_1,S} = \mathcal{T}^i_{M_2,S}$ for every $i \geq 0$, and therefore $\mathcal{T}^*_{M_1,S} = \mathcal{T}^*_{M_2,S}$. By Definition 5.4, this implies that $sol_{\text{GCWA}^*}(M_1, S) = sol_{\text{GCWA}^*}(M_2, S)$. Hence, for each query q over σ_t, we have $cert_{\text{GCWA}^*}(q, M_1, S) = cert_{\text{GCWA}^*}(q, M_2, S)$. □

5.2 A Characterization of GCWA*-Solutions

This section presents a characterization of GCWA*-solutions for schema mappings defined by st-tgds and egds in the spirit of the definition of GCWA-solutions (Definition 4.8).

5.7 Definition ($D^*_{M,S}$)
For every schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$ and every source instance S for M, we define the following set of $L_{\infty\omega}$ sentences over $\sigma_s \cup \sigma_t$:

$$D^*_{M,S} := \{R(\bar{t}) \to \varphi \mid R \in \sigma_s \cup \sigma_t,\ \bar{t} \in Const^{\text{ar}(R)},\ \text{and } \varphi \text{ is a monotonic } L_{\infty\omega} \text{ sentence over } \sigma_s \cup \sigma_t \text{ that is satisfied in every } \subseteq\text{-minimal model } I \text{ of } D_{M,S} \text{ with } \bar{t} \in R^I\}.$$

5.8 Proposition (Characterization of GCWA*-solutions).
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ is a set of st-tgds and egds, and let S be a source instance for M. Then for all ground target instances T for M, the following statements are equivalent:

1. *T is a GCWA*-solution for S under M.*
2. *$S \cup T$ is a model of $D_{M,S} \cup D^*_{M,S}$.*

Proof.

1 $\Longrightarrow$ 2: Suppose that T is a GCWA*-solution for S under M. By Proposition 5.5, T is a ground solution for S under M, and there is a set $\mathcal{T}_0$ of $\subseteq$-minimal solutions for S under M such that

$$T = \bigcup T_0.$$

We have to show that

$$I := S \cup T$$

satisfies $D_{M,S} \cup D^*_{M,S}$. Since T is a solution for S under M, we already have $I \models D_{M,S}$. Thus, it remains to show that $I \models D^*_{M,S}$.

To this end, consider an arbitrary sentence

$$\psi \;:=\; R(\bar{t}) \to \varphi$$

in $D^*_{M,S}$, and assume that

$$I \;\models\; R(\bar{t}).$$

Since $I = S \cup T$ and $T = \bigcup T_0$, there is some $T_0 \in \mathcal{T}_0$ with $\bar{t} \in R^{S \cup T_0}$. Note that $I_0 := S \cup T_0$ is a $\subseteq$-minimal model of $D_{M,S}$. By Definition 5.7, we thus have $I_0 \models \varphi$. Since $I_0 \subseteq I$ and φ is monotonic, it follows that $I \models \varphi$. Consequently, I satisfies ψ. This shows that $I \models D^*_{M,S}$.

2 $\Longrightarrow$ 1: Suppose that

$$I \;:=\; S \cup T$$

is a model of $D_{M,S} \cup D^*_{M,S}$. Since models are ground instances by definition, it follows that T is a ground solution for S under M. To show that T is a GCWA*-solution for S under M, it remains to construct, by Proposition 5.5, a set $\mathcal{T}_0$ of $\subseteq$-minimal solutions for S under M such that $T = \bigcup \mathcal{T}_0$.

Let $\mathcal{T}_0$ be the set of all $\subseteq$-minimal solutions T_0 for S under M with $T_0 \subseteq T$. We claim that $T = \bigcup \mathcal{T}_0$. By construction, we have $\bigcup \mathcal{T}_0 \subseteq T$. Thus it remains to show that it is not the case that $\bigcup \mathcal{T}_0 \subsetneq T$.

Suppose, to the contrary, that $\bigcup \mathcal{T}_0 \subsetneq T$. Then there is a relation symbol $R \in \sigma_{\mathrm{t}}$ and a tuple $\bar{t} \in \mathit{Const}^{\mathrm{ar}(R)}$ such that

$$\bar{t} \in R^T \quad \text{and} \quad \bar{t} \notin R^{T_0} \text{ for all } T_0 \in \mathcal{T}_0. \tag{5.1}$$

On the other hand, there is at least one $\subseteq$-minimal model I_0 of $D_{M,S}$ with $\bar{t} \in R^{I_0}$. Otherwise, $R(\bar{t}) \to \bigvee \emptyset$, which is equivalent to $\neg R(\bar{t})$, would be in $D^*_{M,S}$, so that $\bar{t} \notin R^I \supseteq R^T$ would contradict Eq. (5.1). Let

$$\mathcal{I}_0 \;:=\; \{I_0 \mid I_0 \text{ is a } \subseteq\text{-minimal model } I_0 \text{ of } D_{M,S} \text{ with } \bar{t} \in R^{I_0}\}.$$

Then,

$$\psi \;:=\; R(\bar{t}) \to \varphi \qquad \text{with} \qquad \varphi \;:=\; \bigvee_{I_0 \in \mathcal{I}_0} \bigwedge_{R'(\bar{t}') \in I_0} R'(\bar{t}')$$

is satisfied in *every* $\subseteq$-minimal model of $D_{M,S}$. Since φ is monotonic, we thus have $\psi \in D^*_{M,S}$. Furthermore, since $I \models D^*_{M,S}$ and $I \models R(\bar{t})$, it follows that $I \models \varphi$. In particular, there must be some $I_0 \in \mathcal{I}_0$ such that $I \models \bigwedge_{R'(\bar{t}') \in I_0} R'(\bar{t}')$, and thus, $I_0 \subseteq I$. Note that $I_0 = S \cup T_0$ for some $T_0 \in \mathcal{T}_0$. Together with $\bar{t} \in R^{I_0}$ and $R \in \sigma_{\mathrm{t}}$, this implies that $\bar{t} \in R^{T_0}$. However, this contradicts Eq. (5.1). Consequently, $\bigcup \mathcal{T}_0 = T$. □

Let $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$ be a schema mapping defined by st-tgds and egds, and let S be a source instance for M. Note that each sentence in $\overline{\overline{D_{M,S}}}$ is logically equivalent to a sentence in $D^*_{M,S}$ (since $\neg R(\bar{t})$ is equivalent to $R(\bar{t}) \rightarrow \bigvee \emptyset$). An immediate consequence of Proposition 5.8 is therefore that every GCWA*-solution for S under M is a GCWA-solution for S under M. Furthermore, it follows immediately from the definitions that every EGCWA-solution for S under M is a GCWA*-solution for S under M.

The following result translates Theorem 5 in Minker [1982] from GCWA-solutions to GCWA*-solutions. It shows that for a given schema mapping M and a source instance S for M, the set $D_{M,S} \cup D^*_{M,S}$ is maximally consistent in the sense that the addition of any sentence of the form $R(\bar{t}) \rightarrow \varphi$, where φ is a monotonic $L_{\infty\omega}$ sentence, leads to a set of sentences that is inconsistent with $D_{M,S} \cup D^*_{M,S}$.

5.9 Proposition.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping defined by st-tgds and egds, let S be a nonempty source instance for M, and let $D'_{M,S} := D_{M,S} \cup D^*_{M,S}$.

1. *For all monotonic $L_{\infty\omega}$ sentences φ over $\sigma_s \cup \sigma_t$, we have $D_{M,S} \models \varphi$ if and only if $D'_{M,S} \models \varphi$.*

2. *Let $R \in \sigma_s \cup \sigma_t$, let $\bar{t} \in Const^{\mathrm{ar}(R)}$, let φ be a monotonic $L_{\infty\omega}$ sentence over $\sigma_s \cup \sigma_t$, and let $\psi := R(\bar{t}) \rightarrow \varphi$ with $D'_{M,S} \not\models \psi$. Then,*

   (a) *$D'_{M,S} \cup \{\psi\}$ has no model, or*

   (b) *there is a monotonic $L_{\infty\omega}$ sentence χ over $\sigma_s \cup \sigma_t$ such that $D'_{M,S} \cup \{\psi\} \models \chi$, but $D'_{M,S} \not\models \chi$.*

Proof.

Ad 1: Let φ be a monotonic $L_{\infty\omega}$ sentence over $\sigma_{\mathrm{s}} \cup \sigma_{\mathrm{t}}$. Clearly, $D_{M,S} \models \varphi$ implies $D'_{M,S} \models \varphi$. It remains to show that $D'_{M,S} \models \varphi$ implies $D_{M,S} \models \varphi$.

Suppose that $D'_{M,S} \models \varphi$. Then every $\subseteq$-minimal model of $D'_{M,S}$ satisfies φ. Since every $\subseteq$-minimal model of $D'_{M,S}$ is a $\subseteq$-minimal model of $D_{M,S}$, it follows that φ is true in every $\subseteq$-minimal model of $D_{M,S}$. Since φ is monotonic, this implies that φ is true in every model of $D_{M,S}$. Consequently, $D_{M,S} \models \varphi$.

Ad 2: If $D'_{M,S} \cup \{\psi\}$ has no model, we are done. Suppose that $D'_{M,S} \cup \{\psi\}$ has a model. Let $\mathcal{I}_0$ be the set of all $\subseteq$-minimal models of $D'_{M,S} \cup \{\psi\}$, and consider the monotonic $L_{\infty\omega}$ sentence

$$\chi \;:=\; \bigvee_{I_0 \in \mathcal{I}_0} \bigwedge_{R'(\bar{t}') \in I_0} R'(\bar{t}').$$

We claim that $D'_{M,S} \cup \{\psi\} \models \chi$, and that $D'_{M,S} \not\models \chi$.

$D'_{M,S} \cup \{\psi\} \models \chi$: Let I be a model of $D'_{M,S} \cup \{\psi\}$, and let $I_0 \in \mathcal{I}_0$ be such that $I_0 \subseteq I$. Then, $I_0 \models \bigwedge_{R'(\bar{t}') \in I_0} R'(\bar{t}')$, and therefore, $I_0 \models \chi$. Since χ is monotonic and $I_0 \subseteq I$, this leads to $I \models \chi$.

$D'_{M,S} \not\models \chi$: We first observe:

$$\text{There is a } \subseteq\text{-minimal model } I_0 \text{ of } D_{M,S} \text{ with } I_0 \not\models \psi. \tag{5.2}$$

Indeed, if $\psi = R(\bar{t}) \to \varphi$, this follows immediately from $\psi \notin D^*_{M,S}$, which in turn follows from $D^*_{M,S} \subseteq D'_{M,S}$ and $D'_{M,S} \not\models \psi$. If $\psi = \varphi$, pick some $R' \in \sigma_\mathrm{s}$ and some $\bar{t}' \in R^S$, which is possible since S is nonempty, and observe that $D'_{M,S} \not\models \psi'$, where $\psi' := R'(\bar{t}') \to \varphi$. This is because for every model I' of $D'_{M,S}$ we have $\bar{t}' \in (R')^{I'}$, and $D'_{M,S} \not\models \psi = \varphi$. In particular, $\psi' \notin D^*_{M,S}$, and the claim follows.

Note that every $\subseteq$-minimal model of $D_{M,S}$ is a $\subseteq$-minimal model of $D'_{M,S}$, so that by Eq. (5.2), there is a $\subseteq$-minimal model I_0 of $D'_{M,S}$ with

$$I_0 \not\models \psi. \tag{5.3}$$

Let us fix such an instance I_0.

To finish the proof, suppose, for a contradiction, that $D'_{M,S} \models \chi$. Since I_0 is a model of $D'_{M,S}$, we then have $I_0 \models \chi$. Thus, there is an instance $I'_0 \in \mathcal{I}_0$ with $I_0 \models \bigwedge_{R'(\bar{t}') \in I'_0} R'(\bar{t}')$, that is, $I'_0 \subseteq I_0$. Since I_0 and I'_0 are $\subseteq$-minimal models of $D'_{M,S}$, this implies $I'_0 = I_0$. However, since $I'_0 \in \mathcal{I}_0$, we have $I'_0 \models \psi$, and, in particular,

$$I_0 \models \psi.$$

This clearly contradicts Eq. (5.3). Consequently, $D'_{M,S} \not\models \chi$. □

5.3 The Complexity of Computing GCWA*-Answers

We now turn to the problem of computing GCWA*-answers. As in Section 3.4, we concentrate on the *data complexity*, and given a schema mapping $M = (\sigma_\mathrm{s}, \sigma_\mathrm{t}, \Sigma)$ and a query language L, we seek for algorithms $\mathbb{A}_1, \mathbb{A}_2$ such that

- $\mathbb{A}_1$ takes a source instance S for M as input and computes a solution T for S under M,
- $\mathbb{A}_2$ takes a solution T computed by $\mathbb{A}_1$ and a query $q \in L$ over σ_t as input and computes the GCWA*-answers to q on M and S, and
- both $\mathbb{A}_1$, and $\mathbb{A}_2$ for fixed $q \in L$, run in polynomial time.

In Section 5.3.1, we first deal with monotonic queries and show that on such queries, the GCWA*-answers semantics coincides with the certain answers semantics (Proposition 5.10). Hence, if L is the class of all homomorphism-preserved queries with polynomial time data complexity, and M is a weakly acyclic schema mapping, say, then polynomial time algorithms $\mathbb{A}_1$ and $\mathbb{A}_2$ as above exist: $\mathbb{A}_1$ just has to compute an arbitrary universal solution T for S under M, while $\mathbb{A}_2$ evaluates a query $q \in L$ on T and removes all tuples with nulls from the answer (see Chapter 2).

Section 5.3.2 then considers the case of existential queries, and extensions thereof. For such queries, polynomial time algorithms $\mathbb{A}_1$ and $\mathbb{A}_2$ as above may not exist. As in Section 3.4, we show this by considering, for given schema mappings $M = (\sigma_s, \sigma_t, \Sigma)$, and queries q over σ_t, the problem

$\textsc{Eval}(M, q)$	
Input:	a source instance S for M, and a tuple $\bar{t} \in Dom^{\mathrm{ar}(q)}$
Question:	Is $\bar{t} \in cert_{\mathrm{GCWA}^*}(q, M, S)$?

Recall that the complexity of this problem is a lower bound on the joint complexity of algorithms $\mathbb{A}_1$ and $\mathbb{A}_2$ as above: if, for example, $\textsc{Eval}(M, q)$ is co-NP-complete, then finding T is intractable, or computing $cert_{\mathrm{GCWA}^*}(q, M, S)$ from T is intractable.

5.3.1 Monotonic Queries

The purpose of this short section is to point out that for *monotonic* queries, the GCWA*-answers semantics coincides with the certain answers semantics. This can be seen as further evidence that—as indicated in Section 1.3.1—the certain answers semantics is the "right" semantics for answering monotonic queries. Moreover, all results presented in Chapter 2 carry over to the GCWA*-answers semantics.

5.10 Proposition.
Let M be a schema mapping, let S be a source instance for M, and let q be a monotonic query over the target schema of M. Then, $cert_{GCWA^*}(q, M, S) = cert(q, M, S)$.

Proof. Since every GCWA*-solution for S under M is a solution for S under M, we have $cert(q, M, S) \subseteq cert_{\mathrm{GCWA}^*}(q, M, S)$. It therefore remains to prove $cert_{\mathrm{GCWA}^*}(q, M, S) \subseteq cert(q, M, S)$.

Let $\bar{t} \in cert_{\mathrm{GCWA}^*}(q, M, S)$. We show that $\bar{t} \in q(T)$ for all ground solutions T for S under M, which implies $\bar{t} \in cert(q, M, S)$. To this end, let T be a

ground solution for S under M. Pick an arbitrary $\subseteq$-minimal solution T_0 for S under M with $T_0 \subseteq T$. By Definition 5.4, T_0 is a GCWA*-solution for S under M. Since $\bar{t} \in \mathit{cert}_{\mathrm{GCWA}^*}(q, M, S)$, we thus have $\bar{t} \in q(T_0)$. Since q is monotonic, and since $T_0 \subseteq T$, it follows that $\bar{t} \in q(T)$. □

In the remaining part of this section, we deal with computing the GCWA*-answers to *non-monotonic* queries.

5.3.2 Existential Queries and Beyond

We now turn to *existential queries*, which are FO queries of the form $\exists \bar{x}\, \varphi$, where φ is a quantifier-free FO formula. A particular class of existential queries are *conjunctive queries with negation* ($\mathrm{CQ}^\neg$ queries, for short), which are queries of the form $\exists \bar{x}(L_1 \wedge \cdots \wedge L_k)$ where each L_i is a relational atomic FO formula, or the negation of a relational atomic FO formula. A simple reduction from the Clique problem (see, e.g., Garey and Johnson [1979]) shows that $\textsc{Eval}(M, q)$ is already co-NP-hard for schema mappings M defined by *LAV tgds* and $\mathrm{CQ}^\neg$ queries q with only one negated atom. Here, a *LAV tgd* is a st-tgd of the form $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$, where φ contains only one atom.

5.11 Proposition.
There is a schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$, where Σ consists of two LAV tgds, and a Boolean $CQ^\neg$ query q over σ_t with one negated atomic FO formula such that $\textsc{Eval}(M, q)$ *is* co-NP*-complete.*

Proof. Let $M = (\sigma_\mathrm{s}, \sigma_\mathrm{t}, \Sigma)$, where σ_s consists of binary relation symbols E_0, C_0, σ_t consists of binary relation symbols E, C, A, and Σ consists of the following LAV tgds:

$$\begin{aligned} \chi_1 &:= \forall x \forall y \Big(E_0(x, y) \rightarrow E(x, y) \Big), \\ \chi_2 &:= \forall x \forall y \Big(C_0(x, y) \rightarrow \exists z_1 \exists z_2 (C(x, y) \wedge A(x, z_1) \wedge A(y, z_2)) \Big). \end{aligned}$$

Furthermore, let

$$q := \exists x \exists y \exists z_1 \exists z_2 \Big(C(x, y) \wedge A(x, z_1) \wedge A(y, z_2) \wedge \neg E(z_1, z_2) \Big).$$

We show that $\textsc{Eval}(M, q)$ is co-NP-complete by showing that the complement of $\textsc{Eval}(M, q)$ is NP-complete.

The complement of $\textsc{Eval}(M, q)$ is solved by a nondeterministic Turing machine as follows. Given a source instance S for M, the machine implicitly guesses a GCWA*-solution T for S under M, and accepts S if and only if $T \not\models q$. More precisely, it guesses values $v_c \in \mathit{Dom}$ for each $c \in C_0^S$, and accepts S if and

only if $(v_c, v_{c'}) \in E_0^S$ for all $c, c' \in C_0^S$. This is possible by the requirement that $(v_c, v_{c'}) \in E_0^S$ for all $c, c' \in C_0^S$, which implies that the values v_c can be chosen from $\mathrm{dom}(S)$. Clearly, the machine runs in time polynomial in the size of S.

To show that the complement of $\textsc{Eval}(M, q)$ is NP-hard, we present a reduction from the NP-complete CLIQUE problem (see, e.g., Garey and Johnson [1979]). The CLIQUE problem is to decide, given an undirected graph $G = (V, E)$ without loops and a positive integer k, whether G contains a *clique* of size k, where a clique in G is a set $C \subseteq V$ such that for all $u, v \in C$ with $u \neq v$ we have $\{u, v\} \in E$.

Let $G = (V, E)$ be an undirected graph without loops, and let $k \geq 2$ be an integer. If $k = 1$, then G has a clique of size k if and only if V is nonempty. That is, we can reduce (G, k) to some predefined fixed source instance S for M such that $\mathit{cert}_{\mathrm{GCWA}^*}(q, M, S) = \emptyset$ if and only if V is nonempty (e.g., a source instance S with $C_0^S = \emptyset$ if V is nonempty, and a source instance S with $C_0^S = \{(c, c)\}$ for some $c \in \mathit{Const}$ if V is empty). If $k \geq 2$, we construct the source instance S for M with $E_0^S = E$ and $C_0^S = \{(c_i, c_j) \mid 1 \leq i, j \leq k,\ i \neq j\}$, where $c_1, \ldots, c_k$ is a sequence of pairwise distinct constants that do not occur in V. We claim that G has a clique of size k if and only if $\mathit{cert}_{\mathrm{GCWA}^*}(q, M, S) = \emptyset$.

"Only if": Let $C = \{v_1, \ldots, v_k\}$ be a clique of size k in G, and let T be the target instance for M with $E^T = E$, $C^T = C_0^S$ and $A^T = \{(c_i, v_i) \mid 1 \leq i \leq k\}$. Then T is a $\subseteq$-minimal solution for S under M, and, by Proposition 5.5, a GCWA*-solution for S under M. Furthermore, we have $T \not\models q$. To see this, note that for all $u, v, w_1, w_2 \in \mathrm{dom}(T)$ with

$$T \models C(u, v) \wedge A(u, w_1) \wedge A(v, w_2),$$

there are distinct $i, j \in \{1, \ldots, k\}$ with $u = c_i$ and $v = c_j$, so that $w_1 = v_i$ and $w_2 = v_j$. Since $v_i, v_j \in C$ and $E^T = E$, we thus have $T \models E(w_1, w_2)$ for all such u, v, w_1, w_2. Since T is a GCWA*-solution for S under M, and $T \not\models q$, we have $\mathit{cert}_{\mathrm{GCWA}^*}(q, M, S) = \emptyset$.

"If": Suppose that $\mathit{cert}_{\mathrm{GCWA}^*}(q, M, S) = \emptyset$. Then there is a GCWA*-solution T for S under M with $T \not\models q$. For all $i \in \{1, \ldots, k\}$, let

$$V_i := \{v \in \mathrm{dom}(T) \mid (c_i, v) \in A^T\}.$$

Since $S \cup T \models \chi_2$, each V_i is nonempty. Thus, there is a set $C = \{v_1, \ldots, v_k\}$ such that $v_i \in V_i$ for each $i \in \{1, \ldots, k\}$. Moreover, for all $i, j \in \{1, \ldots, k\}$ with $i \neq j$, we have $(v_i, v_j) \in E$. To see this, observe that $T \models C(c_i, c_j) \wedge A(c_i, v_i) \wedge A(c_j, v_j)$, so that $T \not\models q$ implies $T \models E(v_i, v_j)$. It follows that C is a clique in G of size k (since $k \geq 2$ and G has no loops). □

Adding only one universal quantifier may make the problem undecidable. Specifically, as shown by Proposition 5.12 below, there is a simple schema mapping M and a $\exists^*\forall$ FO query q over M's target schema such that $\textsc{Eval}(M, q)$ is undecidable. Here, a $\exists^*\forall$ FO query is a FO query of the form

$$\exists x_1 \cdots \exists x_k \forall y \, \varphi(x_1, \ldots, x_k, y, \bar{z}),$$

where φ is quantifier-free.

5.12 Proposition.
There is a schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$, where Σ consists of two LAV tgds, and a Boolean $\exists^*\forall$ FO query q over σ_t such that $\textsc{Eval}(M, q)$ is undecidable.

Proof. Similar to Proposition 3.41, we will construct M and q in such a way that the (undecidable) embedding problem for finite semigroups (see Example 1.14) can be reduced to $\textsc{Eval}(M, q)$. Let $M = (\{R\}, \{R_p, R_f\}, \Sigma)$, where R, R_p, R_f are ternary relation symbols, and Σ consists of the st-tgds

$$\begin{aligned}
\chi_1 &:= \forall x_1 \forall x_2 \forall x_3 \Big(R(x_1, x_2, x_3) \to R_p(x_1, x_2, x_3) \Big), \\
\chi_2 &:= \forall x_1 \forall x_2 \forall x_3 \Big(R_p(x_1, x_2, x_3) \to \exists y_1 \exists y_2 \exists y_3 \, R_f(y_1, y_2, y_3) \Big).
\end{aligned}$$

Let $q' := \neg\varphi$, where φ is the FO formula from Example 1.14 with R replaced by R_p, and $\tilde{R}$ replaced by R_f. That is,

$$\begin{aligned}
\neg q' \;\equiv\; & \forall x \forall y \forall z \Big(R_p(x, y, z) \to R_f(x, y, z) \Big) \\
& \wedge \forall x \forall y \forall z \forall z' \Big(R_f(x, y, z) \wedge R_f(x, y, z') \to z = z' \Big) \\
& \wedge \forall x \forall y \Big(\psi_{R_f}(x) \wedge \psi_{R_f}(y) \to \exists z R_f(x, y, z) \Big) \\
& \wedge \forall x \forall y \forall z \forall u \forall v \forall w (R_f(x, y, u) \wedge R_f(u, z, v) \wedge R_f(y, z, w) \to R_f(x, w, v)),
\end{aligned}$$

where

$$\psi_{R_f}(x) \;:=\; \exists x_1 \exists x_2 \exists x_3 \Big(R_f(x_1, x_2, x_3) \wedge \bigvee_{i=1}^{3} x = x_i \Big)$$

defines the set of values contained in R_f. Note that q' is equivalent to a Boolean $\exists^*\forall$ FO query q. Furthermore, $\neg q$ is true in a target instance T for M precisely if $R_p^T \subseteq R_f^T$, and R_f^T encodes the graph of a total associative function $f\colon Y^2 \to Y$ for some finite set Y.

To show that $\textsc{Eval}(M, q)$ is undecidable, we reduce the embedding problem for finite semigroups to $\textsc{Eval}(M, q)$. Let $p\colon X^2 \to X$ be a partial function, where X is a finite set. Construct the source instance S_p for M as in

Example 1.14. We claim that $cert_{\text{GCWA}^*}(q, M, S_p) = \emptyset$ if and only if p is a "yes"-instance of the embedding problem for finite semigroups.

By Proposition 5.5, the GCWA*-solutions for S under M are all the target instances T for M such that $R_p^T = R^S$, and either

1. $R^S = R_p^T = R_f^T = \emptyset$, or
2. R_f^T is a nonempty finite subset of $Const^3$.

Therefore, $cert_{\text{GCWA}^*}(q, M, S) = \emptyset$ if and only if there is a GCWA*-solution T for S under M such that R_f^T is the graph of a total function $f\colon \text{dom}(T)^2 \to \text{dom}(T)$ that is associative and extends p. This is the case precisely if p is a "yes"-instance of the embedding problem for finite semigroups. □

5.4 Computing GCWA*-Answers to Universal Queries

This section presents the technically most challenging result of this thesis: Theorem 5.15, which states that there is a polynomial time algorithm for computing the GCWA*-answers to universal queries under schema mappings defined by certain st-tgds from the core of the universal solutions.

A *universal query* is a FO query of the form $q(\bar{x}) = \forall \bar{y}\, \varphi(\bar{x}, \bar{y})$, where φ is quantifier-free. Recall that we assume *domain-independence* of queries. Note also that, in Example 1.25, if we consider the query

$$q'(x) \;:=\; \forall y\Big(R'(x, y) \to R'(y, x)\Big)$$

and the source instance S' for M with $E^{S'} = \{(a, a)\}$, then the certain answers to q' on M and S' are empty, contrary to the expectation under the CWA-based point of view that the answer to q' on M and S' is the answer to q' on the copy of S', namely $\{a\}$.

An upper bound on the complexity of computing the GCWA*-answers to universal queries under schema mappings defined by st-tgds is:

5.13 Proposition.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of st-tgds, and let q be a universal query over σ_t. Then Eval(M, q) *belongs to* co-NP.

A proof of Proposition 5.13 is given at the end of Chapter 5, in Section 5.4.5.

In what follows, we prove that under schema mappings defined by st-tgds that are *packed* in the sense as defined below, the GCWA*-answers to universal queries can be computed in polynomial time.

5.14 Definition (packed st-tgd)
A st-tgd $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\bigwedge_{i=1}^{n} R_i(\bar{u}_i))$ is *packed* if for all $i,j \in \{1,\ldots,n\}$ with $i \neq j$, there is a variable in $\bar{z}$ that occurs both in $\bar{u}_1$ and in $\bar{u}_2$.

Although not as expressive as schema mappings defined by st-tgds, schema mappings defined by packed st-tgds seem to form an interesting class of schema mappings. Note that packed st-tgds still allow for non-trivial use of existential quantifiers in the heads of st-tgds. For example, consider a schema mapping M defined by st-tgds $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$, where ψ contains at most two atoms with variables from $\bar{z}$. Then M is logically equivalent to a schema mapping defined by packed st-tgds, since every st-tgd θ in M is logically equivalent to a set Θ of packed st-tgds, where the size of Θ is at most the number of atomic formulas in the head of θ. An example of a st-tgd that is *not* packed is $\forall x(P(x) \rightarrow \exists z_1\exists z_2\exists z_3(E(x,z_1) \wedge E(z_1,z_2) \wedge E(z_2,z_3)))$.

We are now ready to state the main result of this section:

5.15 Theorem.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of packed st-tgds, and let q be a universal query over σ_t. Then there is a polynomial time algorithm that takes a target instance T for M as input, and outputs a relation

$$R \;\subseteq\; \Big(\mathrm{const}(T) \cup \mathrm{dom}(q)\Big)^{\mathrm{ar}(q)}$$

such that for all source instances S for M, we have

$$T \cong \mathrm{Core}(M,S) \quad\Longrightarrow\quad R = \mathit{cert}_{GCWA^*}(q,M,S).$$

Before we tackle the proof of Theorem 5.15, let us note the following. First, the algorithm from Theorem 5.15 runs in polynomial time for every given target instance for M, but can output an arbitrary $\mathrm{ar}(q)$-ary relation over $\mathrm{const}(T) \cup \mathrm{dom}(q)$ if T does not happen to be isomorphic to $\mathrm{Core}(M,S)$ for some source instance S for M. Second, Theorem 5.15 and Theorem 2.28 immediately imply:

5.16 Corollary.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of packed st-tgds, and let q be a universal query over σ_t. Then there is a polynomial time algorithm that takes a source instance S for M as input and outputs $\mathit{cert}_{GCWA^*}(q,M,S)$. In particular, $\textsc{Eval}(M,q)$ *is in* PTIME.

Furthermore, an interesting consequence of Theorem 5.15 arises in combination with the results of Section 2.1. Let M be a schema mapping defined by packed st-tgds, and let S be a source instance for M. The results of Section 2.1 and Theorem 5.15 tell us that the same solution for S under M, namely

Core(M, S), can be used to efficiently compute both the certain answers to homomorphism-preserved queries on M and S, and the GCWA*-answers to universal queries on M and S (if M and the query are not part of the input). If one considers to compute only the certain answers to homomorphism-preserved queries on M and S, and the GCWA*-answers to universal queries on M and S, it suffices therefore to compute Core(M, S), which, by Theorem 2.28, is possible in polynomial time if M is not part of the input.

Let us now turn to the proof of Theorem 5.15. Note that Theorem 5.15 easily follows from:

5.17 Theorem.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of packed st-tgds, and let q be a universal query over σ_t. Then there is a polynomial time algorithm that takes a target instance T for M and a tuple $\bar{t} \in Const^{\mathrm{ar}(q)}$ as input, and if $T \cong \mathrm{Core}(M, S)$ for some source instance S for M, then it decides whether $\bar{t} \in cert_{GCWA^*}(q, M, S)$.

Indeed, each tuple in $cert_{\mathrm{GCWA}^*}(q, M, S)$ belongs, in particular, to $q(T)$ for all target instances T for M with $T \cong \mathrm{Core}(M, S)$. Since q is domain-independent, this implies that $cert_{\mathrm{GCWA}^*}(q, M, S)$ contains only $\mathrm{ar}(q)$-tuples over $\mathrm{const}(\mathrm{Core}(M, S)) \cup \mathrm{dom}(q)$. Thus, given a target instance T for M with $T \cong \mathrm{Core}(M, S)$, we obtain $cert_{\mathrm{GCWA}^*}(q, M, S)$ by

- considering all $\mathrm{ar}(q)$-tuples $\bar{t}$ over $\mathrm{const}(T) \cup \mathrm{dom}(q)$ in turn, and
- determining all those tuples $\bar{t}$ with $\bar{t} \in cert_{\mathrm{GCWA}^*}(q, M, S)$ using the algorithm from Theorem 5.17.

By Theorem 5.17 and the fact that q is not part of the input, this yields a polynomial time algorithm for computing the GCWA*-answers to q on M and S. Note that if T does not happen to be isomorphic to Core(M, S) for some source instance S for M, this algorithm runs in polynomial time as well (since the algorithm from Theorem 5.17 runs in polynomial time), but may output an arbitrary $\mathrm{ar}(q)$-ary relation over $\mathrm{const}(T) \cup \mathrm{dom}(q)$. In particular, this proves Theorem 5.15.

The remaining part of this section is devoted to the proof of Theorem 5.17. Let us begin by showing how membership of a tuple in $cert_{\mathrm{GCWA}^*}(q, M, S)$ can be decided on the basis of an instance T with $T \cong \mathrm{Core}(M, S)$.

5.4.1 GCWA*-Answers and the Core of the Universal Solutions

Let $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$ be a schema mapping, where Σ is a set of packed st-tgds, and let q be a universal query over σ_{t}. Given an instance T with $T \cong \mathrm{Core}(M, S)$

for some source instance S for M, and a tuple $\bar{t} \in \mathit{Const}^{\mathrm{ar}(q)}$, how can we decide whether $\bar{t}$ belongs to $\mathit{cert}_{\mathrm{GCWA}^*}(q, M, S)$?

Note that $\bar{t}$ does *not* belong to $\mathit{cert}_{\mathrm{GCWA}^*}(q, M, S)$ if and only if there is a GCWA*-solution $\tilde{T}$ for S under M with $\tilde{T} \models \neg q(\bar{t})$. By Proposition 5.5 and the fact that Σ consists of st-tgds, the latter is the case precisely if there is a nonempty finite set $\mathcal{T}$ of ground $\subseteq$-minimal solutions for S under M with $\bigcup \mathcal{T} \models \neg q(\bar{t})$. We can now reformulate this in terms of $\mathrm{Core}(M, S)$ using the following lemma, which is based on the set $\mathit{poss}(\mathrm{Core}(M, S))$ of the possible worlds of $\mathrm{Core}(M, S)$ (cf., Section 1.3.1).

5.18 Lemma.
Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of st-tgds, and let S be a source instance for M. Then the set of all ground $\subseteq$-minimal solutions for S under M is precisely the set of all $\subseteq$-minimal instances in $\mathit{poss}(\mathrm{Core}(M, S))$.

Proof. Let $T := \mathrm{Core}(M, S)$. It suffices to show that each instance in $\mathit{poss}(T)$ is a solution for S under M, and that each ground $\subseteq$-minimal solution for S under M belongs to $\mathit{poss}(T)$.

Step 1: Each instance in $\mathit{poss}(T)$ is a solution for S under M.
Let f be a valuation of T. We claim that $T' := f(T)$ is a solution for S under M. To this end, let χ be a tgd in Σ of the form $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$, and let $\bar{u}, \bar{v}$ be tuples with $S \models \varphi(\bar{u}, \bar{v})$. We must show that there is a tuple $\bar{w}$ with $T' \models \psi(\bar{u}, \bar{w})$. Since T is a solution for S under M, there is a tuple $\bar{w}'$ with $T \models \psi(\bar{u}, \bar{w}')$. Thus, since f is a homomorphism from T to T', ψ is preserved under homomorphisms, and $f(\bar{u}) = \bar{u}$, we have $T' \models \psi(\bar{u}, \bar{w})$, where $\bar{w} := f(\bar{w}')$. It follows that T' is a solution for S under M.

Step 2: Each ground $\subseteq$-minimal solution for S under M belongs to $\mathit{poss}(T)$.
Let T_0 be a ground $\subseteq$-minimal solution for S under M. We show that T_0 belongs to $\mathit{poss}(T)$.

We first show that T_0 is a *CWA-presolution* for S under M. Indeed, for each justification $j = (\chi, \bar{u}, \bar{v}) \in \mathcal{J}_{M,S}$, where χ has the form $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$, let us pick a tuple $\bar{w}_j \in \mathit{Dom}^{|\bar{z}|}$ with $T_0 \models \psi(\bar{u}, \bar{w}_j)$. Note that such a tuple exists, since T_0 is a solution for S under M. Define $\zeta \colon \mathcal{J}^*_{M,S} \rightarrow \mathit{Dom}$ such that $\zeta(j) = \bar{w}_j$ for each $j \in \mathcal{J}_{M,S}$. Then the CWA-presolution $T_{M,S,\zeta}$ for S under M is a subinstance of T_0, which implies $T_{M,S,\zeta} = T_0$, because T_0 is a $\subseteq$-minimal solution for S under M.

By Proposition 3.6, there is a homomorphism h_0 from $T^* := \mathrm{CanSol}(M, S)$ to T_0 such that

$$h_0(T^*) \;=\; T_0. \tag{5.4}$$

In fact, h_0 is a valuation of T^*, since T_0 is a ground instance. Furthermore, since T^* is a universal solution for S under M, each core of T^* is isomorphic to T, which implies that there is an injective homomorphism $\iota\colon \mathrm{dom}(T) \to \mathrm{dom}(T^*)$ such that

$$\iota(T) \;\subseteq\; T^*. \tag{5.5}$$

Now consider the composition $f := h_0 \circ \iota$ of ι and h_0. Note that f is a valuation of T, since h_0 is a valuation of T. Thus,

$$T' \;:=\; f(T) \;\in\; \mathit{poss}(T), \tag{5.6}$$

and, as shown in step 1 above, T' is a solution for S under M. On the other hand, we have

$$T' \overset{(5.6)}{=} f(T) \;=\; h_0(\iota(T)) \overset{(5.5)}{\subseteq} h_0(T^*) \overset{(5.4)}{=} T_0,$$

which, since T' is a solution for S under M and T_0 is a $\subseteq$-minimal solution for S under M, implies that $T' = T_0$. Thus, by (5.6), $T_0 \in \mathit{poss}(T)$. □

Since T is isomorphic to $\mathrm{Core}(M, S)$, we have $\mathit{poss}(T) = \mathit{poss}(\mathrm{Core}(M, S))$. We are thus left with the following problem: given an instance T with $T \cong \mathrm{Core}(M, S)$ for some source instance S for M, and a tuple $\bar{t} \in \mathit{Const}^{\mathrm{ar}(q)}$, decide whether there is a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $\mathit{poss}(T)$ such that $\bigcup \mathcal{T} \models \neg q(\bar{t})$.

Note that, since q is a universal query, $\neg q$ is logically equivalent to an *existential query* of the form $\exists \bar{y}\, \varphi(\bar{x}, \bar{y})$. Before tackling the general case in Section 5.4.4, the following two sections deal with the case that $\bar{y}$ contains no variable and φ consists of a single atomic FO formula (Section 5.4.2), and the case that φ is a conjunction of atomic FO formulas and negations of atomic FO formulas (Section 5.4.3).

5.4.2 Finding Atoms of $\subseteq$-Minimal Possible Worlds

In this section, we prove one of the two key lemmas, Lemma 5.31, for the proof of Theorem 5.17. Originally, this lemma was proved to solve the following problem in polynomial time, where M is a fixed schema mapping defined by packed st-tgds: given a ground atom A (an atom of the form $R(\bar{c})$, where $\bar{c}$ contains only constants), and an instance T that is isomorphic to $\mathrm{Core}(M, S)$ for some source instance S for M, decide whether A is an atom of some $\subseteq$-minimal instance in $\mathit{poss}(T)$. We will develop the lemma with this particular application in mind, which gives us the opportunity to properly motivate all notions involved in the

lemma's statement. Sometimes, however, we will state results in a more general fashion, to apply them later in the more general setting of Section 5.4.3.

First note that there may be infinitely many $\subseteq$-minimal instances in $poss(T)$, so that it is impossible to check out all these instances. However, letting C be the set of all constants that occur in A, it suffices to consider only the instances in the set $min_C(T)$ defined as follows:

5.19 Definition ($val_C(T)$, $min_C(T)$)
Let T be a naive table, and let $C \subseteq Const$.

- Let $val_C(T)$ be the set of all mappings $f\colon \mathrm{dom}(T) \to \mathrm{dom}(T) \cup C$ such that $f(c) = c$ for all constants $c \in \mathrm{const}(T)$.
- Let $min_C(T)$ be the set of all instances T_0 for which there is some $f \in val_C(T)$ with $f(T) = T_0$, and there is no $f' \in val_C(T)$ with $f'(T) \subsetneq T_0$.

5.20 Proposition.
Let T be a naive table, and let $C \subseteq Const$.

1. *For each $T_0 \in poss(T)$, the following statements are equivalent:*
   (a) *T_0 is a $\subseteq$-minimal instance in $poss(T)$.*
   (b) *There is an instance $T_0' \in min_C(T)$ and an injective valuation v of T_0' such that $v(T_0') = T_0$.*
   (c) *There is an instance $T_0' \in min_C(T)$ and an injective valuation v of T_0' such that $v(T_0') = T_0$, and $v^{-1}(c) = c$ for all $c \in \mathrm{dom}(T_0) \cap C$.*
2. *If T is a core, then $T \in min_C(T)$.*
3. *Each instance in $min_C(T)$ is a core.*

Proof.

Ad 1: Clearly, (c) implies (b). Therefore, it suffices to prove that (a) implies (c), and that (b) implies (a).

We first prove that (a) implies (c). Assume that T_0 is a $\subseteq$-minimal instance in $poss(T)$. We show that there is an instance $T_0' \in min_C(T)$ and an injective valuation v of T_0' such that $v(T_0') = T_0$, and $v^{-1}(c) = c$ for all $c \in \mathrm{dom}(T_0) \cap C$.

Let v_0 be a valuation of T with $v_0(T) = T_0$. Furthermore, let $\overline{v}\colon \mathrm{dom}(T_0) \to \mathrm{dom}(T) \cup C$ be an injective mapping with

$$\overline{v}(c) = c \quad \text{for each } c \in \mathrm{dom}(T_0) \cap (\mathrm{const}(T) \cup C), \tag{5.7}$$

and

$$\overline{v}(c) \in \mathrm{nulls}(T) \quad \text{for each } c \in \mathrm{dom}(T_0) \setminus (\mathrm{const}(T) \cup C). \tag{5.8}$$

Then the composition $f := \overline{v} \circ v_0$ of v_0 and $\overline{v}$ belongs to $val_C(T)$, and

$$T_0' \;:=\; f(T) \;=\; \overline{v}(v_0(T)) \;=\; \overline{v}(T_0). \tag{5.9}$$

Now let v be the inverse of $\overline{v}$ on $\mathrm{dom}(T_0')$. That is, v is an injective mapping from $\mathrm{dom}(T_0')$ to $\mathrm{dom}(T_0)$ such that for all $u \in \mathrm{dom}(T_0')$ and all $u' \in \mathrm{dom}(T_0)$, we have $v(u) = u'$ if and only if $\overline{v}(u') = u$. By (5.7)–(5.9), v is an injective *valuation* of T_0' such that

$$v(T_0') \;=\; T_0, \tag{5.10}$$

and $v^{-1}(c) = \overline{v}(c) = c$ for each constant $c \in \mathrm{dom}(T_0) \cap C$.

It remains to show that $T_0' \in min_C(T)$. We already have $T_0' = f(T)$, where $f \in val_C(T)$. Thus, it suffices to show that there is no $f' \in val_C(T)$ with $f'(T) \subsetneq T_0'$. Suppose, for a contradiction, that there is such an f'. Since v is injective, we then have

$$v(f'(T)) \;\subsetneq\; v(T_0') \;\overset{(5.10)}{=}\; T_0,$$

which is impossible, since $v(f'(T)) \in poss(T)$, and T_0 is a $\subseteq$-minimal instance in $poss(T)$.

To prove that (b) implies (a), let $T_0' \in min_C(T)$ and let v be an injective valuation of T_0'. We show that $T_0 := v(T_0')$ is a $\subseteq$-minimal instance in $poss(T)$. To this end, let $f \in val_C(T)$ be such that $f(T) = T_0'$, and consider the composition $v_0 := v \circ f$ of f and v. Then v_0 is a valuation of T, so that

$$T_0 \;=\; v(T_0') \;=\; v(f(T)) \;=\; v_0(T) \;\in\; poss(T).$$

It remains, therefore, to show that there is no $\tilde{T}_0 \in poss(T)$ with $\tilde{T}_0 \subsetneq T_0$. For a contradiction, assume that there is such a $\tilde{T}_0$. Let $\tilde{v}_0$ be a valuation of T with $\tilde{v}_0(T) = \tilde{T}_0$, and consider the composition $\tilde{f} := v^{-1} \circ \tilde{v}_0$ of $\tilde{v}_0$ and the inverse v^{-1} of v on $\mathrm{dom}(T_0)$. Since v^{-1} is an injective mapping on $\mathrm{dom}(T_0)$, we have

$$\tilde{f}(T) \;=\; v^{-1}(\tilde{v}_0(T)) \;=\; v^{-1}(\tilde{T}_0) \;\subsetneq\; v^{-1}(T_0) \;=\; T_0',$$

which is impossible, since $\tilde{f} \in val_C(T)$ and $T_0' \in min_C(T)$.

Ad 2: Assume that T is a core. Let f be the identity on $\mathrm{dom}(T)$. Clearly, we have $f \in val_C(T)$ and $f(T) = T$. On the other hand, let $f' \in val_C(T)$ be such

that $f'(T) \subseteq T$. Then f' is a homomorphism from T to T, and since T is a core, we cannot have $f'(T) \subsetneq T$. Consequently, $T \in \mathit{min}_C(T)$.

Ad 3: Let $f \in \mathit{val}_C(T)$ be such that $T_0 := f(T)$ belongs to $\mathit{min}_C(T)$. For a contradiction, suppose that T_0 is not a core. Let h be a homomorphism from T_0 to T_0 such that $h(T_0)$ is a core of T_0. Since T_0 is not a core, we have $h(T_0) \subsetneq T_0$. Thus, for the composition $f' := h \circ f$ of f and h, we have

$$f'(T) \;=\; h(f(T)) \;=\; h(T_0) \;\subsetneq\; T_0,$$

which is a contradiction to $T_0 \in \mathit{min}_C(T)$. Hence, T_0 is a core. □

The converse of Proposition 5.20(3) does not hold, that is, a core does not automatically belong to $\mathit{min}_C(T)$:

5.21 Example
Let T be a naive table over $\sigma = \{E, P\}$, where $E^T = \{(a, \bot), (\bot, \bot')\}$ and $P^T = \{b\}$. The mapping $f \in \mathit{val}_C(T)$ with $f(\bot) = a$ and $f(\bot') = b$ then yields the instance $f(T)$, where $E^{f(T)} = \{(a, a), (a, b)\}$ and $P^{f(T)} = \{b\}$. Hence, $f(T)$ is a core. However, $f(T)$ does not belong to $\mathit{min}_C(T)$, since the mapping $f' \in \mathit{val}_C(T)$ with $f'(\bot) = f'(\bot') = a$ yields the instance $f'(T)$ with $E^{f'(T)} = \{(a, a)\}$ and $P^{f'(T)} = \{b\}$, which is a proper subinstance of $f(T)$. □

Note that the size of $\mathit{min}_C(T)$ can still be exponential in the size of T, so that it is not possible to enumerate all instances in $\mathit{min}_C(T)$ in polynomial time, given T and A as input. To tackle this problem, we take advantage of the fact that, by Lemma 2.30, there is a constant b that depends only on M such that all *blocks* of T have size at most b. The following variant of blocks will be convenient:

5.22 Definition (atom block)
Let T be a naive table.

- The *Gaifman graph of the atoms of* T is the undirected graph whose vertices are the atoms of T, and which has an edge between two atoms A and A' of T if and only if $A \neq A'$, and there is a null that occurs both in A and A'.
- An *atom block* of T is the set of atoms in a connected component of the Gaifman graph of the atoms of T.

Note that each atom block of T can be considered as a subinstance of T. Furthermore, for each atom block B of T that contains at least one null, $\mathrm{nulls}(B)$ is a block of T. Since each universal solution for S under M contains a subinstance that is isomorphic to $\mathrm{Core}(M, S)$, Lemma 2.30 immediately implies:

5.23 Lemma.
For every schema mapping $M = (\sigma_s, \sigma_t, \Sigma)$, where Σ consists of st-tgds, there is a positive integer b with the following property. If S is a source instance for M, and B is an atom block of $\mathrm{Core}(M, S)$, *then* $|\mathrm{nulls}(B)| \leq b$.

The following naive algorithm now seems to decide on input T and A whether there is an instance $T_0 \in \mathit{min}_C(T)$ with $A \in T_0$:

1. Compute the atom blocks of T.
2. Consider the atom blocks B of T in turn, and
3. if there is an instance $B_0 \in \mathit{min}_C(B)$ with $A \in B_0$, accept the input; otherwise reject it.

Since $|\mathrm{nulls}(B)| \leq b$ for each atom block B of T, we only have to consider at most $|\mathit{val}_C(B)| = |\mathrm{dom}(B) \cup C|^b$ mappings in step 3 to find all the instances B_0 in $\mathit{min}_C(B)$. Thus, the whole algorithm runs in polynomial time. However, Example 5.24 below shows that this algorithm is incorrect. In particular, Example 5.24 exhibits a naive table T that is a core, and an atom block B of T such that there is an atom of some $\subseteq$-minimal instance $B_0 \in \mathit{poss}(B)$ that is not an atom of any $\subseteq$-minimal instance in $\mathit{poss}(T)$. For all sets $C \subseteq \mathit{Const}$, this implies that there is an atom of some instance $B_0 \in \mathit{min}_C(B)$ that is not an atom of any instance in $\mathit{min}_C(T)$.

5.24 Example
Let T be the naive table over $\{E\}$ with

$$E^T = \{(a, b), (a, \bot), (b, \bot), (b, \bot'), (b, \bot''), (\bot', \bot'')\},$$

and consider the atom block B of T with

$$E^B = \{(b, \bot'), (b, \bot''), (\bot', \bot'')\}.$$

Figure 5.1 represents T and B as a graph. Note that T is a core. It is not hard to see that each $\subseteq$-minimal instance in $\mathit{poss}(B)$ has one of the following forms:

1. $\{E(b, b)\}$,
2. $\{E(b, c), E(c, c)\}$ with $c \in \mathit{Const} \setminus \{b\}$, or
3. $\{E(b, c), E(b, c'), E(c, c')\}$ with $c, c' \in \mathit{Const} \setminus \{b\}$ and $c \neq c'$.

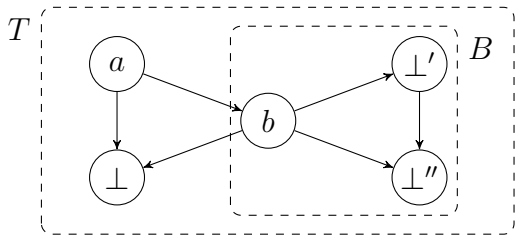


Figure 5.1: The naive table T and the atom block B

Thus, there is a $\subseteq$-minimal instance B_0 in $poss(B)$ of the third form such that $E(c, a)$ is an atom of B_0 (replace c' with a).

However, $E(c, a)$ is not an atom of any $\subseteq$-minimal instance in $poss(T)$. Such an instance must be obtained from T by a valuation v of T with $v(\bot') = c$ and $v(\bot'') = a$, since $E(\bot', \bot'')$ is the only atom of T that could be the "preimage" of $E(c, a)$—all other atoms either have the form $E(a, \cdot)$ or $E(b, \cdot)$. However, let v be a valuation of T with $v(\bot') = c$ and $v(\bot'') = a$, and let $f\colon \mathrm{dom}(T) \to \mathrm{dom}(T)$ be such that $f(a) = a$, $f(b) = b$, $f(\bot') = a$ and $f(\bot) = f(\bot'') = \bot$. Then, for the composition $v' := v \circ f$ of f and v, we have

$$\begin{aligned} v'(T) &= \{E(a,b), E(b,a), E(a,v(\bot)), E(b,v(\bot))\} \\ &\subsetneq \{E(a,b), E(b,a), E(a,v(\bot)), E(b,v(\bot)),\ E(b,c), E(c,a)\} \\ &= v(T). \end{aligned}$$

Thus, $v(T)$ is not $\subseteq$-minimal in $poss(T)$. □

We use a different approach: we identify a subset $\mathcal{S}$ of $\min_C(T)$ of size polynomial in the size of T such that it suffices to consider only the instances in $\mathcal{S}$ in order to decide whether A is an atom of some instance in $\min_C(T)$. More precisely, the set $\mathcal{S}$ consists of all naive tables T_0 for which there is an atom block B of T with $T_0 \in min_C(T, B)$, where $min_C(T, B)$ is defined below. For the definition of $min_C(T, B)$, we fix, for each instance I, a core $\mathrm{Core}(I)$ of I: the result of the algorithm from Lemma 2.31 on input I.

5.25 Definition ($val_C(T, B)$, $minval_C(T, B)$)
Let T be a naive table, let B be an atom block of T, and let $C \subseteq Const$.

- Let $val_C(T, B)$ be the set of all mappings $f \in val_C(T)$ such that
  - $f(\bot) = \bot$ for each null $\bot \in \mathrm{nulls}(T \setminus B)$, and
  - each null that occurs in $f(B) \setminus (T \setminus B)$ belongs to $\mathrm{nulls}(B)$.

- Let $minval_C(T, B)$ be the set of all mappings $f \in val_C(T, B)$ such that there is no $f' \in val_C(T, B)$ with $f'(T) \subsetneq f(T)$.
- Let $min_C(T, B) := \{\text{Core}(f(T)) \mid f \in minval_C(T, B)\}$.

Lemma 2.31 easily leads to:

5.26 Lemma.
There is an algorithm that takes a naive table T and a set $C \subseteq Const$ as input, and outputs a list of all instances that occur in $min_C(T, B)$ for some atom block B of T in time $O((n+|C|)^{2b+4})$, where n is the size of T and b is the maximum number of nulls in an atom block of T.

Proof. Given a naive table T and a set $C \subseteq Const$, we use the following algorithm $\mathbb{A}$ to compute a list of all instances that occur in $min_C(T, B)$ for some atom block B of T:

1. Compute the atom blocks $B_1, \ldots, B_k$ of T.
2. For each $i \in \{1, \ldots, k\}$:
   (a) Compute a list $f_1, \ldots, f_l$ of all mappings in $val_C(T, B_i)$.
   (b) Mark each mapping f_j for which there is no $j' \in \{1, \ldots, l\}$ with $f_{j'}(T) \subsetneq f_j(T)$.
   (c) For each marked mapping f_j, output $\text{Core}(f_j(T))$.

Clearly, $\mathbb{A}$ outputs a list of all instances that occur in $min_C(T, B)$ for some atom block B of T.

Let us now analyze the running time of $\mathbb{A}$. We first consider step 2(a). Let $i \in \{1, \ldots, m\}$. Then the mappings $f_1, \ldots, f_n$ in $val_C(T, B_i)$ can be computed using the following algorithm $\mathbb{B}$:

I. Compute a list $g_1, \ldots, g_p$ of all mappings from $\text{nulls}(B_i)$ to $\text{dom}(T) \cup C$.

II. For each $j \in \{1, \ldots, p\}$:
   (a) Extend g_j to a mapping $f\colon \text{dom}(T) \to \text{dom}(T) \cup C$ such that $f(u) = u$ for each $u \in \text{dom}(T) \setminus \text{nulls}(B_i)$.
   (b) Check whether each atom of $f(B_i)$ that is no atom of $T \setminus B_i$ contains only nulls from B_i. If so, output f.
   (c) Continue with the next j.

The overall running time of $\mathbb{B}$ is at most $O((n + |C|)^{2b})$. To see this, note that step I can be accomplished in time $O((n + |C|)^b)$. Furthermore, step II is repeated at most $|\mathrm{dom}(T) \cup C|^b \leq (n+|C|)^b$ times, where step II(a) needs time at most $O(n)$, and step II(b) needs time at most $O(n^2)$.

Let us now consider step 2(b) and step 2(c) of $\mathbb{A}$. Note that the number l of mappings $f_1, \ldots, f_l$ computed in step 2(a) is at most $|\mathrm{dom}(T) \cup C|^b \leq (n+|C|)^b$. Step 2(b) can therefore be accomplished in time $O(l^2 \cdot |T|) = O((n + |C|)^{2b+1})$. Furthermore, by Lemma 2.31, step 2(c) needs time at most $O(l \cdot n^{b+3}) = O((n+|C|)^{2b+3})$ since the number of nulls in each block of $f_j(T)$ is bounded by b (by the construction of the mappings in $val_C(T, B)$).

Finally, note that T has at most $|T| \leq n$ atom blocks, which implies that step 2 is repeated at most $k \leq n$ times. Using the bounds on the running time for steps 2(a)–(c) obtained above, we conclude that step 2 needs time at most $O(n \cdot (n + |C|)^{2b+3}) = O((n + |C|)^{2b+4})$. This clearly dominates the time needed for step 1. Thus, the running time of $\mathbb{A}$ is at most $O((n + |C|)^{2b+4})$. □

The following Lemma 5.27 tells us that the instances in $min_C(T, B)$ indeed belong to $min_C(T)$. To state and to prove the lemma, the notion of a retraction is convenient. Given a naive table T, a *retraction of* T is a homomorphism h from T to T such that $h(u) = u$ for each element u in the range $h(\mathrm{dom}(T))$ of h. In particular, each atom $R(\bar{u})$ of $h(T)$ belongs to T, and $R(\bar{u}) = \bar{u}$. It is known that a core of T is a naive table T' for which there is a retraction h of T with $h(T) = T'$, and there is no retraction h' of T' with $h'(T') \subsetneq T'$ (see, e.g., Hell and Nešetřil [1992]). A *retraction of T over a set $X \subseteq Dom$* is a retraction h of T such that $h(u) = u$ for each $u \in X \cap \mathrm{dom}(T)$.

5.27 Lemma.
Let T be a naive table, let B be an atom block of T, and let $C \subseteq Const$. Then for each $f \in minval_C(T, B)$, there is a retraction h of $\hat{T} := f(T)$ over the set of the nulls of $f(B) \setminus (T \setminus B)$ such that

1. *$h(\hat{T})$ is a core of $\hat{T}$ and*
2. *$h(\hat{T}) \in min_C(T)$.*

In particular, $min_C(T, B) \subseteq min_C(T)$.

Proof. Fix $\overline{B} := T \setminus B$ and $\mathcal{A} := f(B) \setminus \overline{B}$. Let h be a retraction of $\hat{T}$ over nulls($\mathcal{A}$) such that for

$$\hat{T}_0 \;:=\; h(\hat{T}) \;=\; h(f(T)) \tag{5.11}$$

we have:

$$\text{There is no retraction } h' \text{ of } \hat{T}_0 \text{ over nulls}(\mathcal{A}) \text{ with } h'(\hat{T}_0) \subsetneq \hat{T}_0. \tag{5.12}$$

We show that $\hat{T}_0$ is a core of $\hat{T}$, and that $\hat{T}_0 \in \mathit{min}_C(T)$.

Step 1: $\hat{T}_0$ is a core of $\hat{T}$.
Suppose, for a contradiction, that $\hat{T}_0$ is not a core of $\hat{T}$. Then there is a retraction h' of $\hat{T}_0$ with $h'(\hat{T}_0) \subsetneq \hat{T}_0$. By (5.12), there is some $\bot \in \mathrm{nulls}(\mathcal{A})$ with $h'(\bot) \neq \bot$. Since h' is a retraction of $\hat{T}_0$, and $\mathcal{A} \subseteq \hat{T}_0$, this implies

$$h'(\mathcal{A}) \setminus \overline{B} \;\subsetneq\; \mathcal{A}. \tag{5.13}$$

Now consider the mapping $f'\colon \mathrm{dom}(T) \to \mathrm{dom}(T) \cup C$ that is defined for each $u \in \mathrm{dom}(T)$ by

$$f'(u) \;:=\; \begin{cases} h'(h(f(u))), & \text{if } u \in \mathrm{nulls}(B), \\ u, & \text{otherwise.} \end{cases}$$

Then $f' \in \mathit{val}_C(T, B)$, since $f \in \mathit{val}_C(T, B)$ and h, h' are retractions. Moreover,

$$f'(B) \;\subseteq\; h'(\hat{T}_0) \;=\; h'(\hat{T}_0 \setminus \overline{B}) \cup h'(\overline{B}) \;\subseteq\; h'(\hat{T}_0 \setminus \overline{B}) \cup \overline{B}, \tag{5.14}$$

where the first inclusion is true due to $f'(B) = h'(h(f(B)))$ and (5.11), and the last inclusion is true, because h' is a retraction of $\hat{T}_0$.

Observe that

$$\hat{T}_0 \setminus \overline{B} \;\subseteq\; \mathcal{A}. \tag{5.15}$$

Indeed, let A be an atom of $\hat{T}_0$ with $A \notin \overline{B}$. By (5.11), we have $\hat{T}_0 = h(f(T))$. Together with $f(T) = f(B) \cup f(\overline{B})$, and the fact that h is a retraction of $f(T)$, this implies that $A \in f(B)$ or $A \in f(\overline{B})$. Note that $A \notin f(\overline{B})$, because $f(\overline{B}) = \overline{B}$ and $A \notin \overline{B}$. Hence, $A \in f(B)$, which, together with $A \notin \overline{B}$, implies that $A \in \mathcal{A}$.

Consequently, we have

$$\begin{aligned}
f'(T) &= f'(B) \cup f'(\overline{B}) && \text{since } B \cup \overline{B} = T \\
&= f'(B) \cup \overline{B} && \text{since } f' \in \mathit{val}_C(T, B) \\
&\subseteq h'(\mathcal{A}) \cup \overline{B} && \text{by (5.14) and (5.15)} \\
&\subsetneq \mathcal{A} \cup \overline{B} && \text{by (5.13)} \\
&= f(B) \cup \overline{B} && \text{by the definition of } \mathcal{A} \\
&= f(T) && \text{since } \overline{B} = f(\overline{B}) \text{ and } B \cup \overline{B} = T,
\end{aligned}$$

which is a contradiction to $f \in \mathit{minval}_C(T, B)$. Thus, $\hat{T}_0$ is a core of $\hat{T}$.

Step 2: $\hat{T}_0 \in \mathit{min}_C(T)$.
First observe that $\hat{T}_0 = f_0(T)$, where $f_0 := h \circ f$ and $f_0 \in \mathit{val}_C(T)$. It remains, therefore, to show that there is no $f' \in \mathit{val}_C(T)$ with $f'(T) \subsetneq \hat{T}_0$.

Suppose, for a contradiction, that there is such a mapping f'. Without loss of generality, we can assume that $f'(T) \in \mathit{min}_C(T)$. Moreover,

$$f'(T) \;\subseteq\; \hat{T}_0 \;\overset{(5.11)}{=}\; h(f(T)) \;\subseteq\; f(T) \;=\; f(B) \cup \overline{B}. \tag{5.16}$$

Let us next observe that

$$f'(B) \setminus \overline{B} \;=\; f(B) \setminus \overline{B}. \tag{5.17}$$

Otherwise, we could use f' to construct a mapping $f'' \in \mathit{val}_C(T, B)$ such that $f''(T) \subsetneq f(T)$, which is impossible, because $f \in \mathit{minval}_C(T, B)$. Indeed, assume that (5.17) is not true. Then,

$$f'(B) \setminus \overline{B} \;\subsetneq\; f(B) \setminus \overline{B}, \tag{5.18}$$

since by (5.16) we have $f'(B) \subseteq f(B) \cup \overline{B}$. Let $f''\colon \mathrm{dom}(T) \to \mathrm{dom}(T) \cup C$ be such that for all $u \in \mathrm{dom}(T)$,

$$f''(u) \;=\; \begin{cases} f'(u), & \text{if } u \in \mathrm{nulls}(B) \\ u, & \text{otherwise.} \end{cases}$$

It is not hard to see that $f'' \in \mathit{val}_C(T, B)$. Moreover,

$$f''(T) \;=\; (f'(B) \setminus \overline{B}) \cup \overline{B} \;\overset{(5.18)}{\subsetneq}\; (f(B) \setminus \overline{B}) \cup \overline{B} \;=\; f(T).$$

Now, (5.17) implies that the mapping $h'\colon \mathrm{dom}(T) \to \mathrm{dom}(T) \cup C$ that is defined for each $u \in \mathrm{dom}(T)$ by

$$h'(u) \;=\; \begin{cases} u, & \text{if } u \in \mathrm{nulls}(B) \\ f'(u), & \text{otherwise} \end{cases}$$

is a homomorphism from $f(T) = f(B) \cup \overline{B}$ to $f'(T)$. Furthermore, by (5.16) we have $f'(T) \subseteq f(T)$, and therefore, the identity on $\mathrm{dom}(f'(T))$ is a homomorphism from $f'(T)$ to $f(T)$. This implies that $f'(T)$ and $f(T)$ are homomorphically equivalent, and therefore, their cores are isomorphic. Since $\hat{T}_0$ is a core of $f(T)$ by step 1, and $f'(T)$ is a core by Proposition 5.20(3), we have $\hat{T}_0 \cong f'(T)$. This contradicts the assumption that $f'(T) \subsetneq \hat{T}_0$. □

Clearly, the union of the sets $min_C(T, B)$ over all atom blocks B of T does not cover the whole set of instances in $min_C(T)$. However, Lemma 5.31 below tells us that for each atom A of some instance $T_0 \in min_C(T)$ there is an atom block B of T and an instance $T_B \in min_C(T, B)$ that contains an atom A' isomorphic to A in the following sense:

5.28 Notation
We say that two atoms A_1, A_2 are *isomorphic*, and we write $A_1 \cong A_2$, if and only if the instances $\{A_1\}$ and $\{A_2\}$ are isomorphic.

Note that two atoms $R(u_1, \ldots, u_r)$ and $R'(u'_1, \ldots, u'_{r'})$ are isomorphic if and only if $R = R'$, $r = r'$, and for all $i, j \in \{1, \ldots, r\}$, $u_i \in Const$ if and only if $u'_i \in Const$, $u_i \in Const$ implies $u_i = u'_i$, and $u_i = u_j$ if and only if $u'_i = u'_j$.

Given an atom $A = R(u_1, \ldots, u_r)$ and a mapping $f\colon X \to Dom$ with $\{u_1, \ldots, u_r\} \subseteq X$, we let $f(A)$ be the atom $R(f(u_1), \ldots, f(u_r))$.

Finally, Lemma 5.31 is based on the following notion of *packed atom block*:

5.29 Definition (packed atom block)
An atom block B of a naive table is called *packed* if and only if for all atoms $A, A' \in B$ with $A \neq A'$, there is a null that occurs both in A and A'.

Immediately from the definitions, we obtain:

5.30 Proposition.
If $M = (\sigma_s, \sigma_t, \Sigma)$ is a schema mapping, where Σ consists of packed st-tgds, and S is a source instance for M, then each atom block of $\mathrm{Core}(M, S)$ *is packed.*

We are now ready to state the present section's main result:

5.31 Lemma.
Let T be a naive table such that T is a core and each atom block of T is packed. Let $C \subseteq Const$, and let $T_0 \in min_C(T)$. Then for each atom $A \in T_0$, there is

1. *an atom block B of T,*
2. *an instance $T_B \in min_C(T, B)$,*
3. *an atom $A' \in T_B$ with $A' \cong A$, and*
4. *a homomorphism h from T_B to T_0 with $h(T_B) = T_0$ and $h(A') = A$.*

Proof. Let T be a naive table such that T is a core and each atom block of T is packed, let $C \subseteq Const$, and let $T_0 \in min_C(T)$. Let $B_1, \ldots, B_n$ be an

enumeration of all the atom blocks of T. We prove a stronger statement, namely:

> For each $i \in \{1, \ldots, n\}$, there is an instance $T_i \in \mathit{min}_C(T, B_i)$ and a homomorphism h_i from T_i to T_0 with $h_i(T_i) = T_0$ such that the following is true: For each atom $A \in T_0$, there is an index $i \in \{1, \ldots, n\}$ and an atom $A' \in T_i$ with $h_i(A') = A$ and $A' \cong A$. $(\star)$

Let $f \in \mathit{val}_C(T)$ be such that

$$f(T) \;=\; T_0.$$

The construction of the instances T_i and the homomorphisms h_i proceeds in three steps. First, we "split" f into mappings $f_1 \in \mathit{val}_C(T, B_1), \ldots, f_n \in \mathit{val}_C(T, B_n)$ such that each $f_i(B_i)$ is isomorphic to $f(B_i)$. Second, we use these mappings to construct the instances T_i and the homomorphisms h_i. Third, we show that for each atom $A \in T_0$, there is an index $i \in \{1, \ldots, n\}$ and an atom $A' \in T_i$ with $h_i(A') = A$ and $A' \cong A$.

Step 1: "Split" f into mappings $f_1 \in \mathit{val}_C(T, B_1), \ldots, f_n \in \mathit{val}_C(T, B_n)$ such that each $f_i(B_i)$ is isomorphic to $f(B_i)$.
Let $i \in \{1, \ldots, n\}$. We construct a mapping $f_i \in \mathit{val}_C(T, B_i)$ and an injective homomorphism r_i from $f_i(B_i)$ to $f(B_i)$ with $r_i(f_i(B_i)) = f(B_i)$ as follows. Pick an injective mapping

$$\bar{r}_i \colon \mathrm{dom}(f(B_i)) \to \mathrm{const}(f(B_i)) \cup \mathrm{nulls}(B_i)$$

such that $\bar{r}_i(c) = c$ for each $c \in \mathrm{const}(f(B_i))$, and $\bar{r}_i(\bot) \in \mathrm{nulls}(B_i)$ for each $\bot \in \mathrm{nulls}(f(B_i))$. Then let $f_i \colon \mathrm{dom}(T) \to \mathrm{dom}(T) \cup C$ be such that for each $u \in \mathrm{dom}(T)$, we have

$$f_i(u) \;:=\; \begin{cases} \bar{r}_i(f(u)), & \text{if } u \in \mathrm{nulls}(B_i) \\ u, & \text{otherwise.} \end{cases}$$

By the construction of f_i, we have $f_i \in \mathit{val}_C(T, B_i)$. Furthermore, for each atom A of $f(B_i)$, we have $\bar{r}_i(A) \cong A$. In particular, each atom of $f(B_i)$ is isomorphic to an atom of $f_i(B_i)$, and vice versa. Let r_i be the inverse of $\bar{r}_i$ on $\mathrm{dom}(f_i(B_i))$. Then, r_i is an injective homomorphism from $f_i(B_i)$ to $f(B_i)$ with

$$r_i(f_i(B_i)) \;=\; r_i(\bar{r}_i(f(B_i))) \;=\; f(B_i). \tag{5.19}$$

In particular,

$$r_i(A) \;\cong A \quad \text{for all atoms } A \in f_i(B_i). \tag{5.20}$$

Step 2: Construction of the instances T_i and the homomorphisms h_i.
Let $i \in \{1, \ldots, n\}$, and pick a mapping $g_i \in \mathit{minval}_C(T, B_i)$ with $g_i(T) \subseteq f_i(T)$. By Lemma 5.27, there is a retraction h'_i of $g_i(T)$ over the set of the nulls of $g_i(B_i) \setminus (T \setminus B_i)$ such that

$$T_i \;:=\; h'_i(g_i(T)) \;\in\; \mathit{min}_C(T). \tag{5.21}$$

Now consider the mapping $h_i\colon \mathrm{dom}(g_i(T)) \to \mathrm{dom}(T_0)$ such that for each $u \in \mathrm{dom}(g_i(T))$, we have

$$h_i(u) \;:=\; \begin{cases} r_i(u), & \text{if } u \in \mathrm{dom}(g_i(T) \setminus (T \setminus B_i)) \\ f(u), & \text{otherwise.} \end{cases}$$

Note that r_i is defined for all values that occur in $\mathrm{dom}(g_i(T) \setminus (T \setminus B_i))$, since $g_i(T) \subseteq f_i(T) = f_i(B_i) \cup (T \setminus B_i)$, and therefore,

$$g_i(T) \setminus (T \setminus B_i) \;\subseteq\; f_i(B_i). \tag{5.22}$$

Furthermore,

$$h_i(g_i(T) \setminus (T \setminus B_i)) \;=\; r_i(g_i(T) \setminus (T \setminus B_i)) \overset{(5.22)}{\subseteq} r_i(f_i(B_i)) \overset{(5.19)}{=} f(B_i) \;\subseteq\; T_0,$$

and

$$h_i(T \setminus B_i) \;=\; f(T \setminus B_i) \;\subseteq\; f(T) \;=\; T_0,$$

which yields $h_i(g_i(T)) \subseteq T_0$. In particular, we have

$$h_i(h'_i(g_i(T))) \;\subseteq\; h_i(g_i(T)) \;\subseteq\; T_0.$$

Since the composition $h_i \circ h'_i \circ g_i$ belongs to $\mathit{val}_C(T)$, and $T_0 \in \mathit{min}_C(T)$, it follows that $h_i(h'_i(g_i(T))) = T_0$, and hence,

$$h_i(T_i) \overset{(5.21)}{=} h_i(h'_i(g_i(T))) \;=\; T_0.$$

Hence, the restriction $\tilde{h}_i$ of h_i to $\mathrm{dom}(T_i)$ is a homomorphism from T_i to T_0 with $\tilde{h}_i(T_i) = T_0$.

Step 3: For each atom $A \in T_0$ there is an $i \in \{1, \ldots, n\}$ and an atom $A' \in T_i$ with $h_i(A') = A$ and $A' \cong A$.
Let

$$T^* \;:=\; \bigcup_{i=1}^{n} \Big(g_i(B_i) \setminus (T \setminus B_i)\Big).$$

To prove ($\star$), it suffices to show that there is a mapping $r\colon \mathrm{dom}(T^*) \to \mathrm{dom}(T_0)$ such that

1. $r(T^*) = T_0$,
2. $r(A') \cong A'$ for each $A' \in T^*$, and
3. $r(A') = h_i(A')$ for each $i \in \{1, \ldots, n\}$ and each $A' \in g_i(B_i) \setminus (T \setminus B_i)$.

Indeed, let $A \in T_0$. Since $r(T^*) = T_0$ by condition 1, there is some $A' \in T^*$ with $r(A') = A$. So, by the construction of T^*, there is an $i \in \{1, \ldots, n\}$ such that $A' \in g_i(B_i) \setminus (T \setminus B_i) \subseteq T_i$. Condition 3 then yields $h_i(A') = r(A') = A$, and since $r(A') \cong A'$ by condition 2, we have $A' \cong A$.

Let $r\colon \mathrm{dom}(\bigcup_{i=1}^n f_i(B_i)) \to \mathrm{dom}(T_0)$ be such that for each $i \in \{1, \ldots, n\}$ and each $u \in \mathrm{dom}(f_i(B_i))$, we have

$$r(u) \;=\; r_i(u).$$

This is well-defined, since $\mathrm{nulls}(f_i(B_i)) \cap \mathrm{nulls}(f_j(B_j)) = \emptyset$ for all distinct $i, j \in \{1, \ldots, n\}$, and each r_i is the identity on constants. We claim that r satisfies conditions 1–3 above.

By the construction of r and h_i, r already satisfies conditions 2 and 3. To see that r satisfies condition 2, let $A' \in T^*$. Then there is some $i \in \{1, \ldots, n\}$ with $A' \in g_i(B_i) \setminus (T \setminus B_i)$. By $r(A') = r_i(A')$, (5.20), and (5.22), we thus have $r(A') \cong A'$. To see that r satisfies condition 3, let $i \in \{1, \ldots, n\}$ and $A' \in g_i(B_i) \setminus (T \setminus B_i)$. Then, $r(A') = r_i(A') = h_i(A')$, where the last equality holds due to the construction of h_i.

It thus remains to show that $r(T^*) = T_0$. Let us first show that $r(T^*) \subseteq T_0$. Note that, by (5.22), we have

$$T^* \;\subseteq\; \bigcup_{i=1}^{n} f_i(B_i).$$

Hence,

$$r(T^*) \subseteq r\left(\bigcup_{i=1}^{n} f_i(B_i)\right) = \bigcup_{i=1}^{n} r(f_i(B_i)) = \bigcup_{i=1}^{n} r_i(f_i(B_i)) \overset{(5.19)}{=} \bigcup_{i=1}^{n} f(B_i) = T_0.$$

To show that $r(T^*) = T_0$, we show that there is a mapping $f^* \in \mathit{val}_C(T)$ with $f^*(T) = T^*$. Then, the composition $f' := r \circ f^*$ of f^* and r belongs to $\mathit{val}_C(T)$. In particular, since $T_0 \in \mathit{min}_C(T)$ and $f'(T) = r(f^*(T)) = r(T^*) \subseteq T_0$, this implies $r(T^*) = T_0$, and the proof is complete.

Thus, it remains to show that there is a mapping $f^* \in \mathit{val}_C(T)$ with $f^*(T) = T^*$. We find this mapping by constructing a sequence of appropriate mappings that are obtained from the mappings $g_1, \ldots, g_n$.

Let us first modify $g_1, \ldots, g_n$ as follows. Choose an arbitrary "renaming" of the nulls of T. That is, pick an injective mapping $\rho\colon \mathrm{dom}(T) \to \mathrm{const}(T) \cup$

$(\mathit{Null} \setminus \text{nulls}(T))$ such that $\rho(c) = c$ for each constant $c \in \text{const}(T)$. Note that ρ maps each null of T to a unique null that does not occur in T. Let

$$\mathcal{X} \;:=\; \rho(\text{nulls}(T)).$$

For each $i \in \{1, \ldots, n\}$ then define $\hat{g}_i \colon \text{dom}(T) \cup C \cup \mathcal{X} \to \text{dom}(T) \cup C \cup \mathcal{X}$ such that for each $u \in \text{dom}(T) \cup C \cup \mathcal{X}$,

$$\hat{g}_i(u) \;:=\; \begin{cases} g_i(\rho^{-1}(u)), & \text{if } u \in \rho(\text{nulls}(B_i)) \text{ and } g_i(\rho^{-1}(u)) \in \text{nulls}(B_i) \\ \rho(g_i(\rho^{-1}(u)), & \text{if } u \in \rho(\text{nulls}(B_i)) \text{ and } g_i(\rho^{-1}(u)) \notin \text{nulls}(B_i) \\ u, & \text{otherwise.} \end{cases}$$

Note that for each $i \in \{1, \ldots, n\}$,

$$\hat{g}_i(\rho(B_i)) \setminus \rho(T) \;=\; g_i(B_i) \setminus (T \setminus B_i). \tag{5.23}$$

Now let $\hat{g} := \hat{g}_n \circ \cdots \circ \hat{g}_2 \circ \hat{g}_1$. Furthermore, let $\hat{g}^0 := \rho$, and for each $s \geq 0$, let $\hat{g}^{s+1} := \hat{g} \circ \hat{g}^s$. Since $\hat{g}$ is the identity on $\text{dom}(T) \cup C$, it is easy to see that

$$\hat{g}^1(T) \;\supseteq\; \hat{g}^2(T) \;\supseteq\; \hat{g}^3(T) \;\supseteq\; \cdots .$$

Since $\hat{g}^1(T)$ is finite, there exists therefore an integer $s_0 \geq 1$ such that $\hat{g}^{s_0}(T) = \hat{g}^s(T)$ for each $s \geq s_0$. Let $f^* := \hat{g}^{s_0}$.

We show that $f^*(T) = T^*$. First observe that

$$T^* \;=\; \bigcup_{i=1}^{n} \Big(g_i(B_i) \setminus (T \setminus B_i) \Big) \;=\; \hat{g}^1(T) \setminus \rho(T) \;=\; f^*(T) \setminus \rho(T).$$

To see that T^* is not a proper subinstance of $f^*(T)$, we show that $f^*(T)$ contains no atoms from $\rho(T)$.

For a contradiction, suppose that $f^*(T)$ contains an atom $A \in \rho(T)$. Then, $A \in \rho(B_i)$ for some $i \in \{1, \ldots, n\}$. Since $\hat{g}(f^*(T)) = f^*(T)$, we know that $\hat{g}$ is a bijection on $\text{dom}(f^*(T))$. Furthermore, since $\hat{g}$ is the identity on $\text{dom}(T) \cup C$, we have $\hat{g}(\bot) \in \mathcal{X}$ for each $\bot \in \mathcal{X}$. It follows that

$$\hat{g}_i(A) \cong A, \qquad \text{and} \qquad \hat{g}_i(A) \in \rho(B_j) \quad \text{for some } j \in \{1, \ldots, n\} \setminus \{i\}.$$

Let $A' := \rho^{-1}(A)$. By the construction of $\hat{g}_i$, we have

$$g_i(A') \cong A', \qquad \text{and} \qquad g_i(A') \in B_j \quad \text{for some } j \in \{1, \ldots, n\} \setminus \{i\}.$$

Since B_i is packed and g_i maps each null in A' to a null in B_j, each atom in $g_i(B_i)$ contains a null from B_j. Together with $g_i \in \mathit{val}_C(T, B_i)$, this implies $g_i(B_i) \subseteq B_j$. In other words, g_i is a homomorphism from T to $T \setminus B_i$, which contradicts the fact that T is a core. Consequently, $f^*(T) = T^*$.

Altogether, the proof of Lemma 5.31 is complete. □

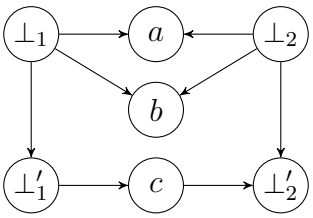
A straightforward polynomial time algorithm that, given T and A, decides whether $A = R(\bar{t})$ is an atom of some $\subseteq$-minimal instance in $poss(T)$ is now as follows. Let C be the set of all constants in $\bar{t}$. Consider each atom block B of T, and each $T_0 \in min_C(T, B)$ in turn, and accept the input if and only if $A \in T_0$ for some T_0. By Lemma 5.26, the instances T_0 can be computed in polynomial time, so the whole procedure runs in polynomial time.

The following example shows that the proof of Lemma 5.31 fails if T contains atom blocks that are not packed.

5.32 Example
Let E be a binary relation symbol, and consider the naive table T over $\{E\}$ with

$$E^T = \{(\bot_1, a), (\bot_1, b), (\bot_1, \bot_1'), (\bot_1', c), (\bot_2, a), (\bot_2, b), (\bot_2, \bot_2'), (c, \bot_2')\}.$$

Viewed as a graph, T looks as follows:



Note that T is a core, and that T has the two atom blocks

$$B_1 = \{E(\bot_1, a), E(\bot_1, b), E(\bot_1, \bot_1'), E(\bot_1', c)\}, \text{ and}$$
$$B_2 = \{E(\bot_2, a), E(\bot_2, b), E(\bot_2, \bot_2'), E(c, \bot_2')\}.$$

Note also that neither B_1 nor B_2 is packed.

Consider $f \in val_\emptyset(T)$ with $f(\bot_1) = f(\bot_2) = a$ and $f(\bot_1') = f(\bot_2') = b$. Then it is not hard to see that

$$f(T) = \{E(a, a), E(a, b), E(b, c), E(c, b)\} \in min_\emptyset(T).$$

Furthermore, for the mappings f_i created in the proof of Lemma 5.31, we have

$$f_1(T) = \{E(a, a), E(a, b), E(b, c), E(\bot_2, a), E(\bot_2, b), E(\bot_2, \bot_2'), E(c, \bot_2')\}$$

and

$$f_2(T) = \{E(a, a), E(a, b), E(c, b), E(\bot_1, a), E(\bot_1, b), E(\bot_1, \bot_1'), E(\bot_1', c)\}.$$

For $g_i \in \mathit{val}_\emptyset(T, B_i)$ with $g_i(\bot_i) = \bot_{3-i}$ and $g_i(\bot_i') = b$, it holds that $g_i \in \mathit{minval}_\emptyset(T, B_i)$, and moreover,

$$\begin{aligned} g_1(T) &= \{E(\bot_2, a), E(\bot_2, b), E(\bot_2, \bot_2'), E(c, \bot_2'), E(b, c)\}, \\ g_2(T) &= \{E(\bot_1, a), E(\bot_1, b), E(\bot_1, \bot_1'), E(\bot_1', c), E(c, b)\}. \end{aligned}$$

It is not hard to see that $g_1(T)$ and $g_2(T)$ belong to $\mathit{min}_\emptyset(T)$. Finally, note that $E(a, a)$ and $E(a, b)$ occur in $f_i(B_i)$, but neither $g_1(T)$ nor $g_2(T)$ contains an atom A' with $A' \cong E(a, a)$ or $A' \cong E(a, b)$. □

5.4.3 Existentially Quantified Conjunctions of Atomic Formulas and Negations of Atomic Formulas

In this section, we prove the second key lemma for the proof of Theorem 5.17:

5.33 Lemma.
Let $q(\bar{x}) = \exists \bar{y}\, \varphi(\bar{x}, \bar{y})$ be an existential query over a schema σ, where φ has the form $\bigwedge_{i=1}^{p} \varphi_i$, and each φ_i is an atomic FO formula or the negation of an atomic FO formula. Then for each positive integer b, there is a polynomial time algorithm that decides:

CoreEval$_{\sigma,b}$	
Input:	*a naive table T over σ such that T is a core and each atom block of T is packed and contains at most b nulls, and a tuple $\bar{t} \in \mathit{Const}^{\mathrm{ar}(q)}$*
Question:	*Is there a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $\mathit{poss}(T)$ such that $\bigcup \mathcal{T} \models q(\bar{t})$?*

The remainder of this section presents a proof of Lemma 5.33. Let $q(\bar{x}) = \exists \bar{y}\, \varphi(\bar{x}, \bar{y})$ be as in the hypothesis of Lemma 5.33. Without loss of generality, there is no variable that occurs both in $\bar{x}$ and in $\bar{y}$, and φ has the form $\bigwedge_{i=1}^{p} \varphi_i$, where each φ_i is a relational atomic FO formula, the negation of a relational atomic FO formula, or the negation of an equality.

Let b be a positive integer. Then CoreEval$_{\sigma,b}$ can be solved as follows. Assume we are given a naive table T over σ such that T is a core and each atom block of T is packed and contains at most b nulls, and a tuple $\bar{t} \in \mathit{Const}^{\mathrm{ar}(q)}$ as input. In a first step, we rewrite φ to a formula ψ by replacing each variable x in $\bar{x}$ by the corresponding constant assigned to x by $\bar{t}$. That is, if $\bar{x} = (x_1, \ldots, x_k)$ and $\bar{t} = (t_1, \ldots, t_k)$, then ψ is obtained from φ by replacing, for each $i \in \{1, \ldots, k\}$, each occurrence of the variable x_i in φ by t_i. Let

$$\tilde{q} \;:=\; \exists \bar{y}\, \psi(\bar{y}).$$

To check whether there is a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $poss(T)$ such that $\bigcup\mathcal{T} \models q(\bar{t})$, it now suffices to check whether there is a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $poss(T)$ such that $\bigcup\mathcal{T} \models \tilde{q}$.

Let us assume that

$$\psi(\bar{y}) \quad = \quad \bigwedge_{i=1}^{k} R_i(\bar{x}_i) \ \wedge \ \bigwedge_{i=1}^{l} \neg Q_i(\bar{w}_i) \ \wedge \ \bigwedge_{i=1}^{m} \neg v_i = v_i'.$$

Let $C := \mathrm{dom}(\psi)$, and for each $i \in \{1, \ldots, k\}$, let X_i be the set of all variables in $\bar{x}_i$. Given an assignment α for a set X of variables, and a tuple $\bar{t}$ over $X \cup Const$, we sloppily write $\alpha(\bar{t})$ for the tuple obtained from $\bar{t}$ by replacing each occurrence of a variable $x \in X$ in $\bar{t}$ with $\alpha(x)$.

To check whether there is a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $poss(T)$ such that $\bigcup\mathcal{T} \models \tilde{q}$, we now do the following:

1. For each $i \in \{1, \ldots, s\}$, where

$$s \ := \ k + \sum_{i=1}^{l} |\bar{w}_i| + 2 \cdot m,$$

   we pick an injective mapping $\rho_i \colon \mathrm{dom}(T) \cup C \to Dom$ such that $\rho_i(c) = c$ for each $c \in \mathrm{const}(T) \cup C$, and such that for all distinct $i, j \in \{1, \ldots, s\}$, we have $\mathrm{nulls}(\rho_i(T)) \cap \mathrm{nulls}(\rho_j(T)) = \emptyset$.

2. We compute, for each $i \in \{1, \ldots, k\}$, the set

$$\begin{aligned} \mathcal{X}_i \ := \ \{(T_0, \alpha) \mid\ & T_0 = \rho_i(T_0') \text{ for some } T_0' \in min_C(T, B) \\ & \text{and an atom block } B \text{ of } T, \\ & \alpha \colon X_i \to \mathrm{dom}(T_0), \text{ and } \alpha(\bar{x}_i) \in R_i^{T_0}\}. \end{aligned}$$

3. We join pairs in $\mathcal{X}_1, \ldots, \mathcal{X}_k$ that are *compatible* in the sense described below to single pairs $(\tilde{T}, \tilde{\alpha})$ obtained by identifying as few values in $T_1, \ldots, T_k$ as possible such that for each $i \in \{1, \ldots, k\}$ we have $\tilde{\alpha}(\bar{x}_i) \in R_i^{\tilde{T}}$.

4. We check whether for one of the pairs $(\tilde{T}, \tilde{\alpha})$ from step 3, the extension $\tilde{T} \cup \bigcup_{i=k+1}^{s} \rho_i(T)$ of $\tilde{T}$ by $s - k$ disjoint copies of T satisfies $\tilde{q}$. If yes, then we accept the input; otherwise, we reject it.

To formally describe the last two steps of the algorithm, we need to introduce some more technical machinery.

Intuitively, $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$ are compatible if, by identifying values that occur in $\bigcup_{i=1}^{k} \alpha_i(X_i)$, it is possible to modify $\alpha_1, \ldots, \alpha_k$ to assignments $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_k$ such that

- they agree on common variables ($x \in X_i \cap X_j$ implies $\tilde{\alpha}_i(x) = \tilde{\alpha}_j(x)$), and
- there is a bijection between $\alpha_i(X_i)$ and $\tilde{\alpha}_i(X_i)$ that preserves constants, that is,
  - $\alpha_i(x) \in \mathit{Const}$ or $\tilde{\alpha}_i(x) \in \mathit{Const}$ implies $\tilde{\alpha}_i(x) = \alpha_i(x)$, and
  - $\alpha_i(x) = \alpha_i(x')$ if and only if $\tilde{\alpha}_i(x) = \tilde{\alpha}_i(x')$.

In particular, the assignments $\alpha_1, \ldots, \alpha_k$ can be "joined" to a single assignment $\tilde{\alpha}$. Formally, this is captured as follows:

5.34 Definition (compatible)
Let $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$. We say that $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are *compatible* if and only if there is an equivalence relation $\sim$ on $D := \bigcup_{i=1}^{k} \alpha_i(X_i)$ such that

1. for all $i, j \in \{1, \ldots, k\}$ and each $x \in X_i \cap X_j$, we have $\alpha_i(x) \sim \alpha_j(x)$,
2. for all $u, u' \in D$, if $u \sim u'$ and $u \in \mathit{Const}$, then $u = u'$, and
3. for each $i \in \{1, \ldots, k\}$ and all $x, x' \in X_i$, we have $\alpha_i(x) = \alpha_i(x')$ if and only if $\alpha_i(x) \sim \alpha_i(x')$.

Given $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$, we can use the following algorithm to check whether $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are compatible, and if so, to find an equivalence relation $\sim$ as in Definition 5.34.

5.35 Algorithm (Compatibility check)
Input: $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$
Output: a relation $\sim$ as in Definition 5.34 if $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are compatible; otherwise "not compatible"

1. Initialize $\sim$ to be $\{(u, u) \mid u \in \alpha_i(X_i) \text{ for some } i \in \{1, \ldots, k\}\}$.
2. For all $i, j \in \{1, \ldots, k\}$ and each $x \in X_i \cap X_j$, add $(\alpha_i(x), \alpha_j(x))$ to $\sim$.
3. For each $i \in \{1, \ldots, k\}$ and all $x, x' \in X_i$ with $\alpha_i(x) = \alpha_i(x')$, add $(\alpha_i(x), \alpha_i(x'))$ to $\sim$.
4. Update $\sim$ to be the symmetric and transitive closure of $\sim$.
5. If $\sim$ satisfies conditions 2 and 3 of Definition 5.34, then output $\sim$; otherwise output "not compatible".

5.36 Proposition.
Given $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$, Algorithm 5.35 correctly decides in time $O(\|T\|)$, where $\|T\|$ is the size of T, whether $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are compatible, and if so, outputs an equivalence relation $\sim$ on $D := \bigcup_{i=1}^{k} \alpha_i(X_i)$ that satisfies conditions 1–3 of Definition 5.34; in fact, $\sim$ is the smallest such equivalence relation (in terms of set inclusion).

Proof. Since k and $X_1, \ldots, X_k$ are constant, it should be clear that each of the steps 1–5 can be accomplished in constant time, after building the necessary data structures from the input in time $O(\|T\|)$ (note that for each $i \in \{1, \ldots, k\}$, we have $\|T_i\| \leq \|T\|$, so that the length of the input is in $O(\|T\|)$).

Let us now show that the algorithm is correct. Assume first that the algorithm outputs a relation $\sim$. By the construction of the algorithm, $\sim$ is an equivalence relation on D that satisfies conditions 1–3 of Definition 5.34. In particular, $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are compatible.

Before we prove the other direction, let us prove that, if $\sim^*$ is an arbitrary equivalence relation on D that satisfies conditions 1–3 of Definition 5.34, then the relation $\sim$ produced in steps 1–4 of the algorithm satisfies $\sim \subseteq \sim^*$. Let $\sim^*$ be an equivalence relation on D that satisfies conditions 1–3 of Definition 5.34. Since $\sim^*$ satisfies condition 1, $\sim^*$ contains all pairs that are put into $\sim$ in step 2 of the algorithm. Furthermore, since $\sim^*$ satisfies condition 3, $\sim^*$ contains all pairs that are put into $\sim$ in step 3 of the algorithm. Finally, since $\sim^*$ is an equivalence relation, and thus reflexive, symmetric and transitive, $\sim^*$ contains all pairs that are put into $\sim$ in steps 1 and 4 of the algorithm. Consequently, we have $\sim \subseteq \sim^*$.

This, in particular, shows that if the algorithm outputs a relation $\sim$, then this is the smallest equivalence relation on D (in terms of set inclusion) that satisfies conditions 1–3 of Definition 5.34.

Now assume that $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are compatible. Then there is an equivalence relation $\sim^*$ that satisfies conditions 1–3 of Definition 5.34. Furthermore, the relation $\sim$ produced in steps 1–4 of the algorithm is an equivalence relation on D such that

- $\sim$ satisfies condition 1 of Definition 5.34,
- for each $i \in \{1, \ldots, k\}$ and all $x, x' \in X_i$, $\alpha_i(x) = \alpha_i(x')$ implies $\alpha_i(x) \sim \alpha_i(x')$, and
- $\sim \subseteq \sim^*$, as shown above.

Since $\sim^*$ satisfies conditions 2 and 3 of Definition 5.34, this implies that $\sim$ satisfies conditions 2 and 3 of Definition 5.34, and therefore, $\sim$ is the output of the algorithm. □

Let $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$ be compatible. We define the join of $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ in terms of the following algorithm:

5.37 Algorithm (Join)
Input: $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$, which are compatible
Output: a tuple $(\tilde{T}, \tilde{\alpha})$

1. Let $\sim$ be the output of Algorithm 5.35 on input $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$.
2. Pick some linear order $\preceq$ on the elements of $D := \bigcup_{i=1}^{k} \alpha_i(X_i)$,[1] and for each $u \in D$, let $\hat{u}$ be the minimal element in $[u] := \{u' \in D \mid u' \sim u\}$ with respect to $\preceq$.
3. For each $i \in \{1, \ldots, k\}$, pick a mapping $r_i \colon \mathrm{dom}(T_i) \to \mathit{Dom}$ such that for each $u \in \mathrm{dom}(T_i)$,
$$r_i(u) := \begin{cases} \hat{u}, & \text{if } u \in \alpha_i(X_i) \\ u, & \text{otherwise.} \end{cases}$$
4. Output $(\tilde{T}, \tilde{\alpha})$, where $\tilde{T} := \bigcup_{i=1}^{k} r_i(T_i)$, and $\tilde{\alpha} \colon \bigcup_{i=1}^{k} X_i \to \mathit{Dom}$ is such that for each $x \in \bigcup_{i=1}^{k} X_i$,
$$\tilde{\alpha}(x) := \begin{cases} r_1(\alpha_1(x)), & \text{if } x \in X_1 \\ \quad \vdots & \\ r_k(\alpha_k(x)), & \text{if } x \in X_k. \end{cases}$$
Note that $\tilde{\alpha}$ is well-defined by the construction of $\sim$ and $r_1, \ldots, r_k$: if $x \in X_i \cap X_j$, then $\alpha_i(x) \sim \alpha_j(x)$, and thus, $r_i(\alpha_i(x)) = r_j(\alpha_j(x))$.

The crucial property of Algorithm 5.37 for the correctness of our query evaluation algorithm is:

5.38 Proposition.
Given compatible $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$, Algorithm 5.37 runs in time $O(\|T\|)$, where $\|T\|$ is the size of T, and outputs a tuple $(\tilde{T}, \tilde{\alpha})$ with the following properties:

[1] For definiteness, we can generate $\preceq$ as follows. Initialize $\preceq$ to be the empty relation. For increasing $i = 1, 2, \ldots, k$, consider the variables $x \in X_i$ in some predefined fixed order, and if $u := \alpha_i(x)$ does not already occur in $\preceq$, add u as the new maximal element to $\preceq$. This takes constant time since k and $X_1, \ldots, X_k$ are fixed. Using such an ordering guarantees that the algorithm produces a deterministic output, which, however, is not important for the following construction.

1. *For each* $i \in \{1, \dots, k\}$, *there is a mapping* $r_i \colon \operatorname{dom}(T_i) \to Dom$ *such that the following is true:*

   (a) $\tilde{T} = \bigcup_{i=1}^{k} r_i(T_i)$, *and* $\tilde{\alpha} \colon \bigcup_{i=1}^{k} X_i \to Dom$ *is such that for each* $x \in \bigcup_{i=1}^{k} X_i$,

   $$\tilde{\alpha}(x) \;=\; \begin{cases} r_1(\alpha_1(x)), & \text{if } x \in X_1 \\ \quad\vdots & \\ r_k(\alpha_k(x)), & \text{if } x \in X_k. \end{cases}$$

   (b) *For each* $i \in \{1, \dots, k\}$, *we have* $r_i(c) = c$ *for each* $c \in \operatorname{const}(T_i)$, *and* $r_i(\bot) \in Null$ *for each* $\bot \in \operatorname{nulls}(T_i)$.

   (c) *Let* $\sim$ *be the output of Algorithm 5.35 on input* $(T_1, \alpha_1), \dots, (T_k, \alpha_k)$. *Then for all* $i, j \in \{1, \dots, k\}$, *for each* $u \in \operatorname{dom}(T_i)$ *and for each* $u' \in \operatorname{dom}(T_j)$, *we have*

   $$r_i(u) = r_j(u') \iff u = u', \text{ or } u \in \alpha_i(X_i),\ u' \in \alpha_j(X_j) \text{ and } u \sim u'.$$

   (d) *For each equivalence relation* $\sim^*$ *on* $D := \bigcup_{i=1}^{k} \alpha_i(X_i)$ *that satisfies conditions 1–3 of Definition 5.34, for all* $i, j \in \{1, \dots, k\}$, *for each* $u \in \operatorname{dom}(T_i)$ *and for each* $u' \in \operatorname{dom}(T_j)$, *we have*

   $$r_i(u) = r_j(u') \implies \begin{cases} u \sim^* u', & \text{if } u \in \alpha_i(X_i),\ u' \in \alpha_j(X_j), \\ u = u' & \text{otherwise.} \end{cases} \tag{5.24}$$

   (e) *For each* $i \in \{1, \dots, k\}$, r_i *is injective.*

2. *For each* $i \in \{1, \dots, k\}$, *we have* $\tilde{\alpha}(\bar{x}_i) \in R_i^{\tilde{T}}$.

Proof. Let us first show that the algorithm runs in time $O(\|T\|)$. By Proposition 5.36, step 1 needs time at most $O(\|T\|)$. Since k and $X_1, \dots, X_k$ are fixed, step 2 needs only constant time. Furthermore, since k is constant and $\|T_i\| \leq \|T\|$ for each $i \in \{1, \dots, k\}$, step 3 needs time at most $O(\|T\|)$. Finally, step 4 can be accomplished in time $O(\|T\|)$. Altogether, the algorithm terminates after at most $O(\|T\|)$ steps.

Let $(\tilde{T}, \tilde{\alpha})$ be the output of the algorithm. We prove that $(\tilde{T}, \tilde{\alpha})$ has properties 1 and 2.

Ad 1: Let $r_1, \dots, r_k$ be the mappings chosen in step 3 of the algorithm. We prove that (a)–(e) are satisfied for $r_1, \dots, r_k$.

Ad 1(a): This follows directly from the construction of Algorithm 5.37.

Ad 1(b): Let $i \in \{1, \ldots, k\}$. To see that $r_i(c) = c$ for each $c \in \text{const}(T_i)$, let $c \in \text{const}(T_i)$. If $c \notin \alpha_i(X_i)$, then by the construction of r_i we have $r_i(c) = c$. Otherwise, if $c \in \alpha_i(X_i)$, there is some $x \in X_i$ with $\alpha_i(x) = c$, so that by the construction of r_i we have $c = \alpha_i(x) \sim r_i(\alpha_i(x)) = r_i(c)$. Condition 2 of Definition 5.34 then yields $r_i(c) = c$.

To see that $r_i(\bot) \in \mathit{Null}$ for each $\bot \in \text{nulls}(T_i)$, let $\bot \in \text{nulls}(T_i)$. As above, if $\bot \notin \alpha_i(X_i)$, then $r_i(\bot) = \bot \in \mathit{Null}$. Otherwise, $\bot \sim r_i(\bot)$, so that by condition 2 of Definition 5.34, $r_i(\bot) \in \mathit{Null}$.

Ad 1(c): Let $i, j \in \{1, \ldots, k\}$, $u \in \text{dom}(T_i)$ and $u' \in \text{dom}(T_j)$. We show that

$$r_i(u) = r_j(u') \iff u = u', \text{ or } u \in \alpha_i(X_i),\ u' \in \alpha_j(X_j) \text{ and } u \sim u'.$$

Assume first that $u = u'$. If $i = j$, we immediately have $r_i(u) = r_j(u')$. If $i \neq j$, then $\text{nulls}(T_i) \cap \text{nulls}(T_j) = \emptyset$ implies that u and u' are constants, and thus $r_i(u) = u = u' = r_j(u')$ by (b).

Assume next that $u \in \alpha_i(X_i)$, $u' \in \alpha_j(X_j)$ and $u \sim u'$. Then there are $x \in X_i$ and $x' \in X_j$ such that $\alpha_i(x) = u$, $\alpha_j(x') = u'$ and $\alpha_i(x) \sim \alpha_j(x')$. By the construction of r_i and r_j this immediately yields $r_i(u) = r_j(u')$.

Finally, assume that $r_i(u) = r_j(u')$. We distinguish the following cases:

- $u \notin \alpha_i(X_i) \cap \mathit{Null}$ and $u' \notin \alpha_j(X_j) \cap \mathit{Null}$:
  By the construction of r_i, r_j and by (b), we have $r_i(u) = u$ and $r_j(u') = u'$. Since $r_i(u) = r_j(u')$, this implies $u = u'$.

- $u \in \alpha_i(X_i) \cap \mathit{Null}$ or $u' \in \alpha_j(X_j) \cap \mathit{Null}$:
  By symmetry it suffices to deal with the case that $u \in \alpha_i(X_i) \cap \mathit{Null}$. So assume that $u \in \alpha_i(X_i) \cap \mathit{Null}$. By the construction of r_i, we then have

  $$u \ \sim \ r_i(u) \ = \ r_j(u').$$

  We claim that $u' \in \alpha_j(X_j)$. Suppose, to the contrary, that $u' \notin \alpha_j(X_j)$. By the construction of r_j, we have

  $$u' \ = \ r_j(u') \ = \ r_i(u) \ \sim \ u.$$

  Note that $u \in \alpha_i(X_i)$, $r_i(u) = u'$ and the construction of r_i imply that $u' \in D := \bigcup_{p=1}^{k} \alpha_p(X_p)$. Pick $p \in \{1, \ldots, k\}$ and $x \in X_p$ with $\alpha_p(x) = u'$. By $u \in \mathit{Null}$, $r_i(u) = u'$ and (b), we have $u' \in \mathit{Null}$. Moreover, since $u' \in \text{nulls}(T_j)$, $u' = \alpha_p(X_p) \in \text{nulls}(T_p)$, and $\text{nulls}(T_j) \cap \text{nulls}(T_p) = \emptyset$ for $j \neq p$, we must have $p = j$. This, however, implies that $u' \in \alpha_j(X_j)$, which is a contradiction to our assumption that $u' \notin \alpha_j(X_j)$.

Thus, $u' \in \alpha_j(X_j)$. This implies, by the construction of r_j, that

$$u' \ \sim \ r_j(u') \ \sim \ u.$$

In particular, $u \in \alpha_i(X_i)$, $u' \in \alpha_j(X_j)$ and $u \sim u'$, as desired.

Ad 1(d): Let $\sim^*$ be an equivalence relation on D that satisfies conditions 1–3 of Definition 5.34. Furthermore, let $\sim$ be the output of Algorithm 5.35 on input $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$. By Proposition 5.36, $\sim$ is the smallest equivalence relation on D that satisfies conditions 1–3 of Definition 5.34. Thus, for all $i, j \in \{1, \ldots, k\}$, each $u \in \alpha_i(X_i)$ and each $u' \in \alpha_j(X_j)$, we have

$$u \sim u' \implies u \sim^* u'. \tag{5.25}$$

Now let $i, j \in \{1, \ldots, k\}$, $u \in \mathrm{dom}(T_i)$ and $u' \in \mathrm{dom}(T_j)$ such that $r_i(u) = r_j(u')$. By (c), we have $u = u'$, or $u \in \alpha_i(X_i)$, $u' \in \alpha_j(X_j)$ and $u \sim u'$. Thus, by (5.25), we have $u = u'$, or $u \in \alpha_i(X_i)$, $u' \in \alpha_j(X_j)$ and $u \sim^* u'$. In particular, this implies (5.24).

Ad 1(e): Let $i \in \{1, \ldots, k\}$, and let $u, u' \in \mathrm{dom}(T_i)$ be such that $r_i(u) = r_i(u')$. We have to show that $u = u'$. By (c), we have $u = u'$, or $u, u' \in \alpha_i(X_i)$ and $u \sim u'$. If $u = u'$, we are done. So assume that $u, u' \in \alpha_i(X_i)$ and $u \sim u'$. Let $x, x' \in X_i$ be such that $\alpha_i(x) = u$ and $\alpha_i(x') = u'$. Then $\alpha_i(x) \sim \alpha_i(x')$, and by condition 3 of Definition 5.34, we have $u = \alpha_i(x) = \alpha_i(x') = u'$, as desired.

Ad 2: This follows immediately from 1(a) and 1(b). □

We are now ready to present the main algorithm for $\textsc{CoreEval}_{\sigma,b}$:

5.39 Algorithm (Algorithm for $\textsc{CoreEval}_{\sigma,b}$)

Input: a naive table T over σ such that T is a core and each atom block of T is packed and contains at most b nulls; a tuple $\bar{t} \in \mathit{Const}^{|\bar{x}|}$

Output: "yes" if there is a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $\mathit{poss}(T)$ such that $\bigcup \mathcal{T} \models q(\bar{t})$; otherwise "no"

1. Transform q into the query $\tilde{q} = \exists \bar{y}\, \psi(\bar{y})$, where

$$\psi(\bar{y}) \ = \ \bigwedge_{i=1}^{k} R_i(\bar{x}_i) \ \wedge \ \bigwedge_{i=1}^{l} \neg Q_i(\bar{w}_i) \ \wedge \ \bigwedge_{i=1}^{m} \neg v_i = v_i',$$

2. Choose injective mappings $\rho_1, \ldots, \rho_s$ from $\mathrm{dom}(T) \cup C$ to Dom, where $\rho_i(c) = c$ for each $c \in \mathrm{const}(T) \cup C$, and $\mathrm{nulls}(\rho_i(T)) \cap \mathrm{nulls}(\rho_j(T)) = \emptyset$ for all distinct $i, j \in \{1, \ldots, s\}$.

3. Compute the sets $\mathcal{X}_1, \ldots, \mathcal{X}_k$.
4. For each $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$:
   (a) Run Algorithm 5.35 to check whether $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are compatible. If not, continue with next $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$.
   (b) Let $(\tilde{T}, \tilde{\alpha})$ be the result of Algorithm 5.37 on $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$.
   (c) If $\tilde{T} \cup \bigcup_{i=k+1}^{s} \rho_i(T)$ satisfies $\tilde{q}$, output "yes".
5. Output "no".

Let us now show that the algorithm runs in time polynomial in the size of T, and that the algorithm is correct.

5.40 Lemma (Running time).
Algorithm 5.39 runs in time polynomial in $\|T\|$, where $\|T\|$ is the size of T. More precisely, Algorithm 5.39 runs in time

$$O\left(\|T\|^{(k+2)\cdot(b+r+1)+2}\right),$$

where r is the maximum number of variables in a tuple $\bar{x}_i$ over all $i \in \{1, \ldots, k\}$.

Proof. The transformation from q to $\tilde{q}$ in step 1 can easily be done in constant time. Furthermore, it takes time at most $O(\|T\|)$ to pick the mappings $\rho_1, \ldots, \rho_s$ in step 2.

We now bound the running time for computing the sets $\mathcal{X}_i$ for each $i \in \{1, \ldots, k\}$ in step 3. Since k is constant, it suffices to bound the running time for computing a single set $\mathcal{X}_i$. Let $i \in \{1, \ldots, k\}$. To compute $\mathcal{X}_i$, we first compute a list $T_1, \ldots, T_p$ of all instances that occur in $min_C(T, B)$ for some atom block B of T. By Lemma 5.26, this needs time at most

$$O\left(\|T\|^{2b+4}\right),$$

since $|C|$ is constant. Starting with $\mathcal{X}_i$ initialized to be the empty set, we then add for each $j \in \{1, \ldots, p\}$ and each assignment $\alpha\colon X_i \to \mathrm{dom}(T_j)$ with $\alpha(\bar{x}_i) \in R_i^{T_j}$ the pair $(\rho_i(T_j), \rho_i \circ \alpha)$ to $\mathcal{X}_i$. Note that p is bounded by the number of atom blocks of T times the maximum number of mappings in $val_C(T, B)$ for some atom block B of T, which, in turn, is bounded by

$$|T| \cdot (\|T\| + |C|)^b \;=\; O\left(\|T\|^{b+1}\right).$$

Furthermore, for each $j \in \{1, \ldots, p\}$, there are at most

$$|\mathrm{dom}(T_j)|^{|X_i|} \;\leq\; \|T\|^r$$

assignments $\alpha\colon X_i \to \mathrm{dom}(T_j)$. Checking whether such an assignment α satisfies $\alpha(\bar{x}_i) \in R_i^{T_j}$ and computing the pair $(\rho_i(T_j), \rho_i \circ \alpha)$ requires time at most $O(\|T\|)$. Altogether, the worst case running time for computing the sets $\mathcal{X}_1, \ldots, \mathcal{X}_k$ is

$$O\left(\|T\|^{2b+4} + p \cdot \|T\|^r \cdot \|T\|\right) \;=\; O\left(\|T\|^{2b+4} + \|T\|^{b+r+2}\right) \;=\; O\left(\|T\|^{2b+r+4}\right),$$

which is polynomial in $\|T\|$, since b and r are constant.

Next we bound the time for step 4. Note that for each $i \in \{1, \ldots, k\}$, we have

$$|\mathcal{X}_i| \;\leq\; |T| \cdot (\|T\| + |C|)^b \cdot \|T\|^r \;=\; O\left(\|T\|^{b+r+1}\right).$$

Consequently, step 4 is repeated at most

$$O\left(\|T\|^{k\cdot(b+r+1)}\right)$$

times. By Proposition 5.36 and Proposition 5.38, steps 4(a) and 4(b) need time at most $O(\|T\|)$. Furthermore, computing $T^* := \tilde{T} \cup \bigcup_{i=k+1}^{s} \rho_i(T)$ and checking whether $T^* \models \tilde{q}$ in step 4(c) requires time at most $O(\|T\|)$. Thus, step 4 can be accomplished in time

$$O\left(\|T\|^{k\cdot(b+r+1)} \cdot \|T\|\right) \;=\; O\left(\|T\|^{k\cdot(b+r+1)+1}\right).$$

Altogether, Algorithm 5.39 runs in time

$$O\left(\|T\| + \|T\|^{2b+r+4} + \|T\|^{k\cdot(b+r+1)+1}\right) \;=\; O\left(\|T\|^{(k+2)\cdot(b+r+1)+2}\right),$$

which is polynomial in $\|T\|$. □

5.41 Lemma (Correctness).
Algorithm 5.39 is correct. That is, the following two statements are equivalent:

1. *Algorithm 5.39 outputs "yes" on input T and $\bar{t}$.*
2. *There is a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $poss(T)$ such that $\bigcup \mathcal{T} \models \tilde{q}$.*

Proof.

1 $\Longrightarrow$ 2: Assume that Algorithm 5.39 outputs "yes" on input T and $\bar{t}$. Then there are pairs $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$ that are compatible, and given $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$, Algorithm 5.37 outputs an instance $(\tilde{T}, \tilde{\alpha})$ such that the instance $T^* := \tilde{T} \cup \bigcup_{i=k+1}^{s} \rho_i(T)$ satisfies $\tilde{q}$. We construct a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $poss(T)$ such that $\bigcup \mathcal{T} \models \tilde{q}$.

By Proposition 5.38(1), we have $\tilde{T} = \bigcup_{i=1}^{k} r_i(T_i)$, where each r_i is an injective mapping from $\mathrm{dom}(T_i)$ to *Dom* with $r_i(c) = c$ for each constant $c \in \mathrm{const}(T_i)$. For each $i \in \{k+1, \ldots, s\}$, let $T_i := \rho_i(T)$, and let r_i be the identity mapping on $\mathrm{dom}(T_i)$. Then,

$$T^* \;=\; \bigcup_{i=1}^{s} r_i(T_i), \tag{5.26}$$

where each r_i is an injective mapping from $\mathrm{dom}(T_i)$ to *Dom* with $r_i(c) = c$ for each constant $c \in \mathrm{const}(T_i)$.

Note that each T_i is isomorphic to an instance $\hat{T}_i \in \mathit{min}_C(T)$. For $i \in \{1, \ldots, k\}$, this follows from Lemma 5.27 and the fact that T_i is isomorphic to an instance in $\mathit{min}_C(T, B)$ for some atom block B of T. For $i \in \{k+1, \ldots, s\}$, this follows from the fact that $T_i = \rho_i(T)$, that T is a core, and Proposition 5.20(2). For each $i \in \{1, \ldots, s\}$, let f_i be an isomorphism from $\hat{T}_i$ to T_i.

Let $v\colon \mathrm{dom}(T^*) \to \mathrm{const}(T^*) \cup (\mathit{Const} \setminus \mathrm{dom}(\tilde{q}))$ be an injective valuation of T^*. Let $i \in \{1, \ldots, s\}$. Then the composition $v_i := v \circ r_i \circ f_i$ of f_i, r_i and v is an injective valuation of $\hat{T}_i$. Together with $\hat{T}_i \in \mathit{min}_C(T)$ and Proposition 5.20(1), this implies that $v_i(\hat{T}_i)$ is a $\subseteq$-minimal instance in $\mathit{poss}(T)$. Thus, the set

$$\mathcal{T} \;:=\; \{v_i(\hat{T}_i) \mid 1 \leq i \leq s\}$$

is a nonempty finite set of $\subseteq$-minimal instances in $\mathit{poss}(T)$, and

$$\bigcup \mathcal{T} \;=\; \bigcup_{i=1}^{s} v_i(\hat{T}_i) \;=\; \bigcup_{i=1}^{s} v(r_i(T_i)) \;=\; v\left(\bigcup_{i=1}^{s} r_i(T_i)\right) \overset{(5.26)}{=} v(T^*).$$

Since $T^* \models \tilde{q}$, v is injective, and v maps nulls in T^* to constants that do not occur in $\tilde{q}$, we conclude that $\bigcup \mathcal{T} \models \tilde{q}$.

2 $\Longrightarrow$ 1: Assume that there is a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $\mathit{poss}(T)$ such that $\bigcup \mathcal{T} \models \tilde{q}$. We show that Algorithm 5.39 outputs "yes" on input T and $\bar{t}$.

Note that by $\bigcup \mathcal{T} \models \tilde{q}$ and the fact that $\tilde{q}$ is domain-independent, there is an assignment $\beta\colon \mathrm{free}(\psi) \to \mathrm{dom}(\bigcup \mathcal{T}) \cup C$ with

$$\bigcup \mathcal{T} \;\models\; \psi(\beta).$$

In particular, we can pick for each $i \in \{1, \ldots, k\}$ an instance $\hat{T}_i \in \mathcal{T}$ such that

$$\beta(\bar{x}_i) \;\in\; R_i^{\hat{T}_i}.$$

Note that there are at most $s - k$ values in $\beta(\mathrm{free}(\psi)) \setminus C$ that do not occur in $\beta(\bar{x}_i)$ for some $i \in \{1, \ldots, k\}$. Thus, we can fix instances $\hat{T}_{k+1}, \ldots, \hat{T}_s \in \mathcal{T}$ such

that each of the values in $\beta(\text{free}(\psi)) \setminus C$ that does not occur in $\beta(\bar{x}_i)$ for some $i \in \{1, \ldots, k\}$ belongs to $\text{dom}(\hat{T}_j)$ for some $j \in \{k+1, \ldots, s\}$. Note that β is an assignment of ψ with range in $\text{dom}(\hat{T}_1 \cup \cdots \cup \hat{T}_s) \cup C$, and

$$\bigcup_{i=1}^{s} \hat{T}_i \;\models\; \psi(\beta). \tag{5.27}$$

Let $i \in \{1, \ldots, k\}$. By Proposition 5.20(1), there is an instance $\tilde{T}_i \in \mathit{min}_C(T)$ and an injective valuation v_i of $\tilde{T}_i$ such that $v_i(\tilde{T}_i) = \hat{T}_i$, and $v_i^{-1}(c) = c$ for all $c \in \text{dom}(\hat{T}_i) \cap C$. In particular, the atom

$$A_i \;:=\; R_i(v_i^{-1}(\beta(\bar{x}_i)))$$

is an atom of $\tilde{T}_i$. By Lemma 5.31, there is an atom block B_i of T, an instance $T_i' \in \mathit{min}_C(T, B_i)$, an atom $A_i'' \in T_i'$ with $A_i'' \cong A_i$, and a homomorphism h_i' from T_i' to $\tilde{T}_i$ such that $h_i'(T_i') = \tilde{T}_i$ and $h_i'(A_i'') = A_i$. In particular, the composition $h_i := h_i' \circ \rho_i^{-1}$ is a homomorphism from $T_i := \rho_i(T_i')$ to $\tilde{T}_i$ such that

$$h_i(T_i) \;=\; \tilde{T}_i$$

and

$$h_i(A_i') \;=\; A_i,$$

where $A_i' := \rho_i(A_i'') \;\cong\; A_i$. Let α_i be an assignment for X_i such that

$$A_i' \;=\; R_i(\alpha_i(\bar{x}_i)).$$

Note that $(T_i, \alpha_i) \in \mathcal{X}_i$. In the following, we show that $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ can be joined, and extended by $s - k$ disjoint copies of T, so that the resulting instance satisfies $\tilde{q}$. In particular, this will show that Algorithm 5.39 outputs "yes" on input T and $\bar{t}$.

The following properties of the assignments α_i are crucial for showing this:

Claim 1.
Let $i, j \in \{1, \ldots, k\}$, let $x, x' \in X_i$ and let $x'' \in X_j$. Then,

1. $v_i(h_i(\alpha_i(\bar{x}_i))) = \beta(\bar{x}_i)$. *In particular,* $v_i(h_i(\alpha_i(x))) = \beta(x)$.
2. *If* $\beta(x) \in \text{const}(T) \cup C$, *then* $\alpha_i(x) = \beta(x)$.
3. $\alpha_i(x) = \alpha_i(x')$ *if and only if* $\beta(x) = \beta(x')$.
4. $\alpha_i(x) = \alpha_j(x'')$ *implies* $\beta(x) = \beta(x'')$.

Proof.

Ad 1: Recall that $h_i(A_i') = A_i$, that $A_i \in \tilde{T}_i$, and that v_i is injective on $\mathrm{dom}(\tilde{T}_i)$. In particular, we have $h_i(\alpha_i(\bar{x}_i)) = v_i^{-1}(\beta(\bar{x}_i))$. Applying v_i to both sides yields $v_i(h_i(\alpha_i(\bar{x}_i))) = \beta(\bar{x}_i)$.

Ad 2: Let $\beta(x) \in \mathrm{const}(T) \cup C$. By 1, we have

$$v_i(h_i(\alpha_i(x))) \;=\; \beta(x), \tag{5.28}$$

which implies

$$h_i(\alpha_i(x)) \;=\; \beta(x). \tag{5.29}$$

Indeed, if $\beta(x) \in \mathrm{const}(T)$, (5.29) follows immediately from (5.28), $\mathrm{const}(T) \subseteq \mathrm{const}(\tilde{T}_i)$, and the fact that v_i is an injective mapping from $\mathrm{dom}(\tilde{T}_i)$ that is the identity on constants. On the other hand, if $\beta(x) \in C$, then (5.29) follows immediately from (5.28), $\beta(x) \in \mathrm{dom}(\hat{T}_i)$, and the fact that $v_i^{-1}(c) = c$ for all $c \in \mathrm{dom}(\hat{T}_i) \cap C$.

Now (5.29) and $h_i(A_i') \cong A_i'$ imply that $\alpha_i(x)$ is a constant, and since h_i is the identity on constants, we have $\alpha_i(x) = \beta(x)$.

Ad 3: By 1, we have $v_i(h_i(\alpha_i(\bar{x}_i))) = \beta(\bar{x}_i)$. Recall also that v_i is injective, and that $h_i(A_i') = A_i \cong A_i'$, which implies that h_i is injective on $\alpha_i(X_i)$. Altogether, the composition $f_i := v_i \circ h_i$ is a bijection from $\alpha_i(X_i)$ to $\beta(X_i)$. This implies that $\alpha_i(x) = \alpha_i(x')$ if and only if $\beta(x) = \beta(x')$.

Ad 4: Let $\alpha_i(x) = \alpha_j(x'')$. If $i = j$, then $\beta(x) = \beta(x'')$ follows immediately from 3. So assume that $i \neq j$. Since $\alpha_i(x) \in \mathrm{dom}(T_i)$, $\alpha_j(x'') \in \mathrm{dom}(T_j)$ and $\mathrm{nulls}(T_i) \cap \mathrm{nulls}(T_j) = \emptyset$, $\alpha_i(x)$ and $\alpha_j(x'')$ must be constants. By 1 and the fact that the homomorphisms h_i, h_j as well as the valuations v_i, v_j are the identity on constants, we conclude that $\beta(x) = \alpha_i(x) = \alpha_j(x'') = \beta(x'')$. ⌟

We now show that $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are compatible, and thus "joinable". To this end, consider the relation

$$\sim \;:=\; \Big\{(\alpha_i(x), \alpha_j(x')) \mid i, j \in \{1, \ldots, k\},\, x \in X_i,\, x' \in X_j,\, \beta(x) = \beta(x')\Big\}$$

on $D := \bigcup_{i=1}^{k} \alpha_i(X_i)$.

Claim 2.

1. *For all $i, j \in \{1, \ldots, k\}$, for each $x \in X_i$ and for each $x' \in X_j$, we have*

$$\alpha_i(x) \sim \alpha_j(x') \iff \beta(x) = \beta(x').$$

2. The relation $\sim$ is an equivalence relation on D that satisfies conditions 1–3 of Definition 5.34.

Proof.

Ad 1: Let $i, j \in \{1, \ldots, k\}$, let $x \in X_i$ and let $x' \in X_j$. If $\beta(x) = \beta(x')$, then the definition of $\sim$ immediately yields $\alpha_i(x) \sim \alpha_j(x')$.

On the other hand, let $\alpha_i(x) \sim \alpha_j(x')$. Then there are $i', j' \in \{1, \ldots, k\}$, $y \in X_{i'}$ and $y' \in X_{j'}$ such that

$$\alpha_{i'}(y) = \alpha_i(x) \qquad \text{and} \qquad \alpha_{j'}(y') = \alpha_j(x'), \tag{5.30}$$

and

$$\beta(y) = \beta(y'). \tag{5.31}$$

By (5.30) and Claim 1(4), we have

$$\beta(y) = \beta(x) \qquad \text{and} \qquad \beta(y') = \beta(x'),$$

which by (5.31) yields $\beta(x) = \beta(x')$, as desired.

Ad 2: It is easy to verify that $\sim$ is an equivalence relation on D. Reflexivity and symmetry are clear, and transitivity is easy to show using 1.

It follows easily from 1 that $\sim$ satisfies condition 1 of Definition 5.34: Let $i, j \in \{1, \ldots, k\}$ and let $x \in X_i \cap X_j$. Since $\beta(x) = \beta(x)$, 1 then immediately yields $\alpha_i(x) \sim \alpha_j(x)$.

For proving that $\sim$ satisfies condition 2 of Definition 5.34, let $u, u' \in D$ be such that $u \sim u'$ and $u \in \mathit{Const}$. Since $u \sim u'$, there are $i, j \in \{1, \ldots, k\}$, $x \in X_i$ and $x' \in X_j$ such that $\alpha_i(x) = u$, $\alpha_j(x') = u'$, and

$$\beta(x) \;=\; \beta(x'). \tag{5.32}$$

By Claim 1(1), we have $v_i(h_i(\alpha_i(x))) = \beta(x)$. Since $\alpha_i(x)$ is a constant and h_i, v_i are the identity on constants, this implies that $\alpha_i(x) = \beta(x)$. In particular,

$$\beta(x') \overset{(5.32)}{=} \beta(x) \;=\; \alpha_i(x) \;\in\; \mathrm{const}(T_i) \;\subseteq\; \mathrm{const}(T) \cup C. \tag{5.33}$$

By Claim 1(2), this yields $\beta(x') = \alpha_j(x')$, and therefore,

$$u \;=\; \alpha_i(x) \overset{(5.33)}{=} \beta(x') \;=\; \alpha_j(x') \;=\; u',$$

as desired.

Finally, for proving that $\sim$ satisfies condition 3 of Definition 5.34, let $i \in \{1, \ldots, k\}$ and let $x, x' \in X_i$. Then,

$$\alpha_i(x) = \alpha_i(x') \overset{\text{Claim 1(3)}}{\Longleftrightarrow} \beta(x) = \beta(x') \overset{\text{Claim 2(1)}}{\Longleftrightarrow} \alpha_i(x) \sim \alpha_i(x'),$$

as desired. ⌟

By Claim 2, $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are compatible. Let (T_0, α_0) be the output of Algorithm 5.37 on input $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$. It remains to show that

$$T^* \ := T_0 \cup \bigcup_{i=k+1}^{s} T_i,$$

where $T_i := \rho_i(T)$ for each $i \in \{k+1, \ldots, s\}$, satisfies $\tilde{q}$. In other words, it remains to construct an assignment α for ψ such that $T^* \models \psi(\alpha)$. This assignment will be constructed with the help of:

Claim 3.
There is a homomorphism h_0 from T_0 to $\hat{T}_0 := \bigcup_{i=1}^{k} \hat{T}_i$ with $h_0(T_0) = \hat{T}_0$, and $h_0(\alpha_0(\bar{x}_i)) = \beta(\bar{x}_i)$ for each $i \in \{1, \ldots, k\}$.

Proof. Proposition 5.38(1) tells us that for each $i \in \{1, \ldots, k\}$, there is an injective mapping $r_i \colon \mathrm{dom}(T_i) \to \mathit{Dom}$ such that

$$T_0 \ = \ \bigcup_{i=1}^{k} r_i(T_i), \tag{5.34}$$

and $\alpha_0 \colon \bigcup_{i=1}^{k} X_i \to \mathit{Dom}$ is such that for each $x \in \bigcup_{i=1}^{k} X_i$,

$$\alpha_0(x) \ = \ \begin{cases} r_1(\alpha_1(x)), & \text{if } x \in X_1 \\ \quad \vdots & \\ r_k(\alpha_k(x)), & \text{if } x \in X_k. \end{cases} \tag{5.35}$$

Furthermore, for all $i, j \in \{1, \ldots, k\}$, for each $u \in \mathrm{dom}(T_i)$ and each $u' \in \mathrm{dom}(T_j)$, we have

$$r_i(u) = r_j(u') \implies u = u', \text{ or } u \in \alpha_i(X_i),\ u' \in \alpha_j(X_j) \text{ and } u \sim u'. \tag{5.36}$$

Let $h_0 \colon \mathrm{dom}(T_0) \to \mathrm{dom}(\hat{T}_0)$ be defined such that for each $i \in \{1, \ldots, k\}$ and each $u \in \mathrm{dom}(r_i(T_i))$, we have

$$h_0(u) \ = \ v_i(h_i(r_i^{-1}(u))). \tag{5.37}$$

We claim that h_0 is a homomorphism from T_0 to $\hat{T}_0$ with $h_0(T_0) = \hat{T}_0$, and that for each $i \in \{1, \ldots, k\}$ we have $h_0(\alpha_0(\bar{x}_i)) = \beta(\bar{x}_i)$.

Step 1: h_0 is well-defined.
To see that h_0 is well-defined, let us assume that $u \in \mathrm{dom}(r_i(T_i)) \cap \mathrm{dom}(r_j(T_j))$, where $i, j \in \{1, \ldots, k\}$ are distinct. Let $u_i := r_i^{-1}(u) \in \mathrm{dom}(T_i)$ and let $u_j := r_j^{-1}(u) \in \mathrm{dom}(T_j)$. We must show that

$$v_i(h_i(u_i)) \ = \ v_j(h_j(u_j)).$$

Since $r_i(u_i) = u = r_j(u_j)$, (5.36) implies that

1. $u_i = u_j$, or
2. $u_i \in \alpha_i(X_i)$, $u_j \in \alpha_j(X_j)$ and $u_i \sim u_j$.

If $u_i = u_j$, then both u_i and u_j are constants, since $\mathrm{nulls}(T_i) \cap \mathrm{nulls}(T_j) = \emptyset$ for $i \neq j$; therefore,

$$v_i(h_i(u_i)) \;=\; u_i \;=\; u_j \;=\; v_j(h_j(u_j)),$$

as desired. On the other hand, assume that there are $x_i \in X_i$ and $x_j \in X_j$ such that $u_i = \alpha_i(x_i)$, $u_j = \alpha_j(x_j)$ and $\alpha_i(x_i) \sim \alpha_j(x_j)$. Then Claim 2(1) implies $\beta(x_i) = \beta(x_j)$. By Claim 1(1), we thus have

$$v_i(h_i(\alpha_i(x_i))) \;=\; \beta(x_i) \;=\; \beta(x_j) \;=\; v_j(h_j(\alpha_j(x_j))),$$

which implies $v_i(h_i(u_i)) = v_j(h_j(u_j))$, as desired. Altogether, this shows that h_0 is well-defined.

Step 2: h_0 is a homomorphism from T_0 to $\hat{T}_0$ with $h_0(T_0) = \hat{T}_0$.
First note that for each $i \in \{1, \ldots, k\}$, we have

$$h_0(r_i(T_i)) \overset{(5.37)}{=} v_i(h_i(T_i)) \;=\; \hat{T}_i.$$

Hence,

$$h_0(T_0) \overset{(5.34)}{=} h_0\left(\bigcup_{i=1}^{k} r_i(T_i)\right) \;=\; \bigcup_{i=1}^{k} h_0(r_i(T_i)) \;=\; \bigcup_{i=1}^{k} \hat{T}_i \;=\; \hat{T}_0.$$

Step 3: For each $i \in \{1, \ldots, k\}$, we have $h_0(\alpha_0(\bar{x}_i)) = \beta(\bar{x}_i)$.
This is easy:

$$h_0(\alpha_0(\bar{x}_i)) \overset{(5.35)}{=} h_0(r_i(\alpha_i(\bar{x}_i))) \overset{(5.37)}{=} v_i(h_i(\alpha_i(\bar{x}_i))) \overset{\text{Claim 1(1)}}{=} \beta(\bar{x}_i).$$

Altogether, the proof of Claim 3 is complete. ⌟

Let h_0 be a homomorphism as in Claim 3. It is easy to extend h_0 to a mapping h on $\mathrm{dom}(T^*) \cup C$ with the following properties:

1. $h(T_0) = h_0(T_0) = \bigcup_{i=1}^{k} \hat{T}_i$,
2. $h(T_i) = \hat{T}_i$ for each $i \in \{k+1, \ldots, s\}$, and
3. $h(c) = c$ for each $c \in C$.

Note that the second condition can be satisfied, since for all $i \in \{k+1, \ldots, s\}$ and for all $j \in \{1, \ldots, s\}$ with $j \neq i$, we have $\mathrm{nulls}(T_i) \cap \mathrm{nulls}(T_j) = \emptyset$, $T_i \cong T$ and $\hat{T}_i \in \mathit{poss}(T)$. Note also that

$$h(T^*) \;=\; \bigcup_{i=1}^{s} \hat{T}_i. \tag{5.38}$$

Let us also extend α_0 to an assignment α for $\mathrm{free}(\psi)$ such that

$$h(\alpha(y)) \;=\; \beta(y) \quad \text{for each } y \in \mathrm{free}(\psi). \tag{5.39}$$

Note that (5.39) holds for all variables y that occur in $\bar{x}_i$ for some $i \in \{1, \ldots, k\}$, because h is an extension of h_0, and α is an extension of α_0. For each variable $y \in \mathrm{free}(\psi)$ that does not occur in $\bar{x}_i$ for some $i \in \{1, \ldots, k\}$, we pick an arbitrary value $u \in \mathrm{const}(T^*) \cup C$ with $h(u) = \beta(y)$ and let $\alpha(y) := u$. Note that such a value u always exists. First recall that the range of β is in $\mathrm{dom}(\bigcup_{i=1}^{s} \hat{T}_i) \cup C$. If $\beta(y) \in \mathrm{dom}(\bigcup_{i=1}^{s} \hat{T}_i)$, then by (5.38) there is some $u \in \mathrm{dom}(T^*)$ with $h(u) = \beta(y)$. On the other hand, if $\beta(y) \in C$, then $h(\beta(y)) = \beta(y)$, because h is the identity on constants, so that we can choose $u = \beta(y)$.

We are finally ready to show that $T^* \models \psi(\alpha)$. First note that by Proposition 5.38(2), we have $\alpha_0(\bar{x}_i) \in R_i^{T_0}$ for each $i \in \{1, \ldots, k\}$; since $T_0 \subseteq T^*$ and α extends α_0, this implies

$$\alpha(\bar{x}_i) \in R_i^{T^*} \quad \text{for each } i \in \{1, \ldots, k\}. \tag{5.40}$$

Furthermore, we have

$$\alpha(\bar{w}_i) \notin Q_i^{T^*} \quad \text{for each } i \in \{1, \ldots, l\}. \tag{5.41}$$

Otherwise, if there is some $i \in \{1, \ldots, l\}$ with $\alpha(\bar{w}_i) \in Q_i^{T^*}$, then by (5.38) and (5.39), we have

$$\beta(\bar{w}_i) \in Q_i^{\bigcup_{i=1}^{s} \hat{T}_i},$$

which is impossible by (5.27). Finally, we have

$$\alpha(v_i) \neq \alpha(v_i') \quad \text{for each } i \in \{1, \ldots, m\}. \tag{5.42}$$

Indeed, let $i \in \{1, \ldots, m\}$. By (5.39), we have $h(\alpha(v_i)) = \beta(v_i)$ and $h(\alpha(v_i')) = \beta(v_i')$. On the other hand, (5.27) implies that $\beta(v_i) \neq \beta(v_i')$, so that $\alpha(v_i)$ and $\alpha(v_i')$ must be distinct. Altogether, (5.40)–(5.42) imply that $T^* \models \psi(\alpha)$.

In particular, Algorithm 5.39 outputs "yes" on input T and $\bar{t}$. □

Note that this completes the proof of Lemma 5.33.

5.4.4 Putting Things Together

In this section, we give a proof of Theorem 5.17.

Let $M = (\sigma_{\mathrm{s}}, \sigma_{\mathrm{t}}, \Sigma)$ be a schema mapping, where Σ consists of packed st-tgds, and let q be a universal query over σ_{t}. We show that there is a polynomial time algorithm that takes a target instance T for M and a tuple $\bar{t} \in \mathit{Const}^{\mathrm{ar}(q)}$ as input, and if $T \cong \mathrm{Core}(M, S)$ for some source instance S for M, then it decides whether $\bar{t} \in \mathit{cert}_{\mathrm{GCWA}^*}(q, M, S)$.

As shown in Section 5.4.1, we have $\bar{t} \notin \mathit{cert}_{\mathrm{GCWA}^*}(q, M, S)$ if and only if there is a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $\mathit{poss}(T)$ with $\bigcup \mathcal{T} \models \neg q(\bar{t})$. Now observe that $\neg q$ is logically equivalent to a query $\bar{q}$ of the form

$$\bar{q}(\bar{x}) \;=\; \bigvee_{i=1}^{m} q_i(\bar{x}),$$

where each q_i is an existential query of the form

$$q_i(\bar{x}) \;=\; \exists \bar{y}_i \bigwedge_{j=1}^{n_i} \varphi_{i,j},$$

and each $\varphi_{i,j}$ is an atomic FO formula or the negation of an atomic FO formula. Indeed, since q is a universal query, we have $\neg q \equiv \exists \bar{y}\, \varphi(\bar{x}, \bar{y})$, where φ is quantifier-free. By transforming φ into "disjunctive normal form", we obtain a query of the form $\exists \bar{y} \bigvee_{i=1}^{m} \bigwedge_{j=1}^{n_i} \varphi_{i,j}$, where each $\varphi_{i,j}$ is an atomic FO formula or the negation of an atomic FO formula. By moving existential quantifiers inwards, we finally obtain $\bar{q}$. It remains therefore to decide whether there is some $i \in \{1, \ldots, m\}$ and a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $\mathit{poss}(T)$ such that $\bigcup \mathcal{T} \models q_i(\bar{t})$.

We are now ready to describe the algorithm. By Lemma 5.23, there is a positive integer b such that for each source instance S for M and each atom block B of $\mathrm{Core}(M, S)$, we have $|\mathrm{nulls}(B)| \leq b$. Fix such a constant b. Furthermore, by Proposition 5.30, each atom block of $\mathrm{Core}(M, S)$ is packed. Given a target instance T for M and a tuple $\bar{t} \in \mathit{Const}^{\mathrm{ar}(q)}$ as input, the algorithm then proceeds as follows.

1. Determine the atom blocks of T and check whether for each atom block B of T, we have $|\mathrm{nulls}(B)| \leq b$ and that B is packed. If not, reject the input.

2. Check whether T is a core. If not, reject the input.

3. For each $i \in \{1, \ldots, m\}$:

(a) Check whether there is a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $poss(T)$ such that $\bigcup \mathcal{T} \models q_i(\bar{t})$.

(b) If there is such a set $\mathcal{T}$, then reject the input.

4. Accept the input.

Step 1 can easily be accomplished in time polynomial in the size of T. Furthermore, step 2 can be accomplished in time polynomial in the size of T, using the algorithm presented in the proof of Lemma 2.31: the output of this algorithm is T if and only if T is a core. Finally, Step 3 can be accomplished in time polynomial in the size of T by Lemma 5.33. Consequently, the whole algorithm runs in time polynomial in the size of T.

Altogether, the proof of Theorem 5.17 is complete.

5.4.5 Proof of Proposition 5.13

Finally, let us give a proof of Proposition 5.13. Let $M = (\sigma_s, \sigma_t, \Sigma)$ be a schema mapping, where Σ consists of st-tgds, and let q be a universal query over σ_t. As in Section 5.4.4, we can assume that $\neg q$ is logically equivalent to a query $\bar{q}$ of the form

$$\bar{q}(\bar{x}) \;=\; \bigvee_{i=1}^{m} q_i(\bar{x}),$$

where each q_i is an existential query of the form

$$q_i(\bar{x}) \;=\; \exists \bar{y}_i \bigwedge_{j=1}^{n_i} \varphi_{i,j},$$

and each $\varphi_{i,j}$ is an atomic FO formula or the negation of an atomic FO formula.

Let S be a source instance for M, and let $\bar{t} \in Const^{\mathrm{ar}(q)}$. As shown in Section 5.4.1, we have $\bar{t} \notin cert_{\mathrm{GCWA}^*}(q, M, S)$ if and only if there is a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $poss(\mathrm{Core}(M, S))$ with $\bigcup \mathcal{T} \models \neg q(\bar{t})$. Hence, on input S and $\bar{t}$, a nondeterministic Turing machine can decide whether $\bar{t} \notin cert_{\mathrm{GCWA}^*}(q, M, S)$ by computing $\mathrm{Core}(M, S)$, and by determining for each $i \in \{1, \ldots, m\}$, whether there is a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $poss(\mathrm{Core}(M, S))$ with $\bigcup \mathcal{T} \models q_i(\bar{t})$. If so, it accepts the input, and otherwise, it rejects it.

By Theorem 2.28, $\mathrm{Core}(M, S)$ can be computed in time polynomial in the size of S (for fixed M).

In order to check whether there is a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $poss(\mathrm{Core}(M, S))$ with $\bigcup \mathcal{T} \models q_i(\bar{t})$, it suffices to "guess" a set $\mathcal{T}$ of at most

$$s \;:=\; n_i \cdot \max\{\mathrm{ar}(R) \mid R \in \sigma_t\}$$

instances in *poss*(Core(M, S)), and to check whether $\bigcup\mathcal{T} \models q_i(\bar{t})$. Indeed, let $\mathcal{T}$ be a set of $\subseteq$-minimal instances in *poss*(Core(M, S)) with $\bigcup\mathcal{T} \models q_i(\bar{t})$. Then there is an assignment α for the variables in $\bar{x}$ and $\bar{y}_i$ such that $\alpha(\bar{x}) = \bar{t}$ and $\bigcup\mathcal{T} \models \varphi_{i,j}(\alpha)$ for each $j \in \{1, \ldots, n_i\}$. Without loss of generality, assume that $\varphi_{i,1}, \ldots, \varphi_{i,k}$ for $0 \leq k \leq n_i$ are all the relational atomic FO formulas in q_i. For each $j \in \{1, \ldots, k\}$, there is an instance $T_j \in \mathcal{T}$ with $T_j \models \varphi_{i,j}(\alpha)$. Let $\mathcal{T}_0' := \{T_1, \ldots, T_k\} \subseteq \mathcal{T}$. Then $\bigcup\mathcal{T}_0' \models \varphi_{i,j}(\alpha)$ for each $j \in \{1, \ldots, k\}$. To obtain a set $\mathcal{T}_0 \subseteq \mathcal{T}$ that satisfies $\bigcup\mathcal{T}_0 \models q_i(\bar{t})$, we extend $\mathcal{T}_0'$ as follows. Let $j \in \{k+1, \ldots, n_i\}$. Then there are at most $\max\{\mathrm{ar}(R) \mid R \in \sigma_{\mathrm{t}}\}$ values that occur in $\varphi_{i,j}(\alpha)$. In particular, we can pick $\max\{\mathrm{ar}(R) \mid R \in \sigma_{\mathrm{t}}\}$ instances from $\mathcal{T}$ that contain all these values. Add those instances to $\mathcal{T}_0'$. The resulting set $\mathcal{T}_0$ is a subset of $\mathcal{T}$, and satisfies $\bigcup\mathcal{T}_0 \models q_i(\bar{t})$, since $\bigcup\mathcal{T}_0 \models \varphi_{i,j}(\alpha)$ for each $j \in \{1, \ldots, k\}$. Furthermore, $\mathcal{T}_0$ contains at most s instances.

Note also that to find a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in *poss*(Core(M, S)) with $|\mathcal{T}| \leq s$ and $\bigcup\mathcal{T} \models q_i(\bar{t})$, it suffices to consider valuations v of Core(M, S) with range in C, where C contains all constants in Core(M, S), all constants in q_i, all constants in $\bar{t}$, and all constants in $\{c_1, \ldots, c_{s \cdot k}\}$, where k is the number of nulls in Core(M, S), and $c_1, \ldots, c_{s \cdot k}$ is a sequence of pairwise distinct constants that do not occur in Core(M, S), q_i, and $\bar{t}$.

Finally, it is easy for a Turing machine to check whether a given instance $T \in$ *poss*(Core(M, S)) is $\subseteq$-minimal. For each atom $A \in T$, it just has to check that the instance $T \setminus \{A\}$ is no solution for S under M.

Altogether, given a source instance S for M, and a tuple $\bar{t} \in \mathit{Const}^{\mathrm{ar}(q)}$, a nondeterministic Turing machine can check whether $\bar{t} \notin \mathit{cert}_{\mathrm{GCWA}^*}(q, M, S)$. This shows that Eval$(M, q)$ belongs to co-NP, and in particular this proves Proposition 5.13. □

Membership in co-NP seems to be only a very rough upper bound on the complexity of computing the GCWA*-answers to universal queries under schema mappings defined by st-tgds. I conjecture that the complexity of this problem is in polynomial time. In fact, it seems that all what has to be done is to prove Lemma 5.31 for the case that the atom blocks of T are not packed. This then would yield the analogue of Theorem 5.15 for the case of schema mappings defined by general st-tgds. It is conceivable that Theorem 5.15 even holds for schema mappings defined by general st-tgds and egds.

6 Conclusion

Query answering is an important issue in relational data exchange [Fagin et al., 2005a, Kolaitis, 2005, Barceló, 2009]. In this thesis, several semantics for query answering in relational data exchange, and the complexity of evaluating queries with respect to these semantics have been discussed. This concluding chapter finally addresses the following two questions:

1. Which semantics are appropriate for answering which queries? And what are the relations between the different semantics?

2. What kind of solution should one compute so that queries can be evaluated under the desired semantics using this kind of solution? In other words, what are "good" solutions?

We deal with the first question in Section 6.1, and with the second one in Section 6.2. The last section discusses some open questions, and possible extensions of this thesis' results.

6.1 Which Semantics are Appropriate for Answering Which Queries?

The semantics for query answering in relational data exchange that have been discussed throughout this thesis can be grouped into two categories: semantics for answering *monotonic* queries, and semantics for answering *non-monotonic* queries.

For answering *monotonic queries*, the following facts strongly suggest that the *certain answers semantics* proposed by Fagin et al. [2005a] intuitively leads to the desired results. First, the certain answers semantics has been widely used for answering queries preserved under homomorphisms (see, e.g., Kolaitis [2005] and Barceló [2009]), an important subclass of monotonic queries. Many examples have shown that the certain answers semantics leads to the desired results with respect to those examples. Second, many of the other semantics considered in this thesis coincide with the certain answers semantics on monotonic queries. For the GCWA*-answers semantics, this is shown by Proposition 5.10, and analogous proofs show this for the semantics from Chapter 4. For all but two of the semantics from Chapter 3, this holds at least for queries preserved under homomorphisms (cf., e.g., Corollary 3.40). Third, consider a schema mapping M, a source instance S for M, and a monotonic query q. Then one easily verifies that

the certain answers to q on M and S can be defined as the certain answers to q on the $\subseteq$-minimal solutions for S under M. For query answering, it seems to be reasonable to take into account at least the $\subseteq$-minimal solutions for S under M. Thus, intuitively, the certain answers to q on M and S lead to the desired set of answers to q. Finally, monotonic queries are intuitively *unable* to ask whether a relation of an instance *does not* contain a tuple; they can only ask whether a tuple *does* belong to the particular relation. Thus, implicit information in the sense described in Section 1.3.2 intuitively does not influence the certain answers to monotonic queries.

Turning to *non-monotonic queries*, there seems to be no general query answering semantics that is appropriate for all situations. Rather, the answer to the question of the appropriate semantics depends on the particular application and one's point of view. If schema mappings are interpreted "as they are"—without any additional information—then the certain answers semantics intuitively leads to the desired results (cf., Section 1.3.2). However, as argued in Section 1.3.2, it is often natural to interpret schema mappings with additional *implicit information*. In this case, the certain answers semantics can yield answers that intuitively do not seem to be accurate.

For answering queries with respect to implicit information, several semantics have been discussed throughout this thesis:

- On the one hand, we have the CWA-solution-based semantics, the mixed world-solution-based semantics, and the endomorphic images semantics (cf., Chapter 3). These semantics reflect the operational point of view on tgds and egds. That is, tgds are considered as rules to derive new atoms, whereas egds are considered as rules for identifying two values.

- On the other hand, we have the GCWA*-answers semantics (cf., Chapter 5) that reflects a semantic point of view on the schema mapping. That is, schema mappings that are defined by logically equivalent sets of formulas are considered as descriptions of one and the same data translation. In particular, on logically equivalent schema mappings, queries have identical results. As shown in Section 3.6, this is not the case for the above-mentioned semantics. Furthermore, the GCWA*-answers semantics reflects the standard semantics of FO quantifiers, which makes it more intuitive if one is used to, for example, existential quantifiers expressing that there are one, two, three or more elements that satisfy a given property. Finally, the GCWA*-answers semantics is defined for all schema mappings, rather than schema mappings defined by tgds and egds.

For schema mappings defined by full tgds and egds, it is easy to verify that all of the above-mentioned semantics for query answering with implicit information

lead to the same query answers. However, for more general schema mappings, query answers may vary considerably.

The relations between the query answering semantics considered in this thesis are shown in Figure 6.1, and follow easily from the definitions and results of this thesis. In Figure 6.1, an arc from semantics sem_1 to semantics sem_2 indicates that for all schema mappings M for which sem_1 and sem_2 are defined, for all source instances S for M, and for all queries q over M's target schema, we have $sem_1(q, M, S) \subseteq sem_2(q, M, S)$. Furthermore, the semantics $cert^{\exists op}_{\text{mixed}}$ shown in Figure 6.1 is the following mixed world solution-based semantics: For a schema mapping M defined by a set Σ of st-tgds, we let Λ_M be the annotation for M that annotates each position in the head of a st-tgd of Σ, where an existentially quantified variable occurs, with *op*, and all other positions with *cl*. Then for all source instances S for M, and all queries q over M's target schema, $cert^{\exists op}_{\text{mixed}}(q, M, S)$ denotes the certain answers to q on the set $\bigcup_{(T,\Lambda_T)\in sol_{\text{mixed}}(M,\Lambda_M,S)} poss(T, \Lambda_T)$.

6.2 What Kind of Solution Should one Compute?

Assuming an appropriate semantics for query answering in a particular situation is identified, what kind of solution should one compute so that queries can be evaluated under the desired semantics using this kind of solution? In other words, what are "good" solutions? The answer to this question depends, as the reader might have guessed, on the particular semantics, the schema mapping and the query to be answered.

In a lot of cases, *universal solutions* are good for computing query answers. Under almost all of the semantics considered in this thesis, the answers to *queries preserved under homomorphisms* can be computed directly from an arbitrary universal solution. For example, as shown by Fagin et al. [2005a] (see also Section 2.1), the certain answers to queries preserved under homomorphisms can be computed directly from an arbitrary universal solution. More precisely, given a schema mapping M, a source instance S for M, a universal solution T for S under M, and a homomorphism-preserved query q over M's target schema, the certain answers to q on M and S are precisely all tuples that belong to $q(T)$, and contain only constants. Since most of the semantics considered in this thesis coincide with the certain answers semantics on queries preserved under homomorphisms (cf., Section 6.1), this result carries over to those semantics. However, we have seen that universal solutions can be good even in cases where queries are not preserved under homomorphisms. One such case is computing the *certain answers* to *unions of conjunctive queries with at most one inequality per disjunct* with respect to *weakly acyclic schema mappings*

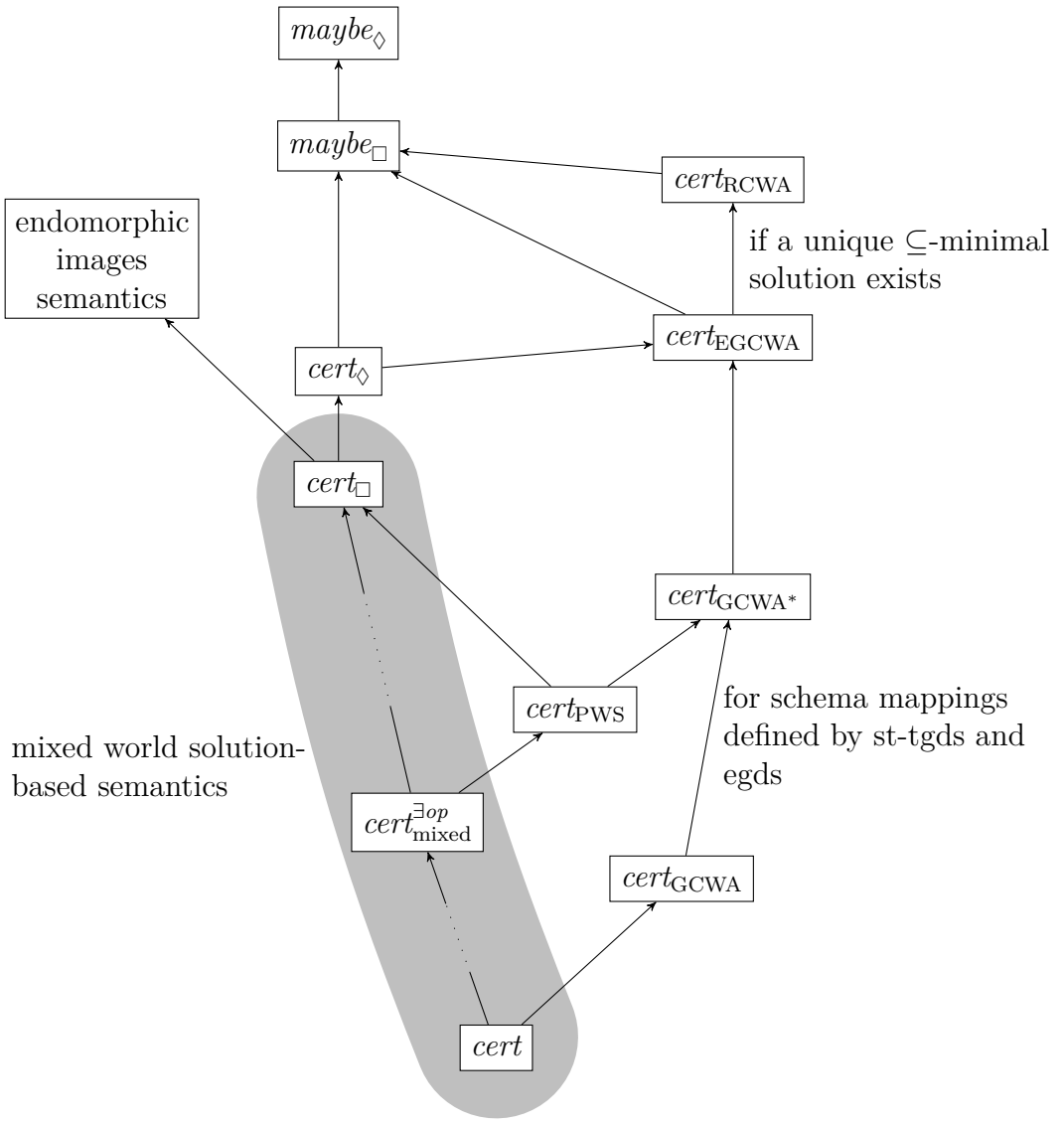


Figure 6.1: Relations between the semantics considered in this thesis. An arc from semantics sem_1 to semantics sem_2 indicates that for all schema mappings M with respect to which sem_1 and sem_2 are defined, for all source instances S for M, and for all queries q over M's target schema, we have $sem_1(q, M, S) \subseteq sem_2(q, M, S)$.

(see Fagin et al. [2005a], or Section 2.4). In this case, one does not evaluate the query directly on the universal solution T, but modifies T first to obtain another solution on which the query is then evaluated. Another case is computing the *GCWA*-answers* to *universal queries* with respect to schema mappings defined by *packed st-tgds* (see Section 5.4). Here, the universal solution must either be the core of the universal solutions, or some universal solution of which the core can be computed efficiently in order for the algorithm presented in Section 5.4 to work correctly. The crucial property is that the core of the universal solutions for a source instance S under a schema mapping M defined by packed st-tgds represents the set of all $\subseteq$-minimal solutions for S under M, on which the GCWA*-solutions for S under M can be defined. These $\subseteq$-minimal solutions are precisely the $\subseteq$-minimal instances in $poss(\text{Core}(M, S))$. Hence, one needs only $\text{Core}(M, S)$ in order to reconstruct all GCWA*-solutions for S under M. This property suffices to compute the GCWA*-answers to universal queries on M and S.

Furthermore, in many cases, universal solutions can be computed in polynomial time from a given source instance. In particular, as shown by Fagin et al. [2005a] (see also Section 2.2.2), this is possible if M is a fixed weakly acyclic schema mapping (cf., Section 2.2). So, for example, if M is a weakly acyclic schema mapping, it suffices to compute an arbitrary universal solution T for a given source instance S for M in polynomial time in order to answer fixed unions of conjunctive queries (with at most one inequality per disjunct), or Datalog queries, later in polynomial time. Moreover, by the results mentioned in Section 2.2.3, the core of the universal solutions can be computed in polynomial time for fixed weakly acyclic schema mappings. On the other hand, we have shown in Section 2.3 that there are schema mappings M (even defined by a set of tgds only) such that the Existence-of-Universal-Solutions(M) problem is undecidable. In particular, there is no algorithm that computes universal solutions for given source instances under M.

Besides the class of universal solutions, no other classes of solutions have been identified such that queries can be evaluated based on a solution from this class. However, in some cases where universal solutions cannot be used, *universal solution sets* introduced by Deutsch et al. [2008] (see also Section 2.4) may be helpful. Universal solutions are not single solutions—they consist of a finite number of solutions. However, it was shown that the certain answers to monotonic queries, or other classes of queries, can be computed from universal solution sets in a similar way as computing the certain answers to homomorphism-preserved queries from a universal solution. Furthermore, universal solution sets can often be computed using an extension of the chase. I refer the interested reader to Deutsch et al. [2008] to learn more about this topic.

6.3 Open Problems and Suggestions For Future Work

Quite a number of interesting research problems remain open. Some of these problems are described in the following.

First of all, since universal solutions are at the core of computing the certain answers to queries preserved under homomorphisms, and queries preserved under homomorphisms are a fundamental class of database queries, I think it is an important task to identify further classes of schema mappings M for which universal solutions can be computed, that is, for which there is an algorithm that takes a source instance S for M as input, and computes a universal solution for S under M if such a solution exists, and outputs that there is no solution otherwise. It is then interesting to analyze the exact complexity of the algorithm. It could be fruitful to first identify classes of schema mappings, for which *chase termination* is decidable, preferably in polynomial time. It seems to be possible to achieve such a decidability result for schema mappings defined by st-tgds and *guarded tgds*, where a tgd $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$ is *guarded* if and only if φ contains an atomic formula $R(\bar{u})$ such that all variables from $\bar{x}$ and $\bar{y}$ occur in $\bar{u}$. Guarded tgds have been considered by Calì et al. [2008, 2009], who studied the problem of evaluating conjunctive queries on the result of the chase on a given instance and a set of guarded tgds. Some ideas from Johnson and Klug [1982], Calì et al. [2004, 2008, 2009] may help to show that chase termination for schema mappings defined by st-tgds and guarded tgds is indeed decidable in polynomial time.

Universal solutions, or special universal solutions like the core of the universal solutions, are good for query evaluation in various settings. For example, universal solutions are good in the case of a weakly acyclic schema mapping and a query that is preserved under homomorphisms. Other results have been mentioned throughout this thesis. It is interesting to identify further classes of schema mappings and queries, for which universal solutions are good. That is, for which query answering semantics *sem*, for which schema mappings M and for which queries q, is there a polynomial time algorithm that, given (an arbitrary or a specific) universal solution for some source instance S for M, computes the answers to q on M and S with respect to *sem*?

It is also an interesting task to identify classes of solutions on which query answers can be computed if universal solutions do not suffice. One should not only restrict attention to single solutions, but consider other representation mechanisms that are useful for query answering, too. For example, universal solution sets should be further studied.

As mentioned at the end of Chapter 5, I believe that it is possible to extend Theorem 5.15 to schema mappings defined by st-tgds and egds. To obtain the result for schema mappings defined by st-tgds, it seems that all what has to be

done is to prove Lemma 5.31 for the case that the atom blocks of T are not packed.

A lot of more work has to be done for understanding the complexity of query answering under the various semantics discussed in this thesis. The fact that for some schema mappings M defined by st-tgds, and for some existential queries q the *data complexity* of computing the GCWA*-answers to q under M is hard does not imply that it could not be in polynomial time for other schema mappings defined by st-tgds, and other queries. The same comment applies to the other semantics. Furthermore, we only considered the data complexity of query answering – we did not consider the *combined complexity*, where the schema mapping and the query belong to the input.

For the case that query answering is hard under the desired semantics, one could also try to *approximate* the set of the query answers. Different kinds of approximations are conceivable. For example, one could try to compute a subset, or a superset respectively, of the answers that is as close to the optimal set of answers as possible.

Finally, it should be possible to fruitfully apply the various approaches for capturing implicit information in schema mappings to the schema mapping operators mentioned in Section 1.2. For example, one could study the composition operator or the inversion operator under the assumption that the only solutions are CWA-solutions, or GCWA*-solutions. For the composition operator, this was done already by Libkin and Sirangelo [2008] with respect to mixed world solutions. It is conceivable that composition and inversion of schema mappings can be specified by schema mappings defined by languages that are strictly less expressive than the languages required in the general case.

Bibliography

Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

Foto N. Afrati and Phokion G. Kolaitis. Answering aggregate queries in data exchange. In *Proceedings of the 27th ACM Symposium on Principles of Database Systems (PODS)*, pages 129–138, June 2008.

Foto N. Afrati, Chen Li, and Vassia Pavlaki. Data exchange in the presence of arithmetic comparisons. In *Proceedings of the 11th International Conference on Extending Database Technology (EDBT)*, pages 487–498, March 2008.

Shun'ichi Amano, Leonid Libkin, and Filip Murlak. XML schema mappings. In *Proceedings of the 28th ACM Symposium on Principles of Database Systems (PODS)*, pages 33–42, June 2009.

Marcelo Arenas and Leonid Libkin. XML data exchange: Consistency and query answering. *Journal of the ACM*, 55(2):Article 7, May 2008.

Marcelo Arenas, Pablo Barceló, Ronald Fagin, and Leonid Libkin. Locally consistent transformations and query answering in data exchange. In *Proceedings of the 23th ACM Symposium on Principles of Database Systems (PODS)*, pages 229–240, June 2004.

Marcelo Arenas, Pablo Barceló, and Juan Reutter. Query languages for data exchange: Beyond unions of conjunctive queries. In *Proceedings of the 12th International Conference on Database Theory (ICDT)*, pages 73–83, March 2009a.

Marcelo Arenas, Jorge Pérez, and Cristian Riveros. The recovery of a schema mapping: Bringing exchanged data back. *ACM Transactions on Database Systems*, 34(4):Article 22, December 2009b.

Albert Atserias, Anuj Dawar, and Phokion G. Kolaitis. On preservation under homomorphisms and unions of conjunctive queries. *Journal of the ACM*, 53 (2):208–237, March 2006.

Pablo Barceló. Logical foundations of relational data exchange. *SIGMOD Record*, 38(1):49–58, March 2009.

Catriel Beeri and Moshe Y. Vardi. A proof procedure for data dependencies. *Journal of the ACM*, 31(4):718–741, October 1984.

Philip A. Bernstein. Applying model management to classical meta-data problems. In *Proceedings of the 1st Conference on Innovative Data Systems Research (CIDR)*, pages 209–220, January 2003.

Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Data integration under integrity constraints. *Information Systems*, 29(2): 147–163, April 2004.

Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 70–80, September 2008.

Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In *Proceedings of the 28th ACM Symposium on Principles of Database Systems*, pages 77–86, June 2009.

Edward P. F. Chan. A possible world semantics for disjunctive databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(2):282–292, April 1993.

Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. On reconciling data exchange, data integration, and peer data management. In *Proceedings of the 26th ACM Symposium on Principles of Database Systems (PODS)*, pages 133–142, June 2007.

Alin Deutsch, Alan Nash, and Jeff Remmel. The chase revisited. In *Proceedings of the 27th ACM Symposium on Principles of Database Systems (PODS)*, pages 149–158, June 2008.

Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory*. Springer, 1999.

Ronald Fagin. Inverting schema mappings. *ACM Transactions on Database Systems*, 32(4):Article 25, November 2007.

Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, May 2005a.

Ronald Fagin, Phokion G. Kolaitis, and Lucian Popa. Data exchange: Getting to the core. *ACM Transactions on Database Systems*, 30(1):174–210, March 2005b.

Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang Chiew Tan. Composing schema mappings: Second-order dependencies to the rescue. *ACM Transactions on Database Systems*, 30(4):994–1055, December 2005c.

Ronald Fagin, Phokion G. Kolaitis, Alan Nash, and Lucian Popa. Towards a theory of schema-mapping optimization. In *Proceedings of the 27th Symposium on Principles of Database Systems (PODS)*, pages 33–42, June 2008a.

Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang Chiew Tan. Quasi-inverses of schema mappings. *ACM Transactions on Database Systems*, 33 (2):Article 11, June 2008b.

Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang Chiew Tan. Reverse data exchange: Coping with nulls. In *Proceedings of the 28th Symposium on Principles of Database Systems (PODS)*, pages 23–32, June 2009.

Ariel Fuxman, Phokion G. Kolaitis, Renée J. Miller, and Wang Chiew Tan. Peer data exchange. *ACM Transactions on Database Systems*, 31(4):1454–1498, December 2006.

Hervé Gallaire, Jack Minker, and Jean-Marie Nicholas. Logic and databases: A deductive approach. *ACM Computing Surveys*, 16(2):153–185, June 1984.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

Georg Gottlob and Alan Nash. Efficient core computation in data exchange. *Journal of the ACM*, 55(2):Article 9, May 2008.

Martin Grohe, André Hernich, and Nicole Schweikardt. Lower bounds for processing data with few random accesses to external memory. *Journal of the ACM*, 56(3):Article 12, May 2009.

Laura M. Haas, Mauricio A. Hernández, C. T. Howard Ho, Lucien Popa, and M. Roth. Clio grows up: From research prototype to industrial tool. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 805–810, June 2005.

Pavol Hell and Jaroslav Nešetřil. The core of a graph. *Discrete Mathematics*, 109(1–3):117–126, November 1992.

André Hernich. Answering non-monotonic queries in relational data exchange. In *Proceedings of the 13th International Conference on Database Theory (ICDT)*, pages 143–154, March 2010.

André Hernich and Nicole Schweikardt. CWA-solutions for data exchange settings with target dependencies. In *Proceedings of the 26th ACM Symposium on Principles of Database Systems (PODS)*, pages 113–122, June 2007.

André Hernich and Nicole Schweikardt. Reversal complexity revisited. *Theoretical Computer Science*, 401:191–205, July 2008.

André Hernich and Nicole Schweikardt. Logic and data exchange: Which solutions are "good" solutions? In Giacomo Bonanno, Benedikt Löwe, and Wiebe van der Hoek, editors, *Logic and the Foundations of Game and Decision Theory – LOFT 8*, volume 6006 of *Lecture Notes in Computer Science*, pages 61–85. Springer-Verlag, 2010.

Tomasz Imielinski and Witold Lipski, Jr. Incomplete information in relational databases. *Journal of the ACM*, 31(4):761–791, October 1984.

David S. Johnson and Anthony C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. In *Proceedings of the 1st ACM Symposium on Principles of Database Systems*, pages 164–169, March 1982.

Phokion G. Kolaitis. Schema mappings, data exchange, and metadata management. In *Proceedings of the 24th ACM Symposium on Principles of Database Systems (PODS)*, pages 61–75, June 2005.

Phokion G. Kolaitis, Jonathan Panttaja, and Wang Chiew Tan. The complexity of data exchange. In *Proceedings of the 25th ACM Symposium on Principles of Database Systems (PODS)*, pages 30–39, June 2006.

Georg Lausen, Michael Meier, and Michael Schmidt. On chase termination beyond stratification, September 2009. CoRR, arXiv:0906.4228v2.

Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2004.

Leonid Libkin. Data exchange and incomplete information. In *Proceedings of the 25th ACM Symposium on Principles of Database Systems (PODS)*, pages 60–69, June 2006.

Leonid Libkin and Cristina Sirangelo. Data exchange and schema mappings in open and closed worlds. In *Proceedings of the 27th ACM Symposium on Principles of Database Systems (PODS)*, pages 139–148, June 2008.

Aleksander Mądry. Data exchange: On the complexity of answering queries with inequalities. *Information Processing Letters*, 94(6):253–257, June 2005.

Bruno Marnette. Generalized schema mappings: From termination to tractability. In *Proceedings of the 28th ACM Symposium on Principles of Database Systems (PODS)*, pages 13–22, June 2009.

Jack Minker. On indefinite databases and the closed world assumption. In Donald W. Loveland, editor, *Proceedings of the International Conference on Automated Deduction (CADE)*, volume 138 of *Lecture Notes in Computer Science*, pages 292–308. Springer-Verlag, June 1982.

Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

Raymond Reiter. On closed world data bases. In Hervé Galaire and Jack Minker, editors, *Logic and Data Bases*, pages 55–76. Plenum Press, 1978.

Benjamin Rossman. Homomorphism preservation theorems. *Journal of the ACM*, 55(3):Article 15, July 2008.

Nan C. Shu, Barron C. Housel, Robert W. Taylor, Sakti P. Ghosh, and Vincent Y. Lum. EXPRESS: A data EXtraction, Processing, and REStructuring system. *ACM Transactions on Database Systems*, 2(2):134–174, June 1977.

Balder ten Cate and Phokion G. Kolaitis. Structural characterizations of schema-mapping languages. In *Proceedings of the 12th International Conference on Database Theory (ICDT)*, pages 63–72, March 2009.

Ron van der Meyden. Logical approaches to incomplete information: A survey. In *Logics for Databases and Information Systems*, pages 307–356. Kluwer, 1998.

Adnan H. Yahya and Lawrence J. Henschen. Deduction in non-Horn databases. *Journal of Automated Reasoning*, 1(2):141–160, June 1985.

Index

Lebenslauf

22.01.1980	geboren in Prenzlau
1986–1992	Grundschule in Schmölln (bei Prenzlau)
1992–1994	Lindenschule in Prenzlau (Gesamtschule mit gymnasialer Oberstufe)
1994–1999	Gesamtschule Talsand in Schwedt/Oder (Gesamtschule mit gymnasialer Oberstufe)
Juni 1999	Abitur
1999–2000	Wehrdienst
2000–2005	Informatikstudium mit Nebenfach Mathematik an der Technischen Universität Berlin
Febr. 2005	Diplom in Informatik Diplomarbeit: *Combining Self-Reducibility with Partial Information Algorithms* (Betreuer: Prof. Dr. Dirk Siefkes und Dr. Arfst Nickelsen)
Mai 2005–Nov. 2007	wissenschaftlicher Mitarbeiter in der Arbeitsgruppe *Logik und Datenbanktheorie* von Prof. Dr. Nicole Schweikardt an der Humboldt-Universität zu Berlin
Nov. 2007–März 2010	wissenschaftlicher Mitarbeiter in der Arbeitsgruppe *Theorie komplexer Systeme* von Prof. Dr. Nicole Schweikardt an der Goethe-Universität Frankfurt am Main
seit Apr. 2010	wissenschaftlicher Mitarbeiter in der Arbeitsgruppe *Logik in der Informatik* von Prof. Dr. Martin Grohe an der Humboldt-Universität zu Berlin