Data Sharing and Exchange: Semantics and Query Answering

by

Rana Awada

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements

For the Ph.D. degree in

Computer Science

School of Electrical Engineering and Computer Science

Faculty of Engineering

University of Ottawa

Abstract

Exchanging and integrating data that belong to worlds of different vocabularies are two prominent problems in the database literature. While data coordination deals with managing and integrating data between autonomous yet related sources with possibly distinct vocabularies, data exchange is defined as the problem of extracting data from a source and materializing it in an independent target to conform to the target schema. These two problems, however, have never been studied in a unified setting which allows both the exchange of the data as well as the coordination of different vocabularies between different sources. Our thesis shows that such a unified setting exhibits data integration capabilities that are beyond the ones provided by data exchange and data coordination separately.

In this thesis, we propose a new setting – called *DSE*, for *Data Sharing and Exchange* – which allows the exchange of data between independent source and target applications that possess independent schemas, as well as independent yet related domains of constants. To facilitate this type of exchange, we extend the source-to-target dependencies used in the ordinary data exchange setting which allow the association between the source and the target at the schema level, with the mapping table construct introduced in the classical data coordination setting which defines the association between the source and the target at the instance level. A mapping table construct defines for each source element, the set of associated (or corresponding) elements in the domain of the target. The semantics of this association relationship between source and target elements change with different requirements of different applications.

Ordinary DE settings can represent DSE settings; however, we show that there exist DSE settings with particular semantics of related values in mapping tables where DE is not the best exchange solution to adopt.

The thesis introduces two DSE settings with such a property. We call the first *DSE with unique identity semantics*. The semantics of a mapping table in this DSE setting specifies that each source element should be uniquely mapped to at least one target element that is associated with it in the mapping table.

In this setting, classical DE is one method to perform a data exchange; however, it is not the best method to adopt, since it can not represent exchange applications, that require – as DC applications – to compute both portions as well as complete sets of certain answers for conjunctive queries. In addition, we show that adopting known DE universal solutions as semantics for such DSE settings is not the best in terms of efficiency when computing certain answers for conjunctive queries.

The second DSE setting that the thesis introduces with the same property is called *DSE with equality semantics*. This setting captures interesting meaning of related data in a mapping table. Such semantics impose that each source element in a mapping table is related to a target element only if both elements are equivalent (i.e they have the same meaning).

We show in our thesis that this DSE setting differs from ordinary DE settings in the sense that additional information could be entailed under such interpretation of related data. Also, this added information needs to be augmented to both the source instance and the mapping table in order to generate target instances that correctly reflect both in a DSE scenario. In other words, we can say that in such a DSE setting, a source instance and a mapping table can be *incomplete* with respect to the semantics of the mapping table.

We formally define the two afore mentioned semantics of a DSE setting and we distinguish between two types of solutions for this setting, named, *universal DSE solutions*, which contain the complete set of exchanged information, and *universal DSE KB-Solutions*, which store a portion of the exchanged information with implicit information in the form of a set of rules over the target. DSE KB-Solutions allow applications to compute on demand both a *portion* and the *complete* set of certain answers for conjunctive queries. In addition, we define the semantics of conjunctive query answering, and we distinguish between sound and complete certain answers for conjunctive queries and we define the algorithms to compute these efficiently. Finally, we provide experimental results which compare the run times to generate DSE solutions versus DSE KB-solutions, and compare the performance of computing sound and complete certain answers for conjunctive queries using both types of solutions.

Acknowledgements

First I would like to express my sincere appreciation to my advisor Dr. Iluju Kiringa for giving me the chance to do my research on a topic that falls in the field of data exchange and data integration which is quite an exciting and evolving field. I am grateful for his patience, motivation, and guidance during my research work. Besides my advisor, my sincere gratitude goes to Dr. Pablo Barcelo for his valuable guidance and direction through out my thesis research. Also, I can not forget the special and big thanks to the professors who contributed in providing me directions and in answering my questions. The immense knowledge and profound research experience provided to me gave me the enthusiasm and support to finish my thesis.

I would like also to thank my friends Mehedi Masud and Flavio Rizzolo who have supported me and given me the enthusiasm through out my research.

Finally, I would like to express my sincere gratitude to my great precious family Mom, Dad, my husband, my two wonderful kids, my brother and sisters, and all the rest for their amazing spiritual support through my research. I want to tell them that I owe them a lot and I am now only because of them being there for me all this time.

Contents

List of Tables

List of Figures

List of Examples

Chapter 1

Introduction

1.1 The Context

Over the past 10 years, a big bulk of work has tackled the problem of integrating information from different sources, and several settings have been introduced to solve this problem. Two main settings proposed in the literature among others are the data exchange setting [1, 2] and the data coordination setting [3, 4, 5, 6].

A data exchange (DE) setting considers the problem of retrieving information that conforms to a source schema, transforming it to conform to an independent target schema, and materializing it in the target. A data coordination (DC) setting, on the other hand, integrates data by allowing access to instances of independent sources that possibly belong to different sets of vocabularies, without necessarily exchanging it and while maintaining autonomy. A DC setting addresses the issue of heterogeneous vocabularies of different sources by introducing a new construct – namely, *mapping table* construct – which stores association relationships between different vocabularies in the form of sets of data mappings. A data mapping in a mapping table maps corresponding elements that belong to different domains of constants.

Each of the DE and DC problems have been tackled and solved separately by formally defining the DE setting and the DC setting; however, neither of those settings solve these two problems when combined together. Therefore, this thesis introduces a new class of settings,

called the *data sharing and exchange* (DSE) settings which solve the problem of exchanging information between two applications that possess distinct schemas, as well as distinct, yet related domains of constants. Despite the importance of this topic (See Section 1.2 below), the fundamentals of such process have not been laid out to date. Exchanging related information that reside in different data sources, and storing it in an independent target application using a unified set of vocabularies is an important problem that occurs in most enterprise applications that involve periodic reporting and/or decision making.

A motivating example of the applications described above would be that of academic institutions, like universities, which accept students' transfers from other foreign universities. When a student does a transfer from a university $Univ_a$ to a university $Univ_b$, he may receive credits for the course work he already completed at $Univ_a$, and use those credits to fulfil some of the requirements for his new credential. For this reason, transfer credit equivalence tables, which are one form of mapping tables, are used to allow $Univ_b$ make those decisions. Transfer credit equivalence tables usually associate a set of courses provided in $Univ_a$ to the set of corresponding courses provided at $Univ_b$. Another example of such applications is biological databases that require to exchange protein information and their related genes. In such type of applications, the necessity of exchanging related information that belong to different sets of vocabularies is quite common and fundamental, since genes and related proteins may have different names in different databases.

1.2 Motivation and Objectives

As we mentioned earlier, data exchange [1, 2] and data coordination settings [3, 4, 5, 6] are among the most popular settings introduced in the literature to solve the problem of integrating information from different sources.

A DE setting considers the problem of taking data residing in a source instance and transforming it to populate an independent target schema. More formally, as given in [1], a DE setting $\mathfrak{S}$ consists of a source schema $\mathbf{S}$, a target schema $\mathbf{T}$, and a set Σ_{st} of source-to-target

(st) dependencies which describe the structural changes made to data as we move it from the source to the target[1]. This setting supports exchange of information between two applications that refer to the same object using the same instance value. We present in Example 1.2.1 a sample DE instance of academic institutions like universities. We use similar source and target schemas in our examples to simplify our discussion throughout the thesis.

Example 1.2.1 *Let* $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ *be a DE setting. Let* $\mathbf{S}$ *be the source schema for the Carleton University (UOC) given in Figure1.1(a) and* $\mathbf{T}$ *be the target schema for the University of Ottawa (UOO) given in Figure1.1(b). Relation* $Student$ (St) *stores students name and age (and address) information. Relation* $Course$ (Cr) *stores courses ids and names information, in addition to the program name which provides each course. Finally, relation* $Enroll$ $(Take)$ *stores the set of courses that each student completed with the grade that he/she received.*

Student(Sname, Sage)
Course(Cid, Cname, Pname)
Enroll(Sname, Cid, Cgrade)
(a)Source Schema for the University of Carleton

St(Sname, Sage, Saddress)
Cr(Cid, Cname, Pname)
Take(Sname, Cname, Cgrade)
(b) Target Schema for the University of Ottawa

Figure 1.1: Source and Target Database Schemas

Student

Sname	Sage
Alex	18

Course

Cid	Cname	Pname
COMP1005	Introduction to Computer Science I	CS

Enroll

Sname	Cid	Cgrade
Alex	COMP1005	80

Figure 1.2: Source Instance I

St

Sname	Sage	Saddress
Alex	18	$\perp$

Take

Sname	Cid	Cgrade
Alex	COMP1005	80

Figure 1.3: Target Instance J

Further, assume that Σ_{st} *consists of the following st-dependencies:*

$(a) \forall x \forall y \big(Student(x, y) \rightarrow \exists z St(x, y, z) \big)$

$(b) \forall x \forall y \forall z \big(Enroll(x, y, z) \rightarrow Take(x, y, z) \big).$

[1] More notations are given in Section 2.1

Given the source instance I in Figure 1.2, then, according to the usual DE terminology [1], a target instance J in Figure 1.3, where $\perp$ is fresh unknown (or null*) value, is a target instance that satisfies Σ_{st} when considered with the source instance I, and thus is called a solution for I under $\mathfrak{S}$.*

□

In Example 1.2.1 we see that ordinary DE settings support exchanging information to target applications that refer to elements ($Alex$, 18, $COMP1005$, etc.) using the same names as in the source application. However, as we mentioned earlier, there exists cases where objects in the source are named and referred to differently in the target. For example, the University of Ottawa uses a different name for a course that is equivalent to the course $COMP1005$ provided at the Carleton University, and elements in the extension of the UOO database schema does not support the use of UOC vocabulary. In this case, the DE solution presented in Example 1.2.1 does not generate a solution J that can be materialized in the UOO application since J should solely use the vocabulary defined by UOO in the target.

Unlike data exchange, DC settings solve the problem of integrating information of different sources, with possibly different yet related domain of constants, by providing access to this data without necessarily moving it [3, 4, 5, 6]. DC settings have been studied mainly in peer-to-peer networks, where sources – called peers – possess equal capabilities and responsibilities in terms of propagating changes and retrieving related information. DC settings, and specifically data sharing settings [5, 7, 8], introduce the mapping table construct to facilitate augmenting information retrieved from one source with related information fetched from remote sources – called *acquainted* sources – which might possess a different domain of constants. A mapping table, $\mathcal{M}$, specifies for each value in one source, the set of related (or corresponding) values that belong to an acquainted source. In such a setting, retrieving information from acquainted sources is performed by re-writing a query issued to one source using a mapping table $\mathcal{M}$ (more than one mapping table could be used) and then collecting the results of those queries from corresponding remote sources.

Formally, a DC setting $\mathfrak{S}$ consists of two schemas $\mathbf{S}_1$ and $\mathbf{S}_2$, and a set of mapping tables

$\{\mathcal{M}\}$. We give in the following example of a data coordination instance that allows integrating information from two different universities that uses different names for related courses and that use different grading systems.

Example 1.2.2 *Let $\mathfrak{S}$ be a DC setting. Suppose that $\mathbf{S}_1$ in $\mathfrak{S}$ is the schema for the University of Carleton and $\mathbf{S}_2$ in $\mathfrak{S}$ is the schema of the University of Ottawa. In reference to Example 1.2.1, assume that $\mathbf{S}_1$ is the same schema as $\mathbf{S}$ and $\mathbf{S}_2$ is the same schema as $\mathbf{T}$. Let I be the instance of $\mathbf{S}_1$ given in Figure 1.5 and J be an instance of $\mathbf{S}_2$ given in Figure 1.6. Further, assume that $\mathbf{S}_1$ and $\mathbf{S}_2$ are two acquainted sources that are connected by the mapping table $\mathcal{M}$ given in Figure 1.4.*

M

S	T
ECOR1606	CSI1390
COMP1005	CSI1390
COMP1005	CSI1790
COMP 4001	CSI4109
CS	CS
ENG	ENG
80	B
70	C
90	A
Alex	Alex
Tom	Tom

Figure 1.4: Mapping Table $\mathcal{M}$

Student

Sname	Sage
Alex	18
Tom	19

Course

Cid	Cname	Pname
ECOR1606	Problem Solving and Computers	ENG
COMP1005	Introduction to Computer Science I	CS
COMP4001	Distributed Computing	CS

Enroll

Sname	Cid	Cgrade
Alex	ECOR1606	80
Tom	COMP 4001	70

Figure 1.5: An instance I of $\mathbf{S}_1$

St

Sname	Sage	Saddress
Ben	18	Ottawa

Cr

Cid	Cname	Pname
CSI1390	Introduction to Computers	CS
CSI1790	Introduction aux Ordinateurs	CS
CSI4109	Introduction to Distributed Computing	CS

Take

Sname	Cid	Cgrade
Ben	CSI1390	C
Ben	CSI4109	A

Figure 1.6: An instance J of $\mathbf{S}_2$

As given in [6, 9], a DC system allows a user to retrieve the list of UOO students that have completed the CS course $CSI1390$ or the CS course $CSI4109$, integrated with the list of UOC students that have completed a CS course that is related – or that corresponds – to these courses according to the semantics of the mapping table $\mathcal{M}$ given in Figure 1.4. The DC system performs this operation by first computing the conjunctive query q, given below, using the list of students that have finished the courses $CSI1390$ and $CSI4109$ at UOO. Then it re-writes q to a conjunctive query q', given below, using the st-mapping table $\mathcal{M}$ to compute the list of UOC students that have completed a course that is related to either $CSI1390$ or $CSI4109$ course according to $\mathcal{M}$. Finally, it augments the results of applying q to the instance J with the result of applying q' to the instance I and returns it to the user.

Suppose the following query q for UOO:

q: **Select** $Sname, Sage, Cid, Cgrade$

From $St, Cr, Take$

Where $St.Sname$ = $Take.Sname$ **And** $Cr.Cid$ = $Take.Cid$ **And** $Cr.Pname$ = CS

And ($Take.Cid$ = $CSI1390$ **OR** $Take.Cid$ = $CSI4109$).

Then a re-written query q' for UOC would be the following:

q': **Select** $Sname, Sage, Cid, Cgrade$

From $Student, Course, Enroll$

Where $Student.Sname$ = $Enroll.Sname$

And $Course.Cid$ = $Enroll.Cid$ **And** $Course.Pname$ = CS

And ($Enroll.Cid$ = $COMP1005$ **OR** $Enroll.Cid$ = $ECOR1606$ **OR**

$Enroll.Cid$ = $COMP4001$).

In this case, in a DC setting, the integrated result returned by q to I would be $\{$ $(Ben, 18, CSI1390, C)$, $(Ben, 18, CSI4109, A)$, $(Alex, 18, ECOR1606, 80)$, $(Tom, 18, COMP4001, 70)$ $\}$. □

We can see in Example 1.2.2 that the amalgamated view of related information provided

by a data coordination setting from UOO and UOC databases is not good enough to provide summaries or make decisions about this data. The reason is that the integrated result of the posed queries provide in reality a consolidated set of information but not a homogeneous view of it since this information still consists of data of different sets of vocabularies that belong to the different connected sources. If we refer to Example 1.2.2, we can see that a UOO user is not able to compare the grades of students from both UOO and UOC universities that completed the UOO course $CSI1390$ versus the grades for the $CSI4109$ UOO course. We can see that unless users are well aware of the association relationships between UOO and UOC courses defined in the mapping table $\mathcal{M}$, it would be impossible for them to perform this comparison for the performance of students.

Furthermore, for a more particular semantics of mapping tables, like the *Equivalence* semantics that we will introduce later in Chapter 5, a DC setting does not provide a good solution for the problem we are addressing in our thesis work[2].

Example 1.2.3 *In reference to Example 1.2.2, suppose $Alex$ at UOC completed the ENG course $ECOR1606$ instead of CS course $COMP1005$ and had the same grade. Then, users can not identify from the result of q, as we show them in Chapter 5, that $Alex$ is considered with respect to UOO and, using the semantics of the mapping table $\mathcal{M}$, as has finished the CS course $CSI1390$. Thus, applying the DC solution to integrate information that belong to different domains of vocabularies is not applicable for some particular semantics of mapping tables.*

Therefore, we conclude from Example 1.2.1 and Example 1.2.2 that integrating related information which reside in different sources with different domains of constants and presenting it using a unified set of vocabularies is a problem that can not be solved by the ordinary DE or DC settings separately.

Such a problem motivated us to introduce a new class of settings that integrate the data exchange and the data coordination approaches, to exchange data between a source and a target

[2]Equivalence semantics mean that two elements are mapped in a mapping table if both have the same meaning.

which possess different schemas and different sets of vocabularies. We call this new class of settings *data sharing and exchange* (DSE) settings. A DSE setting constitutes of a source schema $\mathbf{S}$, a target schema $\mathbf{T}$ and a *mapping relation* [5] $\mathcal{M}$ that does not exist in neither $\mathbf{S}$ nor $\mathbf{T}$. We call it a *source-to-target* mapping relation (st-mapping). $\mathcal{M}$ defines in its extension the value correspondences from the source elements to their related target elements.

To allow the coordination of data in DSE while performing data exchange, we use source-to-target dependencies in Σ_{st} which consist of conjunctions of st-mapping relations $\mathcal{M}$ that specify for each source element exchanged, the set of related target elements to be materialized in the target. Formally, these are FO sentences of the following form: $\forall\bar{x}\forall\bar{y}\forall\bar{z}\,(\phi(\bar{x},\bar{y}) \wedge \mu(\bar{x},\bar{z}) \rightarrow \exists\bar{w}\,\psi(\bar{z},\bar{w}))$, where (i) $\phi(\bar{x},\bar{y})$ and $\psi(\bar{z},\bar{w})$ are conjunctions of relational atoms over $\mathbf{S}$ and $\mathbf{T}$, respectively, (ii) $\mu(\bar{x},\bar{z})$ is a conjunction of atomic formulas that only use the relation symbol $\mathcal{M}$ of the source.

In ordinary data exchange settings [1, 10], authors identified universal solutions as a class of good solutions to compute the set of certain answers for conjunctive queries. Informally, a certain answer [1, 10] for a conjunctive query q in a DE setting $\mathfrak{S}$ contains the set of tuples that belong to the evaluation of q on every DE solution under $\mathfrak{S}$. In data coordination settings and specifically data sharing [5], on the other hand, authors introduced the notions of sound and complete query translations to represent applications that require, on demand, to compute a portion of the set of correct answers of an input conjunctive query rather than computing the complete set, and such that this portion completely represents the complete set of certain answers. We show this in the following example.

Example 1.2.4 *So, in reference to Example 1.2.2, if the student* $Alex$ *in UOC completed both courses* $COMP1005$ *and* $ECOR1606$ *with a grade C, then the answer* $\{$ $(Alex, 18, COMP1005, C)$, $(Alex, 18, ECOR1606, C)$, $(Ben, 18, CSI1390, C)$, $(Ben, 18, CSI4109, A)$, $(Tom, 18, COMP4001, 70)$ $\}$ *is the complete set of certain answers for* q *and the set* $\{$ $\{Alex, 18, COMP1005, C\}$, $\{Ben, 18, CSI1390, C\}$, $\{Ben, 18, CSI4109, A\}$, $\{Tom, 18, COMP4001, 70\}$ $\}$ *is a sound set of certain answers for* q.

Clearly, a DSE setting in its general form can be reduced into a data exchange setting $\mathfrak{S}'$

that consists of a source schema $(\mathbf{S} \cup \{\mathcal{M}\})$ – which includes the st-mapping relation $\mathcal{M}$ in $\mathbf{S}$, a target schema $\mathbf{T}$, and a set of source-to-target tgds Σ_{st}. With such a representation, users can generate target instances that contain the complete set of exchanged information and can generate certain answers of conjunctive queries that contain the complete set of correct answers. However, following the intuition of sound and complete answers in DC settings [5], in some scenarios, users require to be able on demand generate a portion of the set of correct answers rather than the full set. Intuitively, DSE settings that use the ordinary data exchange representation can not provide such flexibility to users. Therefore, we define two DSE settings with particular semantics of related data in st-mapping tables $\mathcal{M}$ – namely, DSE with unique identity semantics (DSE_U) and DSE with equivalence semantics (DSE_E) – that allow users generate such type of answers – we call those sound and complete certain answers for conjunctive queries. Also, we represent DSE_U and DSE_E settings using the *Knowledge Exchange* Framework, introduced in [11]. With such representation we can generate two types of solutions. The first type is called DSE solutions. These store the complete set of exchanged data and return the complete set of certain answers for an input conjunctive query. The second type is called DSE KB-solutions. These store a portion of the exchanged data with implicit information in the form of a set of FO sentences; applying these FO sentences to DSE KB-solutions generate target instances with the complete set of exchanged data, namely DSE solutions. In addition, we identify a specific class of DSE KB-solutions that allow applications to generate a portion of the set of certain answers to a conjunctive query which closely represents the complete set of answers. Such solutions are used on demand in conjunction with the implicit data to generate the complete set of certain answers.

We provide in Section 1.3 the methodology of our research, then we list in Section 1.4 the technical problems that we attacked and our main contributions in regards to DSE settings.

1.3 Methodology

Several settings have been introduced in the literature to solve the problem of integrating information. Data exchange and data coordination settings are two of those that addressed this problem and that were formally defined in the literature. We introduce in this thesis a data sharing and exchange setting that extends both data exchange and data coordination settings to solve the problem of exchanging information between applications that possess different sets of vocabularies. We formally characterize this setting and define the components of it by representing it using the knowledge base exchange setting. We define in our work the semantics of this setting and we distinguished between DSE and DSE KB-solutions. We extended the formal results given in data exchange and data coordination settings, and we provided the data complexity for generating both solutions. We showed that generating both solutions is tractable and we provided algorithms to generate those. In addition, we adopted the concept of most compact solutions introduced in data exchange settings and formally defined the class of minimal DSE KB-solutions. We defined the data complexity to generate those and showed it is tractable. We also defined formally the semantics of conjunctive query answering in a DSE setting and distinguished between sound and complete certain answers. In order to compare the performance of DSE versus DSE KB-solutions – mainly minimal DSE KB-solutions – we implemented a prototype system that generates both DSE and minimal DSE KB-solutions. We also compared the performance of both solutions when computing both sound and complete certain answers. Then finally, we briefly addressed data sharing and exchange in more collaborative settings where source instances contain incomplete information in the form of null values. We identified the best tool to represent the most compact DSE KB-solutions – which might not be minimal DSE KB-solutions in some DSE scenarios. We also showed how generating minimal DSE KB-solutions becomes a harder problem than the case in DSE settings with complete source information.

1.4 Contributions

Our main contributions in our thesis work are the following:

1. We introduce two DSE settings with particular semantics of st-mapping tables, namely, DSE with unique identity semantics (DSE_U) and DSE with equivalence semantics (DSE_E) and we formally define the rules that enforce such semantics.

2. We define two types of solutions – called DSE solutions and DSE KB-solutions – as semantics for both settings. We also characterize universal DSE solutions and universal DSE KB-solutions. In so doing, we define the set of rules necessary to generate a universal DSE solution given a universal DSE KB-solution as input. In addition, we provide complexity results and algorithms to generate both types of solutions in DSE_U and DSE_E settings.

3. We extend the class of minimal knowledge base solutions defined in [11], which are knowledge base solutions that possess minimum amount of explicit information in addition to being cores [10] (a core KB solution is a smallest subset KB solution), and we define the class of minimal universal DSE KB-solutions that have a minimal size and a minimal domain of constants. Also, we specify the complexity results and the algorithms to generate those in both DSE_U and the DSE_E settings.

4. We formally define the set of sound and complete certain answers for conjunctive queries, and we provide the algorithms to generate those when applying conjunctive queries to both universal DSE KB-solutions and minimal universal DSE KB-solutions in DSE_U and DSE_E settings.

5. We define a knowledge exchange algorithm for DSE_E and DSE_U settings, which leverage previous approaches defined in [12, 13, 14] that generate target instances with minimal sizes, in order to generate minimal universal DSE KB-solutions that have a minimal size and a minimal domain of constants.

6. Additional to the above, we implement in a DSE prototype system the knowledge exchange semantics that we introduced in both DSE_U and DSE_E settings. This system leveraged the work done in the state of the art ++Spicy system [15] to generate minimal universal DSE KB-solutions. It effectively generates universal DSE solutions and minimal universal DSE KB-solutions, and it computes certain answers for CQs using both types of solutions.

7. Finally, we address the data sharing and exchange problem in a setting where the source instance could be a result of an earlier exchange from a previous source instance;and hence may contain null values. We leverage the results given in [11] to define the data complexity of generating DSE KB-solutions and that of generating a minimal one.

1.5 Articles

The following list of articles have been published:

1. GDE: General Data Exchange with Schema and Data Level Mappings. AMW 2013 (A longer version will be submitted to *The Journal on Data Semantics*).

2. Sharing and Exchanging Data. KDPD 2013 (A longer version is submitted to *The Journal on Data Semantics*).

1.6 Outline

The remainder of this thesis is organized as follows. Chapter 2 reviews related literature and presents technical preliminaries. Chapter 3 addresses the problem of ordinary data exchange with coordination of related information that belong to worlds of different vocabularies. In this chapter, we introduce the DSE setting to formalize this problem. We also provide the intuition behind the different components of the problem. Chapters 4 and 5 address the DSE settings, DSE_U and DSE_E, respectively. We define the semantics of these DSE settings, as well as the

semantics of conjunctive query answering. Chapters 4 and 5 also provide implementations and evaluations of the proposed algorithms for generating the different solutions introduced in these chapters respectively and for computing sound and complete certain answers. Then, Chapter 6 considers data sharing and exchange where source instances may contain unknown information. We define in this chapter the data complexity for generating universal DSE and MUDSE solutions. Finally, Chapter 7 presents a summary of the research and states future research direction.

Chapter 2

Related Work

The present chapter reviews previous work that we referred to in this thesis. As we mentioned earlier in Chapter 1, the objective of the thesis is to address the problem of exchanging information from one application – called *source* – then generate the corresponding set of related data and store it in an independent application – called *target* – based on a predefined association relationship at both the schema and the instance level between the source and the target.

The chapter starts with preliminaries section about the main database notations that we will be using through-out the thesis. Then, we provide a short technical review of previous work related to the main topics relevant to our thesis work. These are classified in four classes: data exchange, databases with incomplete information, data coordination, and knowledge exchange.

2.1 Preliminaries

A *schema* $\mathbf{R}$ is a finite set $\{R_1, \ldots, R_k\}$ of relation symbols, with each R_i having a fixed arity $n_i > 0$, $1 \leq i \leq k$. Let D be a countably infinite domain. An *instance* I of $\mathbf{R}$ assigns to each relation symbol R_i of $\mathbf{R}$ a n_i-ary relation $R_i^I \subseteq \mathsf{D}^{n_i}$, $1 \leq i \leq k$. All instance of R_i, $1 \leq i \leq n$, are considered to be finite. Sometimes we write $R_i(\bar{t}) \in I$ instead of $\bar{t} \in R_i^I$, and call $R_i(\bar{t})$ a *fact* of I, $1 \leq i \leq k$. The *domain* $dom(I)$ of instance I is the set of all elements that occur

in any of the relations R_i^I. We often define instances by simply listing the facts that belong to them. Further, every time that we have two disjoint schemas $\mathbf{R}$ and $\mathbf{S}$, an instance I of $\mathbf{R}$ and an instance J of $\mathbf{S}$, we define (I, J) as the instance K of schema $\mathbf{R} \cup \mathbf{S}$ such that $R^K = R^I$, for each $R \in \mathbf{R}$, and $S^K = S^J$, for each $S \in \mathbf{S}$. Most of the relevant settings we are referring to in this chapter involve two independent schemas called *source* and *target*. We refer to a schema $\mathbf{S}$ as a source schema and to a schema $\mathbf{T}$ as a target schema. We use symbols $I, I', I_1, \ldots$ to identify source instances and $J, J', J_1, \ldots$ symbols to identify target instances.

We denote constants by lowercase letters $a, b, c, \ldots$, and nulls by symbols $\perp, \perp', \perp_1, \ldots$ The term *null* is used to represent 'values of attributes' in an interpretation of a relation R_i, $1 \leq i \leq k$, such that these values exist but are unknown. An Instance I of a schema $\mathbf{R}$ is considered *incomplete* if it contains unknown information in the form of nulls, and considered *complete* otherwise. Let Const and Var be infinite and disjoint sets of constants and nulls, respectively, and assume that $\mathsf{D} = \mathsf{Const} \cup \mathsf{Var}$. Also, if $R_i(\bar{t})$ is a fact of an instance J of $\mathbf{T}$ and $R_i(\bar{t})$ contains elements c such that $c \in \mathsf{Const}$, then we denote $R_i(\bar{t})$ by $R_i(\bar{t}) \downarrow$.

2.2 Data Exchange

Data Exchange [1, 2, 16, 11, 17] is the problem of transforming a source instance I over a source schema $\mathbf{S}$ and materializing it in an instance J over an independent target schema $\mathbf{T}$ such that the target constraints (in the form of rules) are satisfied by J and the source-to-target constraints (in the form of rules) are satisfied by I and J together. Over the past 10 years, data exchange has been a significant problem that took wide attention in the literature, especially with the advent of *smi-structured* data and data with *incomplete* information. Data exchange is widely used in practice and several systems that support it have been implemented, including the Clio and the ++SPicy systems [18, 19, 15].

The data exchange problem has been studied thoroughly in a setting where the source instance is complete [1, 10, 20, 19]. Also, other work took a step further and addressed the data exchange problem in a more collaborative setting where targets of one exchange instance can

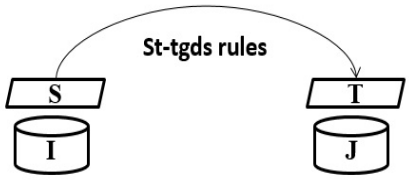

Figure 2.1: A Data Exchange Setting

become the source of a different exchange instance. As we shall see later in Section 2.2.2, target instances in a data exchange setting are allowed to contain null values. Therefore, in the second scenario, sources of certain data exchange instances might end up containing null values. Therefore, authors in [21, 22, 11] addressed the problem of data exchange with incomplete information. We illustrate a DE setting in Figure 2.1.

2.2.1 Relational Data Exchange with Complete Source Information

Formally, a *data exchange* (DE) *setting* is a tuple $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$, where $\mathbf{S}$ is a source schema, $\mathbf{T}$ is a target schema, and $\mathbf{S}$ and $\mathbf{T}$ do not have predicate symbols in common. We consider in this sub-section that for every instance I of $\mathbf{S}$, it holds that $dom(I) \subseteq \mathsf{Const}$ (set of constants); that is, source instances are assumed to be complete. On the other hand, for every instance J of $\mathbf{T}$, it holds that $dom(J) \subseteq \mathsf{Const} \cup \mathsf{Var}$; that is, target instances are allowed to contain *unknown* values in the form of nulls. In the incomplete information terminology, instances with nulls (and, in particular, target data exchange instances) are called *naïve* tables [23]

Σ_{st} is a set of schema mappings, named *source-to-target tuple-generating* dependencies(st-tgd), which are high level specifications that define the relationship between a source and a target schemas. These specifications are interpreted as a set of declarative assertions that captures the interaction between two database schemas at a logical level while hiding the implementation details at the physical level. Schema mappings are widely used in many data management applications like data exchange, data coordination, and data sharing. In particular, schema mappings form the essential building blocks in DE and DC settings. Although both settings

addressed the main issue of integrating data from different sources, their corresponding techniques stayed distinct from one another.

Given a source schema $\mathbf{S}$ and a target schema $\mathbf{T}$, there exists three approaches for specifying schema mappings. These are: *Global-as-view* (GAV), *local-as-view* (LAV), and *global-and-local-as-view* (GLAV) [24, 25]. GAV mappings are FO-sentences of the following form

$$\forall \bar{x}\,(\phi(\bar{x}) \rightarrow\ g(\bar{x})),$$

where $\phi(\bar{x})$ is a conjunction of relational atoms (a view) over $\mathbf{S}$ and g is an element of the target schema $\mathbf{T}$. LAV mappings on the other hand are FO-sentences of the following form

$$\forall \bar{x}\,(g(\bar{x}) \rightarrow\ \phi(\bar{x})),$$

where g is an element of the source schema $\mathbf{S}$ and $\phi(\bar{x})$ is a conjunction of relational atoms over the target schema $\mathbf{T}$. Finally, GLAV mappings are FO-sentences of the following form

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

where $\phi(\bar{x})$ is a conjunction of relational atoms over $\mathbf{S}$ and $\psi(\bar{x}, \bar{z})$ is a conjunction of relational atoms over $\mathbf{T}$. Most data exchange systems studied so far use GLAV mappings in their specifications.

Σ_t consists of a set of *target* constraints that aim at making sure that the target data matches the structure of the target schema. Target constraints can be of two types. The first type – called *target tuple-generating* dependencies (t-tgds) – is a set of rules which enforce the presence of certain tuples in a database instance if certain other tuples are already present. Formally, these are FO-sentences of the form

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

where $\phi(\bar{x})$ and $\psi(\bar{x}, \bar{z})$ are conjunctions of relational atoms over $\mathbf{T}$. Every target t-tgd expresses the containment of one conjunctive query in another conjunctive query. Clearly, both

inclusion dependencies and multivalued dependencies are special cases of t-tgds. The second type of target constraints – called *target equality-generating* dependencies (t-egds) – is a set of rules which express integrity constraints in a database instance. Formally, these are FO-sentences of the form

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

where $\phi(\bar{x})$ is a conjunction of relational atoms over $\mathbf{T}$, and x_i and x_j are variables in $\bar{x}$. Functional dependencies are a special case of t-egds.[1]

Solutions and universal solutions

An instance J of a target schema $\mathbf{T}$ is said to be a *solution* for a source instance I under $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$, if the instance (I, J) of $\mathbf{S} \cup \mathbf{T}$ satisfies every st-tgd in Σ_{st} and J satisfies every t-tgd and t-egd in Σ_t. If $\mathfrak{S}$ is clear from the context, we shall say that J is a solution for I.

Example 2.2.1 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be the DE instance given in Example 1.2.1. Then, the target instance $J = \{St(Alex, 18, \perp), Take(Alex, COMP1005, 80)\}$, where $\perp \in \mathsf{Var}$, is a DE solution under $\mathfrak{S}$ for the source instance I. In fact, in this DE instance, I has infinitely many solutions.* □

The data exchange literature has identified a class of "good" solutions, called the *universal* solutions, that in a precise way represent all other solutions. In order to define those, we refer to the notion of homomorphism between instances.

Definition 2.2.1 *(Homomorphism) Let K_1 and K_2 be instances of the same schema $\mathbf{R}$. A* homomorphism *h from K_1 to K_2 is a function $h : dom(K_1) \rightarrow dom(K_2)$ such that: (1) $h(c) = c$ for every $c \in \mathsf{Const} \cap dom(K_1)$, and (2) for every $R \in \mathbf{R}$ and tuple $\bar{a} = (a_1, \ldots, a_k) \in R^{K_1}$, it holds that $h(\bar{a}) = (h(a_1), \ldots, h(a_k)) \in R^{K_2}$. Notice that this definition of homomorphism slightly differs from the usual one, as the additional constraint that homomorphisms are the identity on the constants is imposed.*

[1] We usually omit universal quantification in front of st-tgds, t-tgds, and e-gds.

Similar to homomorphisms between instances, a homomorphism h from a conjunctive formula $\phi(\bar{x})$ to an instance J is a mapping from the variables $\bar{x}$ to $\mathsf{Const} \cup \mathsf{Var}(J)$ such that for every atom $R(x_1, ..., x_n)$ of ϕ the fact $R(h(x_1), ..., h(x_n))$ is in J.

Let $\mathfrak{S}$ be a DE setting, I a source instance and J a solution for I under $\mathfrak{S}$. Then J is a *universal* solution for I under $\mathfrak{S}$, if for every solution J' for I under $\mathfrak{S}$, there exists a homomorphism from J to J'.

Example 2.2.2 *(Example 1.2.1 cont.) The target instance* $J = \{St(Alex, 18, \perp), Take(Alex, COMP1005, 80)\}$*, where* $\perp \in \mathsf{Var}$*, is a universal DE solution for the source instance* I *under* $\mathfrak{S}$*. However, a target instance* $J_1 = \{St(Alex, 18, Ottawa), Take(Alex, COMP1005, 80)\}$ *is only a DE solution for the source instance* I *since there does not exist a homomorphism* h *from* J_1 *to* J*.*

Given a fixed DE setting $\mathfrak{S}$, there exists a procedure $\texttt{CompUnivSol}_{\mathfrak{S}}$ [1] – based on the *chase* [26] – that computes a universal solution, called *canonical* universal solution, for each source instance I under $\mathfrak{S}$. This procedure is described below.

Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a fixed DE setting. The procedure $\texttt{CompUnivSol}_{\mathfrak{S}}$ takes as input a source instance I and works as follows. For each st-tgd in Σ_{st} of the form:

$$\exists \bar{y}\ \phi(\bar{x}, \bar{y}) \quad \rightarrow \quad \exists \bar{w}\ (T_1(\bar{x}_1, \bar{w}_1) \wedge \cdots \wedge T_k(\bar{x}_k, \bar{w}_k)),$$

where $\bar{x} = \bar{x}_1 \cup \cdots \cup \bar{x}_k$ and $\bar{w} = \bar{w}_1 \cup \cdots \cup \bar{w}_k$, and for each tuple $\bar{a}$ from $dom(I)$ of length $|\bar{x}|$, find all tuples $\bar{b}_1, \ldots, \bar{b}_m$ such that $I \models \phi(\bar{a}, \bar{b}_i)$, $i \in [1, m]$. Then choose m tuples $\bar{\perp}_1, \ldots, \bar{\perp}_m$ of length $|\bar{w}|$ of fresh distinct null values over Var. Relation T_i ($i \in [1, k]$) in the canonical universal solution for I contains tuples $(\pi_{\bar{x}_i}(\bar{a}), \pi_{\bar{w}_i}(\bar{\perp}_j))$, for each $j \in [1, m]$, where $\pi_{\bar{x}_i}(\bar{a})$ refers to the components of $\bar{a}$ that occur in the positions of $\bar{x}_i$.

For each t-tgd in Σ_t of the form:

$$\exists \bar{y}\ \phi(\bar{x}, \bar{y}) \quad \rightarrow \quad \exists \bar{w}\ (T_1(\bar{x}_1, \bar{w}_1) \wedge \cdots \wedge T_k(\bar{x}_k, \bar{w}_k)),$$

apply the same above step of st-tgd while checking for a homomorphism h from $\phi(\bar{x}, \bar{y})$ to J rather than I.

Finally, for each t-egd in Σ_t of the form:

$$\exists \bar{y}\ \phi(\bar{x}, \bar{y}) \quad \rightarrow \quad x_1 = x_2,$$

where x_1 and $x_2 \in \bar{x}$, and for each tuple $\bar{a}$ from $dom(I)$, if $\pi_{x_1}(\bar{a})$ is a null value over Var and $\pi_{x_2}(\bar{a})$ is a constant $c \in dom(J) \cap \mathsf{Const}^{\mathbf{T}}$, then for each occurrence of $\pi_{x_1}(\bar{a})$ in T_i, $i \in [1, k]$, replace $\pi_{x_1}(\bar{a})$ by c. If both $\pi_{x_1}(\bar{a})$ and $\pi_{x_2}(\bar{a})$ are null values over Var, then for each occurrence of $\pi_{x_1}(\bar{a})$ in T_i, $i \in [1, k]$, replace $\pi_{x_1}(\bar{a})$ by $\pi_{x_2}(\bar{a})$. However, if $\pi_{x_1}(\bar{a})$ is a null value over Var and $\pi_{x_2}(\bar{a})$ is a is a constant $c_1 \in dom(J) \cap \mathsf{Const}^{\mathbf{T}}$, then the chase fails and we say that I has no solution.

A chase sequence is a sequence of instances J_0, J_1,..., J_n such that every instance J_{i+1} in it is obtained from J_i by a chase step. A necessary condition identified in [1] for the chase to terminate is that Σ_t contains *weakly acyclic* set of target generating dependencies (tgds). Below we provide the definition of weakly acyclic as given in [1].

Definition 2.2.2 (*Weakly Acyclic Set of tgds*). *"Let Σ be a set of tgds over a fixed schema. Construct the $dependency\ graph$, a directed graph, as follows:*

1. *There is a node for every pair (R, A) with R a relation symbol of the target schema and A an attribute of R; call such a pair (R, A) a position;*

2. *for every tgd $\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}))$ in Σ and for every $x \in \mathbf{x}$ that occurs in ψ, and for every occurrence of the variable x in ϕ in position (R, A_i),*

   (a) *add an edge $(R, A_i) \rightarrow (S, B_j)$ (if it does not already exist) for every occurrence of x in ψ in position (S, B_j);*

   (b) *add a special edge $(R, A_i) \rightarrow (T, C_k)$ (if it does not already exists) for every existentially quantified variable y and for every occurrence of y in ψ in position (T, C_k).*

Note that there may be two edges in the same direction between two nodes, if exactly one of the two edges is special. Then, Σ is weakly acyclic if the dependency graph has no cycles going through a special edge".

In most data exchange work done in the literature, it was customary to exchange data from a source to a target based on st-tgds that consists of positive predicates and equality formulas. On the other hand authors in [27] introduced a simple work around that solved the problem of exchanging data using st-tgds, Σ_{st}, which contain negated predicates and/or inequality formulas. In their solution, they extended the source schema $\mathbf{S}$ to $\hat{\mathbf{S}} := \mathbf{S} \cup \{\hat{R} : \neg R \in \Sigma_{st}\} \cup \{N\}$. They also re-constructed Σ_{st} to $\hat{\Sigma}_{st}$ by replacing each negated literal $\neg R(\bar{x})$ and inequility $x \neq y$ over $\mathbf{S}$ with $\hat{R}(\bar{x})$ and $N(x, y)$ over $\hat{\mathbf{S}}$, respectively. In addition, they added to $\hat{\Sigma}_{st}$ the below set of dependencies:

1. if N appears in $\hat{\Sigma}_{st}$, then :

$$x = y \vee N(x, y)$$

$$x = y \wedge N(x, y) \rightarrow \bot$$

2. for each $\hat{R}(\bar{x})$ in $\hat{\Sigma}_{st}$:

$$R(\bar{x}) \vee \hat{R}(\bar{x})$$

$$R(\bar{x}) \wedge \hat{R}(\bar{x}) \rightarrow \bot$$

If the chase terminates successfully, then we say J_n is a canonical universal solution for I under $\mathfrak{S}$. Furthermore, each relation T_i, $i \in [1, k]$, in the canonical universal solution J_n for I only contains tuples that are obtained by applying this algorithm.

In general terms, canonical universal solutions are naïve tables, as they consist in general of both constants and nulls. Let K be a naïve table over schema $\mathbf{R}$. If ν is a mapping from the nulls mentioned in K into Const, we denote by $\nu(K)$ the instance that is obtained from K by simultaneously replacing each null $\bot \in dom(K)$ with $\nu(K)$. Then we define the set of

representatives of K, denoted by $\mathsf{Rep}(K)$, as:

$$\mathsf{Rep}(K) \quad = \quad \{K' \quad | \quad K' \text{ is an instance over } \mathbf{R}, dom(K') \subseteq \mathsf{Const} \text{ and } \nu(K) \subseteq K'\}.$$

In other words, $\mathsf{Rep}(K)$ consists of all those instances that do not contain nulls and that extend some instance that can be obtained from K by simultaneously replacing nulls with constants. This thesis tackles exchange settings where $\Sigma_t = \emptyset$.

It follows from [1] that for the class of data exchange settings where $\Sigma_t = \emptyset$, every source instance has universal solutions. Further, one can prove that for any fixed setting $\mathfrak{S}$ the *data* complexity of the procedure $\texttt{CompUnivSol}_{\mathfrak{S}}$ is in PTIME. This is a usual and reasonable assumption in data exchange [1], as instances tend to be much bigger than mappings. Summing up:

Theorem 2.2.3 *[1] Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_s t)$ be a fixed DE setting. Then,*

1. *Procedure* $\texttt{CompUnivSol}_{\mathfrak{S}}$ *runs in* LOGSPACE *if* $\Sigma_t = \emptyset$.
2. *Procedure* $\texttt{CompUnivSol}_{\mathfrak{S}}$ *runs in* PTIME *if* $\Sigma_t \neq \emptyset$

Authors in [28] proved that weak acyclic target tgds and fixed schema mappings are an essential condition for maintaining the tractability property of generating canonical universal solutions. This thesis also assumes fixed exchange settings.

DSE settings are an extension of DE settings where we take into consideration the difference in the domains of the source and target applications, which is a result of these being designed and implemented independently. It is possible that these applications use different vocabularies to refer to the same object, and it is possible that these applications require to exchange different yet related information.

The notion of a solution in data exchange differs depending on whether one adopts an *open-world-semantics* (OWS) [29, 23], a *closed-world-semantics* (CWS) [29, 16], or a mix of both approaches [30]. Given a DE setting $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ and a source instance I, a OWS has the following property: every target instance J such that $J \models \Sigma_{st} \cup \Sigma_t$ is a solution for

I under $\mathfrak{S}$. Intuitively, under the OWS, facts that are not explicitly stated to be true or false can be either true or false. On the other hand, a DE setting with a CWS states that each fact in a CWS solution must be directly justified by the source instance and $\Sigma_{st} \cup \Sigma_t$ of the data exchange setting.

Let us refer to Example 1.2.1. If the DE setting has the OWS property, the target instance $J_1 = \{St(Alex, 18, \perp), Take(Alex, CMP1005, 80), Take(Alex, CMP1004, 70)\}$, where $\perp \in$ Var, is considered to be a solution for I under $\mathfrak{S}$. On the other side, in a DE setting that follows the CWS property, J_1 is not considered to be a solution for I under $\mathfrak{S}$. The data exchange work done in [1, 10] followed the OWS approach, whereas the work in [21, 31] followed the CWS approach. We assume the OWS approach for the DSE setting.

Core universal solutions

As we mentioned before, in a DE setting $\mathfrak{S}$ canonical universal solutions are identified to be a class of "good" solutions to materialize in the target. Authors in [10] however have identified a class of "best" solutions in terms of compactness to materialize in a target instance and called those *core* solutions. A target instance J is a universal core solution for I under $\mathfrak{S}$ if there does not exists a target instance $J' \subset J$ such that J' is a universal DE solution for I under $\mathfrak{S}$.

Example 2.2.3 *(Example 1.2.1 cont.) The target instance J given in Figure 1.3 is a core solution for I under $\mathfrak{S}$. However, although the instance $J_1 = \{St(Alex, 18, \perp), Take(Alex, COMP1005, 80), St(Alex, 18, \perp_1)\}$, where $\perp$ and $\perp_1$ are in* Var*, is a universal solution for I in an OWS DE setting, yet J_1 is not a core solution for I under $\mathfrak{S}$. The reason is that $J \subset J_1$ and there exists a homomorphism $h : J \rightarrow J_1$.*

Authors in [10, 32] proved that the problem of generating a core is tractable. Summing up:

Theorem 2.2.4 *[32] Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a fixed DE setting. Then generating a core solution J for a source instance I under $\mathfrak{S}$ is in* LOGSPACE *if $\Sigma_t = \emptyset$. Otherwise, it is in* PTIME *if Σ_t contains a set of e-gds and a weakly acyclic set of t-tgds.*

They have also proved that universal core solutions for a source instance I are unique up to isomorphism. Several efforts have focused on providing algorithms that can be good candidates to be applied in practice for generating core solutions in DE settings [20, 12, 13, 14]. In a DSE setting, we call "most compact" solutions as minimal universal DSE KB-solutions, and those possess a minimum number of tuples and a minimum set of domain of constants. Our thesis designs algorithms that generate MUDSE solutions while leveraging the work done in [14] for generating target instances with minimum number of tuples.

Query answering Existing work on DE settings [1] adopted the notion *certain answer* in DE settings as a semantics for conjunctive query answering. This notion was initially discussed in [33, 34] and was first used by Bertossi et al. in [35, 36]. Then it was widely adopted in numerous contexts [37] including data integration [38], peer-to-peer systems [39, 40], and knowledge base systems [11, 41] as a semantics for query answering.

In data exchange settings [1, 10, 32], authors identified the class of universal solutions as good solutions to compute certain answers for unions of conjunctive queries Q, denoted by $\mathsf{certain}(Q, I)$, over a target schema. A conjunctive query Q is a restricted form of FO-logic that has the following general form: $Q(\bar{x}) = (\bar{x})\exists\bar{y}\phi(\bar{x}, \bar{y})$ where $\bar{x}$ is a set of distinguished variables and $\phi(\bar{x}, \bar{y})$ is a conjunction of predicate formulas. The set $\mathsf{certain}(Q, I)$ for a conjunctive query Q over a target schema $\mathbf{T}$, with respect to a source instance I, contains the set of tuples that exist in the evaluation of Q over every solution J of I. Authors in [1] addressed the problem of computing certain answers for the class of unions of conjunctive queries and unions of conjunctive queries with inequalities. They have proved that computing certain answers for unions of conjunctive queries and unions of conjunctive queries with one inequality is in PTIME. In addition, they showed that computing certain answers of conjunctive queries with two or more inequalities is in CONP.

We consider in this thesis the class of unions of conjunctive queries. We also adopt the set of DE certain answers as one possible semantics for conjunctive query answering, and we call it complete certain answers in the DSE setting. In addition, we design algorithms that compute those using both universal DSE solutions and universal DSE KB-solutions in LOGSPACE.

2.2.2 Data Exchange with Incomplete Source Information

The problem of representing incomplete information and the problem of querying incomplete relational databases has been an important research topic for a long time [34, 33, 23, 42, 11]. Two representation systems defined in [23] for incomplete information are of our interest: *v-tables*, and *c-tables*.

A v-table R is a conventional relational instance in which both variables from Var and constants from Const occur. Also, the representation of R is defined as the set:

$$\mathsf{Rep}(R) = \{\upsilon(R) \mid \upsilon : \mathsf{Var}(R) \rightarrow \mathsf{Const} \text{ is a valuation for the variables of R}\}$$

A c-table R is a v-table in which each tuple is associated with a boolean condition, that is a boolean combination of conjunctions and disjunctions of equality and inequality formulas that involve variables and constants. The representation of R is defined as the following set:

$$\mathsf{Rep}(R) = \{\upsilon(R) |\, \upsilon : \mathsf{Var}(R) \rightarrow \mathsf{Const} \text{ is a valuation for the variables of R such that each } \upsilon(R) \text{ contains a fact } \upsilon(t) \text{ only if the condition } \psi \text{ in t is such that } \upsilon(\psi) = true\}.$$

Solutions and universal solutions In the usual data exchange setting where completeness of the source instance is enforced, authors in [1] showed that for the class of mappings specified by st-tgds, v-tables form a strong representation system to represent universal solutions, and these were called *naïve* tables. Formally, naïve tables being a strong representation system in a DE setting means [1]: let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a DE setting with Σ_{st} as the set of st-tgds; it is true that for each source instance I_1 of $\mathbf{S}$, there exists a naïve instance J of $\mathbf{T}$ such that $\mathsf{Rep}(J) = Sol(I)$, where $Sol(I)$ is the set of all possible solutions for I. In other words, for each $J' \in \mathsf{Rep}(J)$, there exists $I' \in \mathsf{Rep}(I)$ such that $(I, J) \vDash \Sigma_{st}$.

On the other hand, in a more collaborative data exchange setting where a source instance is incomplete, authors of [11] showed that v-tables are not a strong representation system for the class of mappings specified by st-tgds. Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a data exchange setting

with Σ_{st} as the set of st-tgds and I a naïve instance of $\mathbf{S}$, authors of [11] have showed in a counter example that there exists a target naïve instance J such that $\mathsf{Rep}(J) \neq Sol(\mathsf{Rep}(I))$. The counter example is the following:

Example 2.2.4 *[11] let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a DE setting. Assume that $\mathbf{S}$ contains the relational symbol P and $\mathbf{T}$ contains the relational symbols T and R, and Σ_{st} consists of the following st-tgds:*

$$P(x, y) \rightarrow R(x, y), P(x, x) \rightarrow T(x).$$

Moreover, let I be a naïve instance of $\mathbf{S}$ such that $I = \{P(\perp, a)\}$ where $\perp \in \mathsf{Var}$ and $a \in \mathsf{Const}$. Then a naïve instance J cannot represent the fact that: $T(a)$ occur in a solution J' in $\mathsf{Rep}(J)$ only if J' is a solution for an instance I' in $\mathsf{Rep}(I)$ such that $\perp$ is assigned a value a in I'. □

It appears in [11] that the class of *pc-tables* – which are c-tables with conjunctions and disjunctions of equality conditions that involve variables and constants – form a strong representation system for the class of mappings Σ_{st} in DE settings where a source instance is incomplete.

An additional interesting feature about pc-tables is that they retain the tractability property when generating universal DE solutions. Therefore, given a fixed DE setting $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ and an incomplete source instance I, there exists an algorithm $\texttt{CompPctableUnivSol}_{\mathfrak{S}}$ based on the chase [26] that can generate a universal DE solution for I under $\mathfrak{S}$. This procedure works as follows:

For each st-tgd in Σ_{st} of the form:

$$\phi(\bar{x}, \bar{y}) \wedge \theta(\bar{x}', \bar{y}') \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z}),$$

where $\phi(\bar{x}, \bar{y}) = R_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge R_m(\bar{x}_m, \bar{y}_m)$, $\psi(\bar{x}, \bar{z}) = T_1(\bar{x}_1, \bar{z}_1) \wedge \cdots \wedge T_n(\bar{x}_n, \bar{y}_n)$, and

the formula $\theta(\bar{x}', \bar{y}')$ is a conjunction of equalities from variables in $\bar{x}$ and $\bar{y}$, and for each tuple $\bar{a}$ in $dom(I)$ of length $|\bar{x}|$, do the following: find all tuples $\bar{b}_1, \ldots, \bar{b}_f$ of length $|y|$ such that $I \vDash \phi(\bar{a}, \bar{b}_i)$, $1 \leq i \leq f$, then choose f tuples $\bar{\perp}_1, \ldots, \bar{\perp}_f$, and finally generate the tuples $\psi(\bar{a}, \bar{\perp}_i)$, $1 \leq i \leq f$, such that: each tuple $T_j(\pi_{\bar{x}_j}(a_j), \pi_{\bar{z}_j}(\bar{\perp}_j))$, $1 \leq j \leq n$, is associated with a condition $\bigwedge_{k=1}^{k=m} \rho_{R^k} \wedge \theta(\bar{\pi}_{x'}(\bar{a}), \pi_{\bar{y}'}(\bar{b}_j))$.

Example 2.2.5 *[11] (Example 2.2.4 cont.) Applying algorithm* $\texttt{CompPctableUnivSol}_{\mathfrak{S}}$ *to* I *generates the target pc-table* $J = \{R(\perp, a : true), T(n : n = a)\}$ *as a universal DE solution for* I *under* $\mathfrak{S}$*; thus* $\mathsf{Rep}(J) = Sol(\mathsf{Rep}(I))$. □

This present thesis defines a DSE_E setting with incomplete information as an instance of a knowledge exchange setting where source knowledge bases with incomplete information are exchanged. Such a DSE setting extends ordinary data exchange settings to solve the problem of exchanging source knowledge bases with incomplete information into a target knowledge base with pc-tables. The thesis adopts the above mentioned chase algorithm to generate DSE KB-solutions. It also identifies the structure that best represents the most compact of those. Finally, the thesis investigates the data complexities to generate these solutions.

Query answering Authors in [11, 22] investigated the problem of computing certain answers from instances that contain incomplete information. Following the work in [11], let $\mathfrak{S}$ be a DE mapping from $\mathbf{S}$ to $\mathbf{T}$, I be an incomplete instance over $\mathbf{S}$, and Q be a conjunctive query over $\mathbf{T}$, then the set of certain answers of Q over I under $\mathfrak{S}$, denoted by $\mathsf{certain}_{\mathfrak{S}}(Q, I)$, is defined as:

$$\bigcap Q(J) : \text{for each solution } J \text{ for } I \text{ under } \mathfrak{S}$$

2.3 Data Coordination

A data coordination setting allows data sharing between independent applications by providing access to data that reside in independent sources without having to exchange it and while maintaining their autonomy. In addition, a data coordination setting maintains the consistency of

data that resides in different sources by allowing the propagation of necessary updates between these sources.

Data coordination has been mostly studied in *peer-to-peer* (P2P) networks. Peer-to-peer data management systems consist of a set of sources, called *peers*, which are connected by pairwise mappings – named *acquaintances* – which facilitate the propagation of translated queries and transformed information among peers. All peers in a P2P network possess equal capabilities and responsibilities in terms of propagating changes and retrieving related information.

One example of a peer data management domain that has been studied previously in the literature is the health care domain where family physicians, walk-in clinics, hospitals, medical laboratories, pharmacists, and further stakeholders are willing to share information about patients treatments, medications, and test results. In such a domain, updates in one database may need to be propagated to other databases to maintain the consistency of data. In addition, patients information should be accessible for view to each of those domains.

The vision of Peer data management systems was first introduced in [4] and theoretical foundations of it was provided in [43]. Several peer data management frameworks were proposed in literature, such as Piazza [44], Hyperion [7], and PeerDB [45]. In practice, applications of different peers are implemented separately and independently in a peer data management system. Therefore, elements of these applications can end up being represented using different vocabularies. Peer data management systems, and specifically data sharing systems [7] addressed this issue by introducing instance level mappings that associate elements of one peer to corresponding data elements of each acquainted peer.

Formally, a general peer data management system is a tuple $\mathfrak{S} = (\mathcal{P}, \Sigma_p, \Sigma_t)$ where $\mathcal{P}$ is a set of peers $\{P_1, ..., P_n\}$ such that schemas $\mathbf{S}_i$ of P_i, $1 \leq i \leq n$ are pairwise disjoint, Σ_p is a set of peer-to-peer mappings, and Σ_t is a set of local constraints in the form of egds and tgds. Different peer data management systems existing in the literature defined Σ_p differently. One class of peer data management systems defined Σ_p as assertions at a schema level [46, 3, 47, 48], and a second class of data management systems involved instance-level mappings in

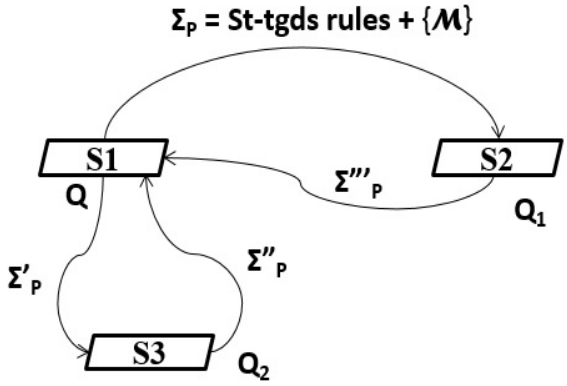

Figure 2.2: A Data Coordination Setting

Σ_p [6, 5, 49, 4, 7, 9]. We are interested in the latter class of data coordination settings. We illustrate a DC setting in Figure 2.2 in which Q, Q_1, and Q_2 are conjunctive queries.

2.3.1 Data coordination with schema-level mappings

In this class of peer data management systems [46, 3, 47, 48], coordination of data is facilitated though the use of mappings between schema elements to allow the propagation of updates and insertions between peers. The mapping of schema elements that allows the rewrite of queries to integrate data.

For example, authors in [47] used schema mappings – they called those coordination formulas – that are a generalization of GLAV mappings that contain expressions of the form $i : \phi$, which means ϕ is true in database i. The formula $\forall i : x.A(x)$ is interpreted as "for all elements of the domain $dom(i)$, A is true". Likewise, $\exists i : x.A(x)$, is read as "there is an element in the domain $dom(i)$ such that A is true".

2.3.2 Data coordination with instance-level mappings

This class of peer data management systems [6, 5, 49, 4, 7, 9] allow the sharing of data in a network of peers that are linked by *mapping tables* and by *mapping expressions*. Mapping tables are instance-level mappings which allow integrating information of seemingly uncon-

nected databases. On the other hand, mapping expressions are schema level mappings which allow the propagation of updated information among connected peers.

Mapping tables store a set of *data mappings* which list pairs of corresponding values between instances of two peers. More formally, Let X and Y be non-empty disjoint sets of attributes in schemas S_1 and S_2 respectively. A mapping table $\mathcal{M}$ from X to Y is a finite set of tuples t over $X \cup Y$ such that: (1) for each attribute $A \in X \cup Y$, $t[A]$ is either a constant in Const, a variable $v \in \mathsf{Var}$ – where v represents an unknown or null value –, or an expression of the form $v - S$ where $v \in V$ and S is a finite subset of $dom(A)$ – which indicates an unknown value that is different than all the elements in S; and (2) each variable appears in at most one mapping.

The data coordination instance given in Example 1.2.2 shows how mapping tables allow integrating information of different sources regardless of whether this data belong to the same or to different worlds of vocabularies.

Notice that the flexibility of mapping tables stems from the fact that they do not require peers to disclose their schemas and they can be established between peers that belong to different worlds. Some coordination systems [5] relied solely on mapping tables to define mappings between peers. In this work, authors defined a query rewriting mechanism that uses mapping tables to translate a locally expressed query at one peer to a set of queries over the connected (or acquainted) peers. They distinguished between two types of query translation algorithms: one algorithm generates sound certain answers (which contain a portion of the complete set of correct answers) while the second generates complete certain answers (which intuitively contain the complete set of correct answers).

In its simplest form, mapping tables are just binary tables containing pairs of corresponding identifiers from two different sources. Formally, given two domains D_1 and D_2, not necessarily disjoint, a mapping table over $(\mathsf{D}_1, \mathsf{D}_2)$ is nothing else than a subset of $\mathsf{D}_1 \times \mathsf{D}_2$. Intuitively, the fact that a pair (d_1, d_2) belongs to the mapping table implies that source value $d_1 \in \mathsf{D}_1$ *corresponds* to a target value $d_2 \in \mathsf{D}_2$. Notice that the exact meaning of "correspondence" between values differs with different applications.

In a DSE setting, we adopt this simple form of mapping tables, we refer to the first attribute as source attribute and the second one as target attribute, and we study how different semantics of mapping tables can have an effect on the definition of semantics of a data sharing and exchange setting. In addition, in a DSE setting, coordination between a source application and a target application is defined by mapping rules that incorporate associations at the schema level as well as the data level, and not only at the data level.

2.4 Knowledge Base Exchange

Knowledge bases have been used to represent various types of data including ontologies in the semantic web, which are expressed using different types of formalisms like OWL [50], RDF [51], and DL [52]. A big bulk of work has recently tackled the problems of reasoning, data access [53, 54, 55, 56, 57, 58], and exchange [41, 59, 60] of knowledge bases that are defined using different Description Logic (DL) languages. DLs is an interesting family of knowledge representation languages that form fragments of First Order Logic and which use two-variable fragments and sometimes counting quantifiers, and with which logical inferencing is decidable.

Authors in [11] extended the work done in [1, 10] and introduced a general framework – called knowledge bases (KB) exchange – for data exchange where a source possesses additional to explicit data, implicit information in the form of rules (full tgds, which are tgds that do not use existential quantication), that can infer new data to be exchanged to the target. We applied this KB exchange framework in both DSE settings that we introduced in Chapter 4 and Chapter 5 and we defined their semantics. In addition, authors in [59] showed how to map a relational database that contains integrity constraints of type primary keys, foreign keys, and non-null constraints to an OWL-based knowledge base. In our work however, we consider containment constraints over the source and target schemas, and we generate target instances J as KB-solutions that store a portion of the explicit data exchanged from a source instance I which satisfy a set of rules, in the form of full tgds, that represents the implicit data necessary to complete I with the implicit information entailed by the semantics of the st-mapping table

$\mathcal{M}$.

Knowledge bases

Formally, a knowledge base [11] over schema $\mathbf{R}$ is a pair (K, Σ), where K is an instance of $\mathbf{R}$ (the explicit data) and Σ is a set of logical sentences over $\mathbf{R}$ (the implicit data). The set of *models* of (K, Σ), denoted by $\mathsf{Mod}(K, \Sigma)$, is defined as the set of instances of $\mathbf{R}$ that contain the explicit data in K and satisfy the implicit data in Σ; We now formally define the notion of knowledge exchange and the main concepts associated with it. Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a knowledge exchange setting. Let (I, Σ_s) be a KB over $\mathbf{S}$ and (J, Σ_t) a KB over $\mathbf{T}$. Then:

1. (J, Σ_t) is a KB-solution for (I, Σ_s) under $\mathfrak{S}$, if for every $J' \in \mathsf{Mod}(J, \Sigma_t)$ there is $I' \in \mathsf{Mod}(I, \Sigma_s)$ such that $(I', J') \models \Sigma_{st}$.

2. In addition, (J, Σ_t) is a *universal* KB-solution for (I, Σ_s) under $\mathfrak{S}$, if for every $I' \in \mathsf{Mod}(I, \Sigma_s)$, and for every J' such that $(I', J') \models \mathfrak{S}$, we have $\mathsf{Rep}(J') \subseteq \mathsf{Rep}(J)$.

Authors in [11] have shown that in a fixed knowledge exchange setting where the source implicit knowledge, defined as a set of tgds, are of type *full tgds*, there exists an algorithm, based on the chase, that generates a target KB-solution in polynomial time. We summarize the result as follows:

Theorem 2.4.1 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a fixed knowledge base exchange setting. Also, let Σ_s and Σ_t be fixed sets of rules, of type full tgds, in the source and target Knowledge bases respectively. Then there exists a* LOGSPACE *procedure that computes, for (I, Σ_s) under $\mathfrak{S}$ a target knowledge base (J, Σ_t).*

In our case, we consider in both DSE_U and DSE_E settings a set of rules of type full tgds in both the source and the target KBs.

In the context of a knowledge base exchange settings, applying the DE concept of a core to materialized explicit data is not enough to generate a "best" KB-solution in terms of compactness. Therefore, authors in [11] defined a minimal KB-solution as follows:

Definition 2.4.2 *Minimal Universal KB Solution: Let* $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ *be a KB exchange setting,* I *be a source instance and* J *a universal KB solution for* I *under* $\mathfrak{S}$*. Then* J *is a* minimal *universal KB solution for* I *under* $\mathfrak{S}$*, if there is no proper subset* J' *of* J *such that* J' *is a universal KB solution for* I *under* $\mathfrak{S}$*.*

The same concept applies to DSE_U and DSE_E settings. We show in Chapter 4 that there exist core universal KB-solutions that are not the most compact KB-solutions in a DSE_U settings.

In a DSE setting, we extended Definition 2.4.2 to define minimal universal DSE KB-solutions as those with minimal number of tuples and minimal domain of constants. Such class of KB-solutions possess the "best" properties in terms of compactness and for query re-writing in DSE settings.

Several approaches tackled the problem of knowledge base exchange [11, 61, 60, 41]. Knowledge base exchange in Description logics showed that adopting ordinary DE universal solutions present several limitations [61, 60, 41] since these can miss some semantics of the source KB. For this reason, authors in [60] introduced a weaker notion of a solution called universal CQ-solutions that are considered as more suitable for query answering. In our work, however, applying the ordinary exchange solution [1] to a source instance that is complete with respect to the semantics of the st-mapping table will generate universal DSE solutions that reflect the semantics in the source accurately, and thus do not possess the limitations exposed in DLs.

Query Answering

In the context of conjunctive query answering in knowledge bases, different approaches were proposed for computing certain answers in DL KBs. Some of those approaches adopt pure query re-writing methods [56], and some others adopt combined methods [58, 57] which basically first expand an ABox $\mathcal{A}$ with additional individuals based on a TBox $\mathcal{T}$ (and independently of an input conjunctive query Q), and then rewrites Q to a query Q' to generate the set of certain answers. It was shown in the literature that the later approach has the advan-

tage of smaller and transparent rewritings of queries and it outperforms pure query rewriting techniques.

In our work, to compute certain answers in DSE_U and DSE_E settings, we adopt a combined approach similar to the work done in [58, 57] and we materialize target elements entailed to be related by the st-mapping table $\mathcal{M}$ in fresh tables named RELATED and EQUAL respectively. Then we rewrite a conjunctive query Q using RELATED/EQUAL to retrieve the set of certain answers of Q. In DSE_E settings, MUDSE solutions possess specific interesting properties, as we mention in chapter 5, such that the query re-writing is much simpler than in DSE_U since the EQUAL table is only joined with distinguished variables in Q when generating the complete set of certain answers, and it is not used when generating the sound set of certain answers.

2.5 Summary

In this chapter, we summarized the state of the art research work in the areas of data exchange for complete and incomplete source data, data coordination and sharing, and knowledge exchange. We first presented basic notations and technical preliminaries related to those fields that we shall use in the following chapters of this thesis. Next, we briefly discussed the semantics of data exchange settings and defined DE solutions and the most compact of those, called core solutions. We also stated the semantics of conjunctive query answering and reviewed the definition of certain answers. Then we reviewed the mapping table construct introduced in data coordination settings, and showed how it is used in query re-writings to integrate data from independent applications. Then, we discussed knowledge base exchange settings. We saw that a knowledge base exchange setting extends an ordinary data exchange settings with a set of rules defined in source and target applications, to infer additional information. Furthermore, we reviewed the semantics of a knowledge exchange setting and the notion of knowledge base solutions. Finally, we summarized the semantics of query answering and introduced the definition of certain answers in knowledge base exchange settings.

Chapter 3

Data Sharing and Exchange

3.1 Introduction

Before we introduce the formal definition and the intuition behind the different components of a DSE setting, we start this chapter with some terminology and notations that we will use in this and the following chapters.

As is the case in usual DE settings, instances of $\mathbf{S}$ in a DSE setting are called source instances and instances of $\mathbf{T}$ are called target instances. We denote source instances by $I, I', I_1, \ldots$ and target instances by $J, J', J_1, \ldots$. Instances of the *source-to-target* mapping (st-mapping) relation $\mathcal{M}$ are called st-mapping tables. By slightly abusing notation, we denote st-mapping tables also by $\mathcal{M}$.

We assume in a DSE setting the existence of two (not necessarily disjoint) countably infinite sets of constants $\mathsf{Const}^{\mathbf{S}}$ and $\mathsf{Const}^{\mathbf{T}}$. $\mathsf{Const}^{\mathbf{S}}$ denotes the set of source constants and $\mathsf{Const}^{\mathbf{T}}$ denotes the set of source and target constants. We distinguish between these two set of constants, since applications that collaborate at the data level usually have different domains of constants (i.e. use different vocabularies). As in the case of usual data exchange, we also assume the existence of a countably infinite set Var of labelled nulls (that is disjoint from both $\mathsf{Const}^{\mathbf{S}}$ and $\mathsf{Const}^{\mathbf{T}}$).

A DSE is not different from usual data exchange in the fact that the domain of a source

instance I is always contained in $\mathsf{Const}^{\mathbf{S}}$, while the domain of a target instance J is allowed to contain labelled nulls and thus can belong to $\mathsf{Const}^{\mathbf{T}} \cup \mathsf{Var}$. Also, each st-mapping table $\mathcal{M}$ is a mapping table over $(\mathsf{Const}^{\mathbf{S}}, \mathsf{Const}^{\mathbf{T}})$; that is, a subset of $\mathsf{Const}^{\mathbf{S}} \times \mathsf{Const}^{\mathbf{T}}$. This explains, as we see in Definition 3.2.1, why we call the first attribute of $\mathcal{M}$ the source attribute, and the second one the target attribute. Thus, the intuition behind source-to-target mapping tables is that they identify, for each source constant, the set of related target constants (that is, they insure coordination at the data level).

In ordinary DC settings, peers can be connected by a set of one or more mapping tables that contain both constants and null values. In a DSE setting however, we used a single st-mapping table that contains only constant values for ease of discussions and clarity of proofs.

3.2 Data Sharing and Exchange Setting

We define data sharing and exchange as the problem of transforming data between independent applications that possess different schemas and different domains of constants, i.e. restructuring data which conform to the schema and vocabulary of one application (source) and generating the corresponding set of data and store it in an independent application (target) while conforming to the schema and the vocabulary of the target. The transformation procedure should be performed according to the set of st-tgds between $\mathbf{S}$ and $\mathbf{T}$ and to the semantics of the data mappings in the st-mapping table $\mathcal{M}$. We illustrate a DSE setting in Figure 3.1. In order to solve this problem we define a new class of settings that extend DE settings with the ability to collaborate at the data level besides the schema level. More formally, we introduce DSE settings as follows:

Definition 3.2.1 (DSE Setting) *A* data sharing and exchange *setting is a tuple* $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$*, where:*

- $\mathbf{S}$ *and* $\mathbf{T}$ *are a source and a target schema, respectively;*
- $\mathcal{M}$ *is a binary relation symbol that appears neither in* $\mathbf{S}$ *nor in* $\mathbf{T}$*, and that is called a*

Figure 3.1: A Data Sharing and Exchange Setting

source-to-target mapping (*we call the first attribute of $\mathcal{M}$ the source attribute S and the second one the target attribute T); and*

- Σ_{st} *consists of a set of* mapping *st-tgds, which are* FO *sentences of the form*

$$\forall \bar{x} \forall \bar{y} \forall \bar{z} \, (\phi(\bar{x}, \bar{y}) \wedge \mu(\bar{x}, \bar{z}) \; \rightarrow \; \exists \bar{w} \, \psi(\bar{z}, \bar{w})),$$

[1] *where (i) $\phi(\bar{x}, \bar{y})$ and $\psi(\bar{z}, \bar{w})$ are conjunctions of relational atoms over* $\mathbf{S}$ *and* $\mathbf{T}$*, resp., (ii) $\mu(\bar{x}, \bar{z})$ is a conjunction of atomic formulas that only use the relation symbol $\mathcal{M}$, (iii) $\bar{x}$ is the tuple of variables that appear in $\mu(\bar{x}, \bar{z})$ in the positions of source attributes of $\mathcal{M}$, and (iv) $\bar{z}$ is the tuple of variables that appear in $\mu(\bar{x}, \bar{z})$ in the positions of target attributes of $\mathcal{M}$.*

The intuition behind DE st-tgds is that they state how source data has to be transformed to corresponding target data (that is, they state coordination at the schema level). However, since in the DSE scenario we are interested in transferring data based also on the correspondence between source and target constants given by the semantics of the st-mapping table that interprets $\mathcal{M}$, the mapping st-tgds extend usual DE st-tgds with a conjunction μ that filters the target data that is related via $\mathcal{M}$ with the corresponding source data.

More formally, given a source instance I and a st-mapping table $\mathcal{M}$, the mapping st-tgd

$$\phi(\bar{x}, \bar{y}) \wedge \mu(\bar{x}, \bar{z}) \; \rightarrow \; \exists \bar{w} \psi(\bar{z}, \bar{w})$$

[1] We usually omit universal quantications in front of st-tgds

Student

Sname	Sage
Alex	18
Tom	19

Course

Cid	Cname	Pname
COMP3005	Database Management Systems	CS
COMP 4001	Distributed Computing	CS

Enroll

Sname	Cid	Cgrade
Alex	COMP3005	80
Tom	COMP 4001	70

Figure 3.2: Source instance I

enforces the following: whenever $I \models \phi(\bar{a}, \bar{b})$, for a tuple $(\bar{a}, \bar{b})$ of constants in $\mathsf{Const}^{\mathbf{S}} \cap dom(I)$, and the tuple $\bar{c}$ of constants in $\mathsf{Const}^{\mathbf{T}}$ is related to $\bar{a}$ via μ (that is, $\mathcal{M} \models \mu(\bar{a}, \bar{c})$), then it must be the case that $J \models \psi(\bar{c}, \bar{d})$, for some tuple $\bar{d}$ of elements in $dom(J) \cap (\mathsf{Const}^{\mathbf{T}} \cup \mathsf{Var})$, where J is the materialized target instance.

We assume in our work that each value $c \in \mathsf{Const}^{\mathbf{S}} \cap dom(\mathcal{M})$ cannot be associated but with the target values $c' \in \mathsf{Const}^{\mathbf{T}} \cap dom(\mathcal{M})$ that are mapped to it in the st-mapping table $\mathcal{M}$. Also, if an element $d \in \mathsf{Const}^{\mathbf{S}}$ is not associated with any target value in $\mathcal{M}$, then source tuples that contain this value will not be exchanged to the target. We refer again to the example we mentioned previously about universities that accept students' transfers from different foreign universities. We show in the following example one way to represent this scenario in a DSE setting.

Example 3.2.1 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE setting. Suppose that $\mathbf{S}$ in $\mathfrak{S}$ is the schema of UOC and $\mathbf{T}$ in $\mathfrak{S}$ is the schema of UOO. Consider the database instance given in Figure 3.2 to be the source instance I, and let the st-mapping table $\mathcal{M}$ in $\mathfrak{S}$ be the one given in Figure 3.3. We assume that $\mathcal{M}$ in this example specifies for each major 'CS' course at UOC, the equivalent major 'CS' course and the set of its pre-requisite courses at UOO. Finally, let Σ_{st} consist of the following st-mapping dependencies:*

(a) $St(x, y, z) \wedge Take(x, w, u) \wedge \mathcal{M}(x, x') \wedge \mathcal{M}(y, y') \wedge \mathcal{M}(w, w') \wedge \mathcal{M}(u, u')$

M

S	T
COMP 4001	CSI4109
COMP 4001	CSI3105
COMP 4001	CSI2110
COMP 3005	CSI3130
COMP 3005	CSI2132
COMP 3005	CSI2110
CS	CS
ENG	ENG
Tom	Tom
Alex	Alex
18	18
19	19
80	B
70	C

Figure 3.3: St-Mapping Table $\mathcal{M}$

$$\rightarrow Student(x', y') \wedge Enroll(x', w', u').$$

It is clear that this DSE instance is exchanging information from the UOC source to the UOO target about UOC students and the list of major 'CS' courses that they completed in UOC. Also, intuitively, a target instance in this DSE instance should reflect what pre-requisite courses at UOO that a UOC student is considered to have completed according to the st-mapping table $\mathcal{M}$. □

One possible solution for the DSE instance given in Example 3.2.1 would be a target instance which for each set of source tuples exchanged, contains the complete set of corresponding target tuples specified by $\mathcal{M}$. We call such type of solutions as *DSE solutions*.

3.3 DSE Solutions

We formally define DSE solutions as follows:

Definition 3.3.1 (DSE Solution) . *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE setting, Σ_{st} be a set of st-tgds, I an instance of $\mathbf{S}$, and $\mathcal{M}$ an st-mapping table. Then,*

- *J is a DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$ if $\langle (I \cup \{\mathcal{M}\}), J\rangle \vDash \Sigma_{st}$*

- *J is a* universal *DSE solution for I and $\mathcal{M}$ if there exists a homomorphism $h : J \to J'$ for every DSE solution J' for I and $\mathcal{M}$ in $\mathfrak{S}$.*

Generally, in a DSE setting $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$, one can easily define a procedure – we call it $\texttt{CompUnivSol}_{\mathfrak{S}}$ – that takes as input a source instance I and a st-mapping table $\mathcal{M}$ and compute a universal DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$. The procedure first transforms the DSE setting $\mathfrak{S}$ into the data exchange setting $\mathfrak{S}' = \big((\mathbf{S} \cup \{\mathcal{M}\}), \mathbf{T}, \Sigma_{st}\big)$. Then it computes $J := \texttt{CompUnivSol}_{\mathfrak{S}'}(I \cup \{\mathcal{M}\})$. It is clear that J is a universal DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$. Further, it follows from [62] that, for a fixed DSE setting $\mathfrak{S}$, the procedure $\texttt{CompUnivSol}_{\mathfrak{S}}$ works in logarithmic space. Summing up, we obtain the following useful proposition:

Proposition 3.3.2 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a fixed DSE setting. The procedure $\texttt{CompUnivSol}_{\mathfrak{S}}$ computes, for a source instance I and an st-mapping table $\mathcal{M}$, a universal DSE solution J for I and $\mathcal{M}$ under $\mathfrak{S}$ in* LOGSPACE.

Following the concept of canonical DE universal solutions, the result of $\texttt{CompUnivSol}_{\mathfrak{S}}(I, \mathcal{M})$ is called a canonical universal DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$.

Example 3.3.1 *In reference to the DSE instance given in Example 3.2.1, it is intuitive that the solution J given in Figure 3.4 is a canonical universal DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$. We see in this example how a DSE setting solves the issue of heterogeneous view of integrated information which arises in a DC setting as we explained in Example 1.2.2, and provides us with universal DSE solutions which contain a homogeneous view of the integrated data. With this observation, we can say that the DSE solution J solves the issue that existed in DC settings and allows users easily to compare the performance of the students Tom and $Alex$ in course $CSI2110$.*

□

Although DSE solutions succeed to provide a consolidated view of exchanged data by referring to same elements using the same target values, as we saw in Example 3.3.1, we

Student

Sname	Sage	Saddress
Alex	18	$\perp_1$
Tom	19	$\perp_2$

Enroll

Sname	Cid	Cgrade
Tom	CSI4109	C
Tom	CSI3105	C
Tom	CSI2110	C
Alex	CSI3130	B
Alex	CSI2132	B
Alex	CSI2110	B

Figure 3.4: Universal DSE Solution J

are interested in DSE settings to provide users the capability of computing a portion of the set of correct answers – called sound certain answers in DC settings [5] – rather than the complete set – called complete certain answers in DC settings [5]. As explained previously in data sharing applications [8], sound certain answers of some conjunctive queries can be informative enough for users to escape the more expensive computation of complete answers for these queries. For this reason, authors in [49] introduced two types of certain answers as semantics for conjunctive query answering. Those are *sound certain answers* and *complete certain answers*.

As we show later in Chapters 4 and 5, some DSE settings $\mathfrak{S}$ with specific properties of st-mapping tables $\mathcal{M}$ can exhibit a fast increase in the size of universal DSE solutions if the multiplicity of $\mathcal{M}$ is *one-to-many* or *many-to-many* and the size of $\mathcal{M}$ increases. For example, exchanging a source table $R(A, B)$ having 1000 records to a target table $R'(A, B)$, where each element in R is mapped to 3 distinct target values in $\mathcal{M}$, will generate 9000 records in R'. For this reason, and to deal with this problem the same way as in DC settings [49], we will adopt in Chapters 4 and 5 sound and complete certain answers as semantics for conjunctive queries.

To allow users in a DSE setting to compute both types of certain answers, we use the concept of a knowledge base instance introduced in [11] and define target instances as part of the semantics of a DSE setting that store a portion of the set of exchanged data instead of the complete set, such that the former reflect the source instance and the st-mapping table

accurately. To support this type of target solutions, we will adopt the knowledge base exchange framework introduced in [11], to represent a DSE setting and we will call target instances in this representation *DSE KB-solutions*. DSE KB-solutions possess a rich set of implicit data in the form of rules such that when these rules are applied to a universal DSE KB-solution J, the remaining set of exchanged information can be recomputed and a universal DSE solution can be regenerated.

Implicit data in a universal DSE KB-solution differ with different semantics of an st-mapping table $\mathcal{M}$ in a DSE setting. In Chapters 4 and 5 of this thesis, we discuss two different DSE settings with different semantics of st-mapping tables $\mathcal{M}$. In each one of these DSE settings, we define a different set of logical rules that serve as implicit information necessary for generating universal DSE solutions whenever we are given universal DSE KB-solutions. In addition, we present in Chapter 5 a DSE setting with specific interpretation of related data in an st-mapping table $\mathcal{M}$ which makes it different than ordinary DE settings. The reason is these specific semantics of st-mapping tables entail additional information in the source instance and the st-mapping table which did not exist in the first place. Therefore, representing this DSE setting using the knowledge base exchange framework of [11] will prove to be the natural way to go.

3.4 Summary

This chapter introduced formally data sharing and exchange settings that allow collaboration between a source and a target at both the schema and the instance level. Similar to universal solutions in the usual DE setting, the chapter formally defined one type of solutions that we called universal DSE solutions: a universal DSE solution is a solution that contains for each source tuple exchanged, the complete set of corresponding target tuples. The next two chapters introduce two different specific DSE settings, each one with a specific property of st-mapping tables $\mathcal{M}$, namely the DSE with unique identity semantics and the DSE with equality semantics. These next two chapters also show how to generate universal DSE KB-solutions and how

to compute both sound and complete certain answers for conjunctive queries using the latter.

Chapter 4

DSE with Unique Identity semantics

The present chapter considers the DSE setting $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$, introduced in Definition 3.2.1 by giving a general interpretation of related data in an st-mapping table $\mathcal{M}$. The general interpretation is similar to the semantics adopted in ordinary DC settings [4, 6, 7], which are reviewed in Chapter 3. The main intuition behind this semantics is as follows: a pair (a, b) holds in an st-mapping table $\mathcal{M}$ if a is related to b, with no particular meaning of this relationship. We will show in what follows a possible way of representing this setting that gives users the privilege to compute both a portion as well as the complete set of correct answers for conjunctive queries, following the intuition of sound and complete certain answers, respectively, from ordinary DC settings [5, 6] .

4.1 Data Sharing and Exchange as Knowledge Base Exchange

Suppose a source instance I and an st-mapping table $\mathcal{M}$ in a DSE setting $\mathfrak{S}$. Informally, we define a complete certain answer of a conjunctive query q in a DSE setting $\mathfrak{S}$ to be the set of answers of q that exist in every DSE solution J of I and $\mathcal{M}$ under $\mathfrak{S}$, as given in Definition 3.3.1. A sound certain answer of q, on the other hand, is a portion of the complete certain answer with particular properties that allows us to deduce the complete set. As we have mentioned earlier, a source element in an st-mapping table $\mathcal{M}$ can be mapped to one or more

corresponding elements in the target. Therefore, for each tuple exchanged using the exchange process explained in Chapter 3, all corresponding tuples are generated in a universal DSE solution. As a direct consequence, applying a conjunctive query to a universal DSE solution generates the complete set of corresponding tuples in a certain answer. However, as we shall explain later and as discussed in DC settings [5], for some queries it is informative enough to receive a portion of the set of certain answers that corresponds to the full set. We give below an example that illustrates the concepts of sound and complete certain answers:

Example 4.1.1 *We refer in this example to the DSE instance given in Example 3.2.1, the st-mapping table given in Figure 3.3, and the universal DSE solution J depicted in Figure 3.4. Let $q(x, y, z, u, v) = Student(x, y, z) \wedge Enroll(x, u, v)$ be a conjunctive query.*

A complete certain answer of q is: $q(J)$ = $\{(Tom, 19, \perp_2, CSI4109, C)$, $(Tom, 19, \perp_2, CSI3105, C)$, $(Tom, 19, \perp_2, CSI2110, C)$, $(Alex, 18, \perp_1, CSI3130, B)$, $(Alex, 18, \perp_1, CSI2132, B)$, $(Alex, 18, \perp_1, CSI2110, B)\}$.

A possible sound certain answer of q on the other hand would be $q(J) = \{(Tom, 19, \perp_2, CSI4109, C)$, $(Alex, 18, \perp_1, CSI3130, B)\}$.

Given the fact that courses $CSI4109$, $CSI3105$, and $CSI2110$ are related to the same source course $COMP4001$ in $\mathcal{M}$, and with the sound answer $\{(Tom, 19, \perp_2, CSI4109, C)\}$ of q, we can deduce that Tom is considered as completed courses $CSI3105$, and $CSI2110$ as well in UOO with grade C. The same reasoning applies to $Alex$.

To support generating both types of certain answers, we represent a DSE setting using the knowledge base exchange framework introduced in [11], and we consider that the st-mapping table $\mathcal{M}$ has the following property: each source constant value a in $\mathcal{M}$ is mapped to at least one target constant value b in $\mathcal{M}$ that uniquely identifies a. More formally, we assume that $\mathcal{M}$ satisfies the following constraint:

$$\forall x \exists y \forall z (\mathcal{M}(x, y) \wedge \mathcal{M}(z, y) \rightarrow x = z) \tag{4.1}$$

M

S	T
COMP 4001	CSI4109
COMP 4001	CSI3105
COMP 4001	CSI2110
COMP 3005	CSI3130
COMP 3005	CSI2132
COMP 3005	CSI2110
COMP 1006	CSI1308

Figure 4.1: An st-mapping table in a DSE$_U$ setting

This property of the st-mapping table leads us to call the corresponding DSE setting **DSE with unique identity semantics**, denoted by *DSE*$_U$. A sample st-mapping table $\mathcal{M}$ that satisfies the constraint (4.1) is given in Figure 4.1. So, since the target value $CSI4109$ is only mapped to the source element $COMP4001$, and not to any other source element; then it can uniquely identify it. The same property holds for the target value $CSI3130$ which uniquely identifies the source element $COMP3005$.

We store the set of values in $(dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{T}})$ that uniquely identify each source value mapped in $\mathcal{M}$ in a fresh table C (to be used later) that is defined in both schemas $\mathbf{S}$ and $\mathbf{T}$. So, by looking at the st-mapping table $\mathcal{M}$ in Figure 4.1, $C = \{CSI3130, CSI2132, CSI1308, CSI4109, CSI3105\}$.

Intuitively, adopting usual universal DSE solutions as a semantics for a DSE$_U$ setting does not support generating sound certain answers for conjunctive queries. Therefore, to have a functionality that returns a sound certain answer in a DSE$_U$ setting means that we need to define a particular type of target solutions that contain a portion of the set of exchanged data and with particular properties. Therefore, we introduce in this chapter a type of DSE solutions that we call **DSE KB-solutions**, which possess such properties. In addition, we define DSE KB-solutions in a way such that they closely and in a precise way represent universal DSE solutions. As we show later, DSE KB-solutions support the generation of both complete certain answers and sound certain answers of conjunctive queries. The following example underlines the intuition behind such solutions in a DSE$_U$ setting.

Student

Sname	Sage	Saddress
Alex	18	$\perp_1$
Tom	19	$\perp_2$

Enroll

Sname	Cid	Cgrade
Tom	CSI4109	C
Alex	CSI3130	B

Figure 4.2: Universal DSE KB-Solution J

Example 4.1.2 *In reference to Example 3.2.1, assume that $\mathcal{M}$ uniquely identifies each major 'CS' course at UOC listed in $\mathcal{M}$, with the equivalent major 'CS' course(s) at UOO. Also, consider that the remaining set of UOO mapped courses are considered as similar/related to their corresponding source UOC courses.*

In some applications, users would be interested in storing in their target application information about students that applied for a program transfer from UOC to UOO. They also want to store the list of UOO major 'CS' courses that correspond to the list of major courses they completed at UOC, without storing the related ones; they rather should be able to generate those related courses only when needed.

We give in Figure 4.2 a possible solution J for I and $\mathcal{M}$ under $\mathfrak{S}$ with such a property. Intuitively, UOO users are able to infer using J information about the major 'CS' courses that UOC students are considered to have completed at UOO. In other words, UOO users could tell that Tom is considered to have completed the $CSI4109$ UOO course. In addition, we show later in this chapter how to define $\mathfrak{S}$ in a way that would allow UOO users to determine, using J and the st-mapping table $\mathcal{M}$, that Tom is also considered, to have completed $CSI3105$ and $CSI2110$ courses, in addition to $CSI4109$. □

To formally define a DSE$_U$ setting in a way that supports what we explained above, we recast the setting in terms of a knowledge base exchange framework as mentioned earlier in Chapter 2, a knowledge base (KB) over schema $\mathbf{R}$ is a pair (K, Σ), where K is an instance of $\mathbf{R}$ (the explicit data) and Σ is a set of logical sentences over $\mathbf{R}$ (the implicit data) that would

complement K with the set of facts entailed by the semantics of those rules when applied to it. Also, the set of *models* of (K, Σ), denoted by $\mathsf{Mod}(K, \Sigma)$, is defined as the set of instances of $\mathbf{R}$ that contain the explicit data in K and the derived data in Σ.

Knowledge base frameworks have been used to represent various types of data, including ontologies in the semantic web, which are expressed using different types of formalisms such as Description Logics [52]. We give below an example, referenced from the work in [11], that illustrates the idea of a knowledge base.

Example 4.1.3 *Given a schema* $\mathbf{S}$ *that consists of the relational symbols* $P(.,.)$ *and* $GP(.,.)$. *Let* (I, Σ) *be a knowledge base where* I *is an instance of* $\mathbf{S}$. *Let* Σ *be the following FO sentence:*

$$\Sigma \quad = \quad \{P(x, y) \wedge P(y, z) \rightarrow GP(x, z)\}$$

Consider $I = \{P(a, b), P(b, d)\}$ *to be an instance of* $\mathbf{S}$. *In this case, a model of* (I, Σ) *that constitutes the implicit data entailed by* Σ *additional to* I *would be* $I' = \{P(a, b), P(b, d), GP(a, d)\}$
□

Following the intuition behind Σ explained in Example 4.1.3, our goal now is to come up with a set of FO sentences, we call it Σ_t^{c}, such that, if we apply those to a DSE KB-solution J, like the one given in Example 4.1.2, a universal DSE solution which consists of the full set of exchanged data, like the one shown in Figure 3.4, would be generated.

Intuitively, in a DSE_U setting, C is the sole set of target values that captures correctly the set of source values exchanged to a target instance. Therefore, we use C as a fundamental part of the FO sentences in Σ_t^{c}. To generate the target elements in $(dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{T}})$ that should be in C, we define the following set of FO sentences d over the schema that includes $\mathcal{M}$ and C, and that uses a fresh relation $C'(x)$:

$$\forall x \forall y \forall z(\mathcal{M}(x,y) \wedge \mathcal{M}(z,y) \wedge x \neq z \rightarrow C'(y)) \tag{4.2}$$

$$\forall x \forall y(\mathcal{M}(x,y) \wedge \neg C'(y) \rightarrow C(y)) \tag{4.3}$$

As we stated in Chapter 2, authors in [27] addressed the problem of chasing dependencies which contain negated predicates and inequality formulas, and applied simple modifications to these dependencies in order to solve this problem. Since rules 4.2 and 4.3 possess such properties, we apply this solution in order to populate the table C. We show in Theorem 4.1.1 below that the problem pf generating C is in LOGSPACE.

Theorem 4.1.1 *Let $\mathfrak{S}$ be a fixed DSE$_U$ setting and $\mathcal{M}$ be an st-mapping table. Then generating C is in* LOGSPACE.

Based on what we explained so far, we define the semantics of the DSE$_U$ setting to be based on the following two items: (a) a target instance that stores a portion of the set of exchanged data which precisely represents the complete set, (b) and a completion process that uses a set of FO sentences to generate a universal DSE solution with the full set of exchanged data.

Authors in [11] identified a knowledge base exchange setting in which implicit data are FO sentences, in the form of full tgds, where generating target KBs is tractable. Following this result, we define below a completion process with a set of FO sentences, Σ_t^c, as a set of full tgds over a schema that includes $\mathbf{T}$ and $\mathcal{M}$, such that applying Σ_t^c to a universal DSE KB-solution J generates a universal DSE solution J' for I and $\mathcal{M}$ under $\mathfrak{S}$, where J' contains the complete set of exchanged data; that is $((I \cup \{\mathcal{M}\}), J') \vDash \Sigma_{st}$

Definition 4.1.2 (Target Completion Process) *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE$_U$ setting. The target completion of $\mathfrak{S}$, denoted by Σ_t^c, is the following set of* FO *sentences over $\mathbf{T} \cup \{\mathcal{M}, \text{RELATED}\}$, where* RELATED *is a fresh binary table:*

1. *For each $T \in \mathbf{T} \cup \{\mathcal{M}\}$ of arity n and $1 \leq i \leq n$:*

   $\forall x_1 \cdots \forall x_n(T(x_1, \ldots, x_i, \ldots, x_n) \rightarrow \text{RELATED}(x_i, x_i))$.

2. $\forall x \forall y \forall z(\mathcal{M}(z,x) \wedge \mathcal{M}(z,y) \wedge C(x) \rightarrow \text{RELATED}(x,y))$.

3. *For each* $T \in \mathbf{T}$ *of arity* n*:*
$\forall x_1, y_1 \cdots \forall x_n, y_n\,(T(x_1,\ldots,x_n) \wedge \bigwedge_{i=1}^{n} \text{RELATED}(x_i, y_i) \rightarrow T(y_1,\ldots,y_n))$.

The first rule defines the reflexive relation RELATED on the domain of the target instance. The second rule captures the target elements that are related to the same source value in the st-mapping table $\mathcal{M}$. The last rule allows to complete the symbols of $\mathbf{T}$, by adding elements declared to be related in RELATED.

In a DSE$_U$ setting, we define source KBs to be of the form $((I \cup \{\mathcal{M}\}), \emptyset)$, which intuitively correspond to data constituting the source instance I and the st-mapping table $\mathcal{M}$. On the other hand, the target KBs we are interested in, are of the form $((J \cup \{\mathcal{M}\}), \Sigma_t^c)$ – we called those DSE KB-solutions – where J contains a portion of the exchanged data existing in a universal DSE solution, and Σ_t^c contains a set of FO sentences over a schema that includes $\mathbf{T}$ and $\mathcal{M}$.

4.2 DSE KB-Solutions

A good bulk of work has recently tackled the problem of exchange of KBs that are defined using different DL languages [60, 61]. In a DSE$_U$ setting, we formalize the notion of a (universal) DSE KB-solution to extend the (universal) KB solution introduced in [11]. The main difference is that in DSE KB-solutions we need to coordinate the source and target information provided by $\mathcal{M}$, as opposed to KB solutions that require no data coordination at all. This is done by establishing precise relationships in a (universal) DSE KB-solution between the interpretation of $\mathcal{M}$ in $\mathbf{S}$ and $\mathbf{T}$, respectively. KB exchange in DL showed that target (universal) KB solutions [11] present several limitations since these can miss some semantics of the source KB [60, 61]. Universal DSE KB-solutions, on the other hand, do not possess those limitations and they reflect the semantics in the source KB and the st-mapping table accurately.

From now on, $K_{\mathbf{R}'}$ denotes the restriction of instance K to a subset $\mathbf{R}'$ of its schema $\mathbf{R}$. We give below the formal definition of a DSE KB-Solution which defines the semantics of a

DSE_U setting.

Definition 4.2.1 (DSE KB-solution) *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE setting, I a source instance, $\mathcal{M}$ an st-mapping table, and J a target instance. Recall that Σ_t^c is the target completion of $\mathfrak{S}$. Then:*

1. *J is a* DSE KB-solution *for I and $\mathcal{M}$ under $\mathfrak{S}$, if for every $K \in \mathsf{Mod}((J \cup \{\mathcal{M}\}), \Sigma_t^c)$ there is $K' \in \mathsf{Mod}((I \cup \{\mathcal{M}\}), \emptyset)$ such that the following hold: (a) $K'_{\mathcal{M}} \subseteq K_{\mathcal{M}}$, and (b) $K_{\mathbf{T}}$ is a DSE solution for $K'_{\mathbf{S}}$ and $K'_{\mathcal{M}}$ under $\mathfrak{S}$.*

2. *In addition, J is a* universal *DSE KB-solution for I and $\mathcal{M}$ under $\mathfrak{S}$, if J is a DSE KB-solution, and for every $K' \in \mathsf{Mod}((I \cup \{\mathcal{M}\}, \emptyset)$ there is $K \in \mathsf{Mod}((J \cup \{\mathcal{M}\}, \Sigma_t^c)$ such that (a) $K_{\mathcal{M}} \subseteq K'_{\mathcal{M}}$, and (b) $K_{\mathbf{T}}$ is a DSE solution for $K'_{\mathbf{S}}$ and $K'_{\mathcal{M}}$ under $\mathfrak{S}$.*

As shown in [11], the notion of (universal) DSE KB-solution generalizes the notion of (universal) solution as defined in data exchange DE [1] and can be easily reduced to the later when $\Sigma_t^c = \emptyset$.

In reference to the DSE_U setting $\mathfrak{S}$ given in Example 3.2.1, given the source instance I depicted in Figure 3.2 and the st-mapping table given in Figure 3.3, we can see that applying rules 4.2 and 4.3 to $\mathcal{M}$ will generate C = {$CSI4109$, $CSI3105$, $CSI3130$, $CSI2132$, CS, ENG, $Alex$, Tom, 18, B, C}. In a first step, applying rule 4.2 to $\mathcal{M}$ generates the table $C' = \{CSI2110\}$. In a second step, applying rule 4.3 to $\mathcal{M}$ and C' generates C.

Also, we can deduce that the target instance J given in Figure 4.2 is a possible universal DSE KB-solution for I and $\mathcal{M}$ under $\mathfrak{S}$, since applying Σ_t^c to J generates the universal DSE solution depicted in Figure 3.4. The following proposition provides us with a simple yet useful characterization of (universal) DSE KB-solutions in a DSE_U setting which shows that these solutions possess this property.

Proposition 4.2.2 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE_U setting, I a source instance, $\mathcal{M}$ an st-mapping table, J a target instance. Then J is a (universal) DSE KB-solution for I and $\mathcal{M}$ under $\mathfrak{S}$ if and only if $\Sigma_t^c(J)$ is a (universal) DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$.*

Given a DSE$_U$ setting $\mathfrak{S}$, I a source instance, $\mathcal{M}$ an st-mapping table, intuitively, applying Σ_t^c to a universal DSE solution J will not alter J since it contains the complete set of exchanged data and it reflects I and $\mathcal{M}$ accurately. As a consequence, we obtain that (universal) DSE solutions are, in a precise way, (universal) DSE KB-solutions. We summarize this result in the following corollary.

Corollary 4.2.3 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE$_U$ setting, I a source instance and $\mathcal{M}$ an st-mapping table. Further, let J be a (universal) DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$. Then J is a (universal) DSE KB-solution for I and $\mathcal{M}$ under $\mathfrak{S}$.*

Following the fact that applying a procedure (based on the *chase* [26]) to a source instance I and an st-mapping table $\mathcal{M}$ generates a universal DSE solutions in LOGSPACE, and with Corollary 4.2.3, we can directly deduce that there exists an algorithm that generates a universal DSE KB-solution in a DSE$_U$ setting in LOGSPACE.

Corollary 4.2.4 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a fixed DSE$_U$ setting, I a source instance and $\mathcal{M}$ a st-mapping table. There exists a* LOGSPACE *algorithm that generates a (universal) DSE KB-solution for I and $\mathcal{M}$ under $\mathfrak{S}$.*

4.3 Minimal DSE KB-Solutions

In the context of ordinary data exchange, "best" solutions – called cores [10] – are universal solutions with minimal size; that is, they have the minimum number of tuples. In knowledge base exchange, on the other hand, "best" solutions are cores that materialize a minimal amount of explicit data [11]. So, minimal solutions in knowledge base exchange can be more compact than core solutions.

In a DSE$_U$ setting, on the other hand, a *minimal* universal DSE KB-solution (MUDSE) is a core universal DSE solution J that contains a minimal amount of explicit data with respect to Σ_t^c, and such that no universal DSE KB-solution with strictly fewer constants is also a uni-

versal DSE KB-solution with respect to Σ_t^c. We give below the formal definition of a MUDSE solution.

Definition 4.3.1 (Minimal universal DSE solution) *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE$_U$ setting, I be a source instance, $\mathcal{M}$ an st-mapping table, and J a universal DSE KB-solution for I and $\mathcal{M}$ under $\mathfrak{S}$. Then J is a MUDSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$, if:*

1. *There is no proper subset J' of J such that J' is a universal DSE KB-solution for I and $\mathcal{M}$ under $\mathfrak{S}$, and*

2. *There is no universal DSE KB-solution J' such that $dom(J') \cap \mathsf{Const}^{\mathbf{T}}$ is properly contained in $dom(J) \cap \mathsf{Const}^{T}$.*

We illustrate MUDSE solutions in the following example.

Example 4.3.1 *(Example 3.2.1 cont.) In reference to the DSE$_U$ setting $\mathfrak{S}$ given in Example 3.2.1, the target instance J given in Figure 4.2 is a possible MUDSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$. It is clear that J satisfies conditions 1 and 2 in Definition 4.3.1.*

□

Condition (2) in the definition of MUDSE solutions is not part of the original definition of minimal solutions in knowledge base exchange [11]. However, this condition is necessary as we shall explain below.

Example 4.3.2 *(Example 4.3.1 cont.)*

Going back to the DSE$_U$ instance $\mathfrak{S}$ given in Example 3.2.1, assume that the source instance I given in Figure 3.2 was extended with the table $Teach$ shown in Figure 4.3, which specifies the teachers and the list of courses they teach at UOC, and $\mathcal{M}$ is instead the st-mapping table given in Figure 4.4. Also, consider that $\mathbf{T}$ is extended with the same relation $Teach(Tid, Cid)$ in $\mathbf{S}$, and Σ_{st} in $\mathfrak{S}$ is extended with the following st-mapping dependency d:

$$d = Teach(x, y) \rightarrow Teach(x, y)$$

In such a case, one possible universal DSE KB-solution could be an instance J_1 that extends the target instance J given in Figure 4.2 with a target table that contains the facts $\{Teach(Anna, CSI3105), Teach(Jeff, CSI4109)\}$. In such case, J does not satisfy condition (2) in Definition 4.3.1 and provides us with redundant information with respect to I and $\mathcal{M}$.

The reason is, we can conclude using $\mathcal{M}$ that $CSI3105$ and $CSI4109$ both reflect the same $COMP4001$ course in UOC. Therefore, we do not consider J_1 to be a MUDSE solution in a DSE$_U$ setting. On the other hand, an instance J_2 that extends the target instance J given in Figure 4.2 with a target table $Teach$ that contains the facts $\{(Anna, CSI3105), (Jeff, CSI3105)\}$ is considered a MUDSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$.

Teach

Tid	Cid
Anna	COMP 4001
Jeff	COMP 4001

Figure 4.3: $Teach$ Source Instance

M

S	T
COMP 4001	CSI4109
COMP 4001	CSI3105
COMP 4001	CSI2110
COMP 3005	CSI3130
COMP 3005	CSI2132
COMP 3005	CSI2110
CS	CS
ENG	ENG
Tom	Tom
Alex	Alex
Anna	Anna
Jeff	Jeff
18	18
19	19
80	B
70	C

Figure 4.4: An st-mapping table $\mathcal{M}$

□

We provide below a procedure $\texttt{CompMUDSE}_{\texttt{U}}\texttt{sol}_{\mathfrak{S}}$, that given a source instance I and an st-mapping table $\mathcal{M}$ in a fixed DSE$_U$ setting $\mathfrak{S}$, generates a canonical MUDSE solution J for I and $\mathcal{M}$ under $\mathfrak{S}$ in LOGSPACE.

$\texttt{CompMUDSE}_{\texttt{U}}\texttt{sol}_{\mathfrak{S}}$:

Input: A source instance I, an st-mapping table $\mathcal{M}$, and a set Σ_{st} of st-tgds.

Output: A canonical MUDSE solution J for I and $\mathcal{M}$ under $\mathfrak{S}$.

1. Populate, using a procedure (based on the *chase* [27]), the table C with elements from $\mathcal{M}$ using the FO rules 4.2 and 4.3 we gave before.

2. Apply a procedure (based on the *chase* [26]) to the instance $(I \cup \{\mathcal{M}\})$, and generate a target instance J.

3. Compute a set of classes $\{C_1, \ldots, C_m\}$ over $dom(C)$ such that c_1 and c_2 exist in C_i if there exists a constant $a \in \mathsf{Const}^{\mathbf{S}}$ such that $\mathcal{M}(a, c_1)$ and $\mathcal{M}(a, c_2)$ hold.

4. Choose a set of witnesses $\{w_1, \ldots, w_m\}$ such that $w_i \in C_i$, for $1 \leq i \leq m$.

5. Compute from J the instance $J' :=$ $\texttt{replace}(J, w_1, \ldots, w_m)$ by replacing each occurrence of target constant $c \in C_i \cap dom(J)$ $(1 \leq i \leq m)$ with $w_i \in C_i$.

6. Apply a procedure (based on the core [10]) to the target instance J' and generate the target instance J_1 that is the core of J'.

The following theorem states the correctness of procedure $\texttt{CompMUDSE}_{\texttt{U}}\texttt{sol}_{\mathfrak{S}}$.

Theorem 4.3.2 *Let $\mathfrak{S}$ be a fixed DSE$_U$ setting, I a source instance, and $\mathcal{M}$ an st-mapping table. Suppose that J^* is a result of applying* $\texttt{CompMUDSE}_{\texttt{U}}\texttt{sol}_{\mathfrak{S}}$. *Then, J^* is a MUDSE for I and $\mathcal{M}$ under $\mathfrak{S}$.*

Following the result given in DE settings [10] that there exists a procedure, based on the greedy algorithm [10] which generates a core universal solution in LOGSPACE, we provide the following result.

Theorem 4.3.3 *Let $\mathfrak{S}$ be a fixed DSE$_U$ setting, I a source instance, and $\mathcal{M}$ an st-mapping table. There exists a* LOGSPACE *procedure that computes a MUDSE solution J^* for I and $\mathcal{M}$ under $\mathfrak{S}$.*

4.4 Query Answering

In data exchange, one is typically interested in the *certain answer* of a conjunctive query Q, that is, the answers of Q that hold in each possible DE solution [1]. In knowledge base exchange, on the other hand, certain answers are those that hold in every model of a knowledge base [60]. In both settings, computing the complete set of correct answers happen to be the right semantics for conjunctive query answering.

In ordinary data coordination setting, however, since one element in the source can be mapped to one or more target elements in mapping tables, authors in [6] augmented the notion of certain answer from DE and KB exchange settings (which they call complete certain answer) with the notion of sound certain answer, which contains a portion of the set of certain answers rather than the complete set. As we mentioned earlier in Chapter 2 about DC settings, for some queries computing a sound certain answer can be informative enough and less expensive to compute than computing a complete certain answer. Following this intuition, we also adopt the notions sound and complete certain answers as semantics for conjunctive query answering in a DSE_U setting.

Let $\mathfrak{S}$ be a DSE_U setting, I be a source instance, and $\mathcal{M}$ be an st-mapping table. Informally, a complete certain answer for a conjunctive query Q over I and $\mathcal{M}$ under $\mathfrak{S}$, denoted by $\mathsf{complete\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, corresponds to the set of tuples that belong to the evaluation of Q over each DSE solution J for I and $\mathcal{M}$ under $\mathfrak{S}$. A sound certain answer, on the other hand, for Q over I and $\mathcal{M}$ under $\mathfrak{S}$, denoted by $\mathsf{sound\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, is the set of tuples P that belong to the evaluation of Q over a DSE KB-solution J' such that applying a query completion program to P, in the style of the target completion program given in Definition 4.1.2, would generate tuples that exist in $\mathsf{complete\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$.

To be used later, we define below a query completion process that completes the set $\mathsf{sound\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ to generate $\mathsf{complete\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$.

Definition 4.4.1 (Query Completion Process) *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE_U setting. Let $Q(\bar{x}) = (\bar{x})\exists\bar{y}\ \phi(\bar{x}, \bar{y}) \wedge \psi(\bar{y})$ be a conjunctive query over $\mathbf{T}$ where: $\bar{x}$ is a set of distin-*

guished variables, $\phi(\bar{x}, \bar{y})$ is a conjunction of predicate formulas with distinct variables, and $\psi(\bar{y})$ is a conjunction of formulas of the form $y_1 = y_2$ where $y_1, y_2 \in \bar{y}$.

The query completion *process, denoted by Σ_q^c, is the conjunction of the following* FO *sentences over the combined schema $\{Q, \mathcal{M}, \text{RELATED}\}$, where* RELATED *is a fresh binary table:*

1. *For each $T \in \mathbf{T} \cup \{\mathcal{M}\}$ of arity n and $1 \leq i \leq n$:*

   $\forall x_1 \cdots \forall x_n (T(x_1, \ldots, x_i, \ldots, x_n) \rightarrow \text{RELATED}(x_i, x_i))$.

2. $\forall x \forall y \forall z (\mathcal{M}(z, x) \wedge \mathcal{M}(z, y) \wedge C(x) \rightarrow \text{RELATED}(x, y))$.

3. $\forall x_1 \cdots \forall x_n \, (Q(x_1, \ldots, x_n) \wedge \bigwedge_{i=1}^{n} \text{RELATED}(x_i, y_i) \rightarrow Q(y_1, \ldots, y_n))$.

We provide below the formal definitions of sound and complete certain answers in a DSE_U setting.

Definition 4.4.2 (DSE Complete Certain Answer) *let $\mathfrak{S}$ be a DSE_U setting, I a source instance, $\mathcal{M}$ an st-mapping table, and Q a FO conjunctive query over $\mathbf{T}$.*

A complete certain answer of Q, $\mathsf{complete\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, corresponds to the set of tuples that belong to the evaluation of Q over $K_{\mathbf{T}}$, for each DSE solution J for I and $\mathcal{M}$ under $\mathfrak{S}$ and $K \in \mathsf{Mod}((J \cup \{\mathcal{M}\}), \Sigma_t^c)$. In other words:

$$\mathsf{complete\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q) = \bigcap_{K \in \mathsf{Mod}((J \cup \{\mathcal{M}\}), \Sigma_t^c),\ for\ each\ DSE\ KB\text{-}solution\ J} Q(K_{\mathbf{T}})$$

Definition 4.4.3 (DSE Sound Certain Answer) *let $\mathfrak{S}$ be a DSE_U setting, I a source instance, $\mathcal{M}$ an st-mapping table, and Q a FO conjunctive query over $\mathbf{T}$.*

A sound certain answer of Q, $\mathsf{sound\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, corresponds to the set of tuples P that belong to the evaluation of Q over a DSE KB-solution J for I and $\mathcal{M}$ under $\mathfrak{S}$, such that applying Σ_q^c to P, denoted by $\Sigma_q^c(P)$, returns the set $\mathsf{complete\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$.

The DSE_U instance of Example 4.1.1 given at the beginning of this chapter illustrates both sound and complete certain answers concepts in a DSE_U setting. Now, after formally defining these concepts, we discuss below how to compute those using DSE and MUDSE solutions.

In DE, certain answers of unions of conjunctive queries were evaluated by directly posing them over universal solutions [23, 1], and then discarding tuples with null values. In KB exchange, different approaches were applied in description logic KBs for computing certain answers of conjunctive queries. Some of those adopted pure query re-writing techniques [56] and different query optimization methods [63, 64]. Others used a *combined* approach [57, 58] which aim at materializing a portion of the entailed facts in a pre-computation step, then re-write a conjunctive query to compute the certain answers of it.

Let $\mathfrak{S}$ be a DSE_U setting, I a source instance, $\mathcal{M}$ an st-mapping table, J a MUDSE solution, and J_1 a universal DSE solution. Also, let $Q(\bar{x}) = (\bar{x})\exists\bar{y}\ \phi(\bar{x},\bar{y}) \wedge \psi(\bar{y})$ be a conjunctive query over $\mathbf{T}$ where: $\bar{x}$ is a set of distinguished variables, $\phi(\bar{x},\bar{y})$ is a conjunction of predicate formulas with distinct variables, and $\psi(\bar{y})$ is a conjunction of formulas of the form $y_1 = y_2$ where $y_1, y_2 \in \bar{y}$.

To compute $\mathsf{sound\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, we adopt a method similar to the combined approach given in DL KB-exchange settings [57, 58]. It first applies rules 1 and 2 (See Definition 4.2.1) in the target completion process Σ_t^c to populate the table RELATED with elements entailed to be related by $\mathcal{M}$. Then, re-writes $Q(x_1, \ldots, x_n)$ to: $Q'(x_1, \ldots, x_n) = (\bar{x})\exists\bar{y}\exists\bar{w}$ $\phi(\bar{x},\bar{y}) \wedge \psi'(\bar{y})$ where $\psi'(\bar{y})$ is of the form RELATED $(y_1, y') \wedge$ RELATED(y_2, y') for each formula $y_1 = y_2$ in $\psi(\bar{y})$, to generate the set of sound certain answers.

Similar to the case in DE settings, one way to compute $\mathsf{complete\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ is to pose Q over the universal DSE solution J_1, and then discard tuples with nulls. However, we show later in Section 4.5, that there exists a more efficient method, in terms of execution times, to compute those using MUDSE solutions.

Intuitively, the set $\mathsf{complete\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ cannot be simply obtained by posing Q to the MUDSE solution J, since J might be incomplete with respect to the data entailed from Σ_t^c. Therefore, we present below two possible methods for computing those complete

certain answers using J.

The first method is to complete J with the information entailed by the target completion program Σ_t^c as a first step, and this is done by applying Σ_t^c to J (denoted as $\Sigma_t^c(J)$) and generating a complete target instance $\hat{J}$, then apply Q to $\hat{J}$ as a second step and discard tuples with null values.

A second method, on the other hand, leverages the combined approach we used to compute sound-certain$_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$. It first populates the table RELATED and re-writes Q to a query Q' the same way we did to compute sound-certain$_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$. Then it completes the evaluation of Q' on J by returning the answer of $\hat{Q'}(z_1, \ldots, z_n) = Q'(x_1, \ldots, x_n) \wedge \bigwedge_{i=1}^{n} \text{RELATED}(x_i, z_i)$.

We state the correctness of our query re-writing method in the following proposition:

Proposition 4.4.4 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE$_U$ setting, I a source instance, $\mathcal{M}$ an st-mapping table, J a MUDSE solution, and $Q = (\bar{x})\exists\bar{y}\ \phi(\bar{x}, \bar{y}) \wedge \psi(\bar{y})$ a fixed conjunctive query over $\mathbf{T}$. Then,* complete-certain$_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ = $\hat{Q}(J)$ *where* $\hat{Q}(z_1, \ldots, z_n) = (\bar{x})\exists\bar{y}\exists\bar{w}$ $\phi(\bar{x}, \bar{y}) \wedge \text{RELATED}(y_1, w_1) \wedge \text{RELATED}(\ y_2, w_1) \wedge \cdots \wedge \bigwedge_{i=1}^{n} \text{RELATED}(x_i, z_i), w_i \in \bar{w}$.

After we proved the correctness of our query re-writing method to generate complete certain answers, we can easily deduce that the part of that method which generates the set of sound certain answers is correct. We summarize this result in the following corollary.

Corollary 4.4.5 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE$_U$ setting, I a source instance, $\mathcal{M}$ an st-mapping table, J a MUDSE solution, and Q a fixed conjunctive query over $\mathbf{T}$. Then,* sound-certain$_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ = $Q'(J)$ *where* $Q'(z_1, \ldots, z_n) = (\bar{x})\exists\bar{y}\exists\bar{w}\ \phi(\bar{x}, \bar{y}) \wedge$ RELATED $(y_1, w_1) \wedge$ RELATED$(\ y_2, w_1) \wedge \cdots, w_i \in \bar{w}$.

In addition, we prove in the following proposition that it is possible to compute the set of certain answers of a conjunctive query Q in LOGSPACE.

Proposition 4.4.6 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a fixed DSE$_U$ setting and Q a fixed union of conjunctive queries. There exist* LOGSPACE *procedures that compute* sound-certain$_{\mathfrak{S}}((I \cup$

$\{\mathcal{M}\}), Q)$ *and* $\mathsf{complete\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$*, given a source instance* I *and an st-mapping table* $\mathcal{M}$*.*

4.5 Experiments

4.5.1 Experiments Objective

The objective of this section is to provide experimental results which show how universal DSE solutions and minimal universal DSE KB-solutions behave in terms of exchange times and query execution times, should users decide to generate the aforementioned solutions in target applications. To do so, we implemented the DSE_U knowledge exchange semantics to generate universal DSE solutions and MUDSE solutions. We have leveraged the exchange algorithms introduced in the state of the art ++Spicy system [15] to generate those solutions.

Furthermore, we implemented the two approaches for query re-writing introduced in Section 4.4 to compute in our experiments both sound and complete sets of certain answers for conjunctive queries. We compare the performance of universal DSE solutions versus MUDSE solutions by comparing query execution times when generating sound and complete sets of certain answers.

4.5.2 Experimental Setup

Our experiments were conducted on a Lenovo workstation with a Dual-Core Intel(R) 1.80GHz processor running Windows 7, and equipped with 4GB of memory and a 297 GB hard disk. We used the Python(v2.7) programming language to write the code and we stored the source and the target instances in a PostgreSQL database system. Both instances were simulated with two independent schemas in the same database.

The experiments use the DSE scenario we have in our examples through out the thesis about students and teachers performing university transfers from UOC to UOO. The source schema $\mathbf{S}$ and the target schema $\mathbf{T}$ extend those that were used initially in Example 1.2.1. The

source schema $\mathbf{S}$ given in Figure 4.5(a) represents the schema for the University of Carleton and the target schema $\mathbf{T}$ given in Figure 4.5(b) represents the schema for the University of Ottawa. As we mentioned previously in Chapter 1, relation $Student$ (St) stores students' ids, names, and ages (and addresses) information. Relation $Course$ (Cr) stores courses ids and names information, in addition to the program name which provides each course. Relation $Enroll$ ($Take$) stores the set of courses that each student completed with the final (and average midterm) grade(s) that he/she received in these courses. Relation $Teacher$ (Tch) stores teachers' ids and names (and addresses) information. Finally, relation $Teach$ ($Tutor$) stores the teachers ids and the list of courses' ids that each teacher taught.

Student(Sid, Sname, Sage)
Course(Cid, Cname, Pname)
Enroll(Sid, Cid, Cgrade)
Teacher(Tid, Tname)
Teach(Tid,Cid)
(a)Source Schema for the University of Carleton

St(Sid, Sname, Sage, Saddress)
Cr(Cid, Cname, Pname)
Take(Sid, Cid, Cfgrade, Cmgrade)
Tch(Tid, Tname, Taddress)
Tutor(Tid,Cid)
(b) Target Schema for the University of Ottawa

Figure 4.5: Source and Target Database Schemas

The set of mapping st-tgds we used in our DSE scenarios are the following:

$(a)\ Student(x, y, z) \wedge \mathcal{M}(x, x') \wedge \mathcal{M}(y, y') \wedge \mathcal{M}(z, z') \rightarrow \exists w'\ St(x', y', z', w').$

$(b)\ Student(x, y, z) \wedge Enroll(x, w, u) \wedge \mathcal{M}(x, x') \wedge \mathcal{M}(w, w') \wedge \mathcal{M}(u, u') \rightarrow$
$\exists v'\ Take(x', w', u', v').$

$(a)\ Teacher(x, y) \wedge \mathcal{M}(x, x') \wedge \mathcal{M}(y, y') \rightarrow \exists z'\ Tch(x', y', z').$

$(b)\ Teacher(x, y) \wedge Teach(x, z)\ \mathcal{M}(x, x') \wedge \mathcal{M}(z, z') \rightarrow Tutor(x', z').$

4.5.3 Experimental Results

Based on what we discussed previously, universal DSE solutions generated in DSE$_U$ scenarios where st-mapping tables possess one-to-many multiplicity constraints are less compact than

MUDSE solutions. Therefore, intuitively, we can expect that as the number of related elements per source element increases, the run times to generate universal DSE solutions will also increase. We compare in our experiments the execution times to generate both types of solutions with different sizes of st-mapping tables. In addition, we reflect in the following experimental results how such a property influences the efficiency of computing complete and sound certain answers for conjunctive queries.

DSE and MUDSE Solutions Generation Times

We measure in this experiment the different execution times to generate universal DSE solutions versus MUDSE solutions as the number of UOO courses (in target) mapped to each UOC course (in source) increases.

We used in this experiment a source instance I that contained 5000 records and we mapped each UOC course to different ranges of target UOO courses (1-to-2, 2-to-3, 3-to-4, etc.). We can clearly see in Figure 4.6 how the increase in the number of mapped target elements can deteriorate the execution times to generate universal DSE solutions much more than the times to generate MUDSE solutions. In fact, we can see that the execution times to generate MUDSE solutions hardly changed. This is due to two facts: (a) the time required to populate RELATED table using $\mathcal{M}$ hardly increases with such a small increase of the number of mapped target elements and (b) the size of the generated MUDSE solutions doesn't increase with such changes. The reason of the latter fact is that, MUDSE solutions (as specified in Definition 4.3.1) possess one target elements for a each source element exchanged.

Conjunctive Query Computing Times

We have selected a set of 9 queries of different complexities to compare the performance of computing complete and sound certain answers using a universal DSE solution versus a MUDSE solution (following the two methods introduced in Section 4.4). We provide the list of queries in Table 4.1.

We applied the list of input queries to a DSE instance where each UOC course is mapped

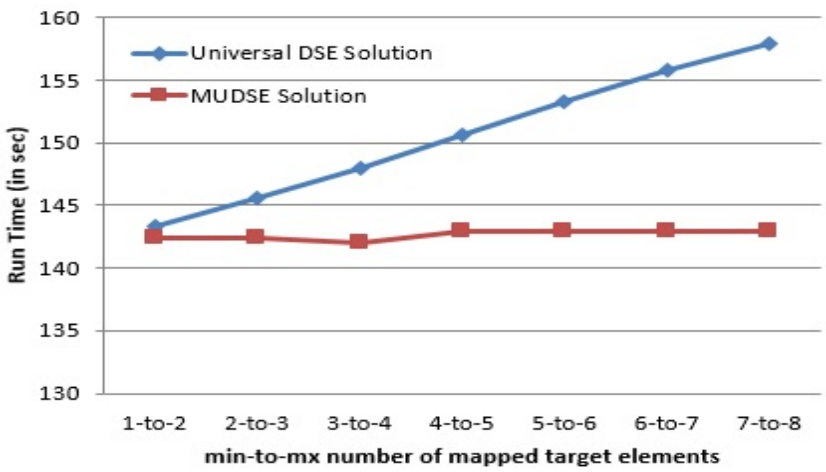

Figure 4.6: MUDSE and Universal DSE solutions Generation Times

to a minimum of 3 and a maximum of 4 UOO courses. In addition, we assumed that UOO adopts two different grading schemes which are the percentage grading scheme (percent over 100), and the letters grading scheme (often A, B, C, etc.) while UOC only uses the first scheme. In other words, each UOC grade is mapped in $\mathcal{M}$ to two different grades in UOO. The remaining elements of the source data, like students' ids, students' names, and teachers' similar information were mapped to itself in $\mathcal{M}$.

We chose a universal DSE solution, that is a core of itself, and that contained around 50,000 records, and a MUDSE solution that contained around 14,800 records. Intuitively, computing sound certain answers consume in worst cases the same execution times as when computing complete certain answers. In fact, as can be seen in Figure 4.7, computing complete certain answers can be fairly more expensive than computing the former in queries that join target elements which possess more than one related elements.

Furthermore, we can see in Figure 4.7 how MUDSE solutions can behave much better than universal DSE solution when computing complete certain answers. The deterioration in performance of query executions against the DSE solution appeared in some queries more than others, like in queries $Q7$, $Q8$, and $Q9$, because of two reasons: (a) these queries apply join operations to the $Take$ and $Tutor$ tables that involve target elements which are related to more than one target element in RELATED, and (b) $Take$ table possess two columns of data that

$Q1$	*Fetch the name and age of each student enrolled in a course*
$Q2$	*Fetch the name and age of each student with the names of courses he completed*
$Q3$	*Fetch the name of each student with the name and grade of each course he finished*
$Q4$	*Fetch the list of teachers that already taught a course*
$Q5$	*Fetch the list of teachers' names and the list of courses they taught*
$Q6$	*Fetch the names of students with the names of courses they completed and the name of the teacher that taught each course*
$Q7$	*Fetch the list of pairs of students' names and ages that took the same course*
$Q8$	*Fetch the list of pairs of teachers' ids that taught common students*
$Q9$	*Fetch the list of pairs of students' ids that shared common grades*

Table 4.1: List of Queries

possess this property.

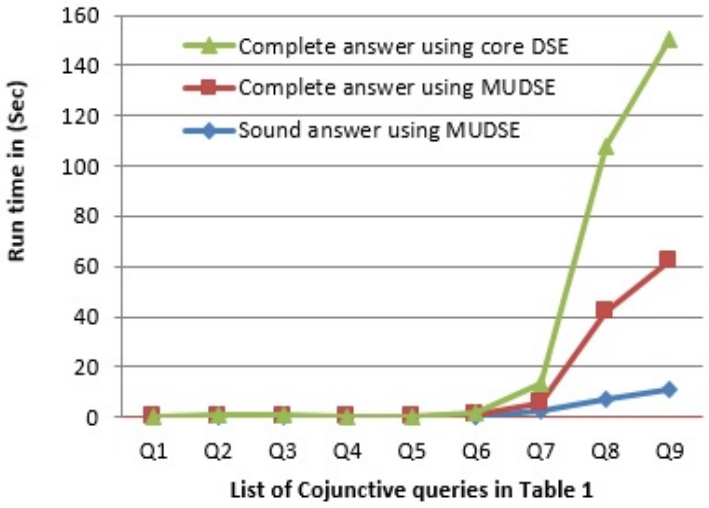

Figure 4.7: Queries of Table 4.1 run times

4.6 Summary

We have defined a DSE setting with a general semantics of related data in an st-mapping table $\mathcal{M}$. This semantics assumes that an element of the universe of the source may be related to several elements of universe of the target. We used the knowledge exchange framework [11] to represent DSE settings with this semantics. We formally defined the semantics of this setting

(called DSE_U setting) and distinguished between two types of solutions: DSE and DSE KB-solutions. We also identified among the later solutions that offer the most compact sets of tuples called minimal solutions, and we provided an algorithm that generates these minimal solutions. Furthermore, we defined the semantics of conjunctive query answering in a DSE_U setting, and we distinguished between complete and sound certain answers. Then, we provided different algorithms to generate both kinds of answers using DSE and MUDSE solutions. Finally, we provided an experimental comparison of the performance of the different algorithms that we introduced through out this chapter.

Chapter 5

DSE with Equality Semantics

The present chapter studies a DSE problem that adheres to a more sophisticated notion of related data than the more general notion discussed in Chapter 4. The semantics of related data in this particular DSE problem is such that a source element is mapped to a target element in an st-mapping table $\mathcal{M}$ only if both their respective values are equivalent. That is, the presence of a pair (a, b), where a is a source element and b is a target element, in an st-mapping table $\mathcal{M}$ means that a and b are considered by the target to be two different names that refer to the same object. We introduce a DSE setting called DSE with Equality semantics, denoted by (DSE_E), to represent this particular sharing and exchange problem.

5.1 Data Sharing and Exchange Setting With Equality Semantics

5.1.1 Equality Semantics

As we shall see later through out this chapter, DSE_E is different than ordinary DE and DC settings because data mappings specified by specialists in an st-mapping table $\mathcal{M}$ along with the equivalence semantics of related data can lead to new elements being considered to be equivalent according to the target, and which do not exist neither in $\mathcal{M}$ nor in the source.

Formally, a DSE$_E$ setting $\mathfrak{S}$, is a setting where Definition 3.2.1 in Section 3 applies to $\mathfrak{S}$. In addition, the st-mapping table $\mathcal{M}$ in $\mathfrak{S}$ is subject to an equivalence relation, denoted by $\sim$. This relation identifies new source or target elements that are *inferred* by the semantics of $\mathcal{M}$ as equivalent objects. We use the notation $a \sim b$ to intuitively denote that elements a and b, where $\{a, b\} \subseteq \mathsf{Const}^{\mathbf{S}}$ (or $\{a, b\} \subseteq \mathsf{Const}^{\mathbf{T}}$) are inferred by the semantics of an st-mapping table $\mathcal{M}$ as equivalent. We formally define the semantics of the equivalence relation $\sim$ as follows:

Definition 5.1.1 (Semantics of $\sim$) *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE$_E$ setting and $\mathcal{M}$ an st-mapping table. The semantics of the equivalence relation $\sim$ in $\mathcal{M}$ is:*

- $\forall x \forall y \forall z(\mathcal{M}(x, y) \wedge \mathcal{M}(x, z) \rightarrow y \sim z)$
- $\forall x \forall y \forall z(\mathcal{M}(x, y) \wedge \mathcal{M}(z, y) \rightarrow x \sim z)$

In addition to that, if there exists a tuple $t(a_1, \ldots, a_i, \ldots, a_n)$ in a source instance I (a target instance J) and $a_i \sim a'_i$, then a tuple $r(a_1, \ldots, a'_i, \ldots, a_n)$ exists in I (in J), where $i \leq n$.

Before we start the discussion on the semantics of a DSE$_E$ setting, we first show in counter examples in what way ordinary data coordination and data exchange settings are quite different than DSE$_E$ settings. We give next in Example 5.1.1 a DSE$_E$ scenario which can not be solved by the ordinary data coordination settings [7, 5].

Example 5.1.1 *Let $\mathfrak{S}$ be a DC setting. Suppose that $\mathbf{S}_1$ in $\mathfrak{S}$ is a schema for the University of Carleton and $\mathbf{S}_2$ in $\mathfrak{S}$ is the schema of the University of Ottawa.*

Let I be the instance of $\mathbf{S}_1$ given in Figure 5.2 and J be an instance of $\mathbf{S}_2$ given in Figure 5.3. Further, assume that $\mathbf{S}_1$ and $\mathbf{S}_2$ are connected by the st-mapping table $\mathcal{M}$ given in Figure 5.1.

Assume in this example that UOO credits a 'CS' course to a student doing a program transfer from UOC only if this student, according to $\mathcal{M}$, finishes an equivalent 'CS' course in UOC. Therefore, according to DC settings [5], when a query q is posed to the instance J which computes the list of students considered by UOO to have finished 'CS' courses, q will

M

S	T
ECOR1606	CSI1390
COMP1005	CSI1390
COMP1005	CSI1790
COMP 4001	CSI4109
CS	CS
ENG	ENG
80	B
70	C
90	A
Alex	Alex
Tom	Tom

Figure 5.1: Mapping Table $\mathcal{M}$

Student

Sname	Sage
Alex	18
Tom	19

Course

Cid	Cname	Pname
ECOR1606	Problem Solving and Computers	ENG
COMP1005	Introduction to Computer Science I	CS
COMP4001	Distributed Computing	CS

Enroll

Sname	Cid	Cgrade
Alex	ECOR1606	80
Tom	COMP 4001	70

Figure 5.2: An instance I of $\mathbf{S}_1$

St

Sname	Sage	Saddress
Ben	18	Ottawa

Cr

Cid	Cname	Pname
CSI1390	Introduction to Computers	CS
CSI1790	Introduction aux Ordinateurs	CS
CSI4109	Introduction to Distributed Computing	CS

Take

Sname	Cid	Cgrade
Ben	CSI1390	C
Ben	CSI4109	A

Figure 5.3: An instance J of $\mathbf{S}_2$

be re-written to a query q' to retrieve a similar list from UOC following the semantics of $\mathcal{M}$. A query q' can be the following:

q': Select $Sname$ From $Course$, $Enroll$ Where $Course.Cid$ = $Enroll.Cid$ And $Course.Pname$ = $'CS'$.

We can see that posing query q' to I returns the result $\{Tom\}$. □

We notice in Example 5.1.1 though that $Alex$ is not considered by UOO as having completed a 'CS' course at UOC. Therefore, if $Alex$ does a transfer to the CS program in UOO, he will not be credited the *Introduction to Computers* course with code $CSI1390$. However, if the semantics of the mapping table $\mathcal{M}$ in this example specifies that course $CSI1390$ in UOO is equivalent to the ENG course $ECOR1606$ in UOC, and course $CSI1390$ in UOO is equivalent to the CS course $COMP1005$ in UOC, then it can be deduced that courses $ECOR1606$ and $COMP1005$ are considered as equivalent with respect to UOO according to $\mathcal{M}$. Therefore, given the fact that $Enroll(Alex,\ ECOR1606,\ 80) \in I$ in Figure 5.2, and according to the semantics in $\mathcal{M}$, $Alex$ is considered to have finished the equivalent 'CS' course $COMP1005$ in UOC and he should be credited the 'CS' course $CSI1390$ with a grade 80 if he is transfered to UOO.

We can deduce from the scenario given in Example 5.1.1 that source and target data can be incomplete with respect to the "implicit" information provided by the semantics of the st-mapping table $\mathcal{M}$. Also, it is clear that query re-writing techniques in data coordination settings [43, 8], and even if these techniques coordinate different vocabularies, cannot solve a DSE$_E$ problem and capture the above specified missing data. Plus, such missing information entailed by the equivalence semantics of $\mathcal{M}$ is of course valuable and cannot be discarded when the mapping table is adopted as a coordination tool between two environments. Therefore, we define a DSE$_E$ setting that captures this missing information using a set of implicit data, in the form of rules, defined over the combined schemas $(\mathbf{S}\cup\{\mathcal{M}\})$ and $(\mathbf{T}\cup\{\mathcal{M}\})$, and stored in the source and the target instances respectively. So, in Example 5.1.1, I is incomplete with respect to the semantics of $\mathcal{M}$ since $Alex$ is considered by UOO according to the $\sim$-relation in $\mathcal{M}$ as have finished the 'CS' course *Introduction to Computer Science I* with $Cid = COMP1005$.

The additional information $ECOR1606 \sim COMP1005$ entailed by the data mappings in $\mathcal{M}$ and the equivalence semantics of $\mathcal{M}$ could not be identified, if we represented the DSE$_E$ problem using the DC setting.

5.1.2 Pre-Solution for DSE$_E$ Setting

As a second thought on this problem, it might appear that DSE$_E$ can be represented by an ordinary DE setting $\mathfrak{S}$, as we explained in Chapter 3, where $(I \cup \{\mathcal{M}\})$ defines the source instance and an instance J such that $((I \cup \{\mathcal{M}\}), J) \vDash \Sigma_{st}$ is considered a target solution for I and $\mathcal{M}$ under $\mathfrak{S}$. However, we show in what follows that, similarly as is the case in DC settings, a DSE$_E$ problem is different than a DE problem because it is not adequate just to use the st-mapping table as a coordination tool between the source and target environments to solve it, but it is also necessary to store additional implicit data, in the form of rules, in both environments to capture the real semantics of this setting. Therefore, in order to avoid confusions from now on we refer to J in the former case as a *pre-solution* for I and $\mathcal{M}$ under $\mathfrak{S}$.

Example 5.1.2 *(Example 5.1.1 cont.). Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE setting. Suppose that $\mathbf{S}$ in $\mathfrak{S}$ is the schema of the University of Carleton and $\mathbf{T}$ in $\mathfrak{S}$ is the schema of the University Ottawa.*

Let $\mathcal{M}$ in $\mathfrak{S}$ be the st-mapping table $\mathcal{M}$ given in Figure 5.1, and I be the source instance given in Figure 5.2. We consider the target instance J to be initially empty before the exchange process takes place. Also, let Σ_{st} consist of the following st-mapping dependencies:

(a) $Student(x, y) \wedge Enroll(x, w, u) \wedge Course(w, v, \text{`}CS\text{'}) \wedge \mathcal{M}(x, x') \wedge \mathcal{M}(y, y')$
$\rightarrow \exists z St(x', y', z).$

(b) $Student(x, y, z) \wedge Enroll(x, w, u) \wedge Course(w, v, \text{`}CS\text{'}) \wedge \mathcal{M}(x, x') \wedge \mathcal{M}(w, w') \wedge \mathcal{M}(u, u')$
$\rightarrow Take(x', w', u').$

It is clear that this DSE$_E$ instance is exchanging information about students that were

enrolled in 'CS' courses at UOC with the list of 'CS' courses they have completed. The instance $J = \{St(Tom, 19, \perp),\ Take(Tom, CSI4109, C)\}$*, where* $\perp$ *is a fresh unknown (or null) value, is a target instance that satisfies* Σ_{st} *when considered with the source instance* I *and the st-mapping table* $\mathcal{M}$*. In the ordinary DE setting,* J *is called a universal solution [1] for* I *and* $\mathcal{M}$ *under* $\mathfrak{S}$*. We, on the other hand, call* J *in a DSE*$_E$ *setting as a universal pre-solution for* I *and* $\mathcal{M}$ *under* $\mathfrak{S}$*.* □

The same discussion we held earlier in which we explained the reason why semantics of DC settings are different from those in a DSE$_E$ setting holds for the case of a DE setting. We can see in Example 5.1.2 that applying the semantics of data exchange in a DSE$_E$ setting will not capture the "implicit" information in $\mathcal{M}$ which specify that $Alex$ should be considered to have completed the 'CS' course *Introduction to Computer Science I* with $Cid = COMP1005$ in UOC and that he should be exempted from taking the equivalent UOO course *Introduction to Computers* course with $Cid = CSI1390$ according to $\mathcal{M}$, should he apply for a transfer to UOO. Therefore, we can easily deduce that in a DSE$_E$ setting $\mathfrak{S}$, we cannot identify solutions with pre-solutions. One reason is that a source instance I in $\mathfrak{S}$ can be incomplete with respect to the semantics of $\mathcal{M}$. A second reason why pre-solutions are not the best semantics in a DSE$_E$ setting, as we will explain later in Section 5.2, is that data mappings in an st-mapping table $\mathcal{M}$ can also be *incomplete* with respect to the semantics of $\mathcal{M}$.

Given the above reasoning, one might again think to solve a DSE$_E$ problem by reducing it to a DE setting $\mathfrak{S}$ with a set of dependencies defined over combined schemas $(\mathbf{S} \cup \{\mathcal{M}\})$ and $(\mathbf{T} \cup \{\mathcal{M}\})$. One might think just to complete the source instance I, the st-mapping table $\mathcal{M}$, and the generated universal pre-solution J for I and $\mathcal{M}$ under $\mathfrak{S}$ with the additional information entailed by the semantics of $\mathcal{M}$, and generate a universal DSE solution to be stored in the target. However, usually the real reason behind defining dependencies over a databases schema is to ensure that data stored in the extension of this schema follows the structure of it [65], and not to represent certain semantics. Also, such constraints are not treated as additional data, *implicit* data, whose purpose is to entail new facts in addition to the stored ones, but used as a check whether each new tuple that needs to be stored follows the

structure of the underlying schema. Consequently, applying source and target dependencies to source and target instances, respectively, in the usual DE process [1] is a fundamental step to generate solutions that reflect the source instance accurately and that are 'good' enough to compute certain answers for conjunctive queries. So, to apply such intuition to Example 5.1.2 and aim at generating a 'good' solution that can provide these certain answers, which we refer to as complete certain answers, it is fundamental that both explicit data stored in the source instance I, depicted in Figure 5.2, and implicit data entailed by the semantics of $\mathcal{M}$, which is the tuple $Enroll(Alex, COMP1005, 80)$, are exchanged and stored in the target to generate a universal DSE solution, which intuitively can generate such answers.

We show in this chapter that the semantics of a DSE$_E$ setting vary from those of a DE setting since there still exists a type of solutions, different than the DSE solutions introduced in Definition 3.2.1, which consist of a portion of the full set of exchanged data, and yet are still good solutions. The latter will be proved in Section 5.5 to be more efficient than universal DSE solutions for conjunctive query answering.

More precisely, we show in the following section how natural it appears to represent a DSE$_E$ setting using the knowledge exchange framework introduced in [11]. We also show how universal DSE KB-solutions, given in Definition 4.2.1, can define a natural semantics for a DSE$_E$ setting and, as was the case in DSE$_U$ settings, can generate a set of sound certain answers that minimally represents in a precise way the complete set of certain answers for a conjunctive query Q. In some situations it can be enough for users to see a portion from each set of equivalent elements in the target instance. The reason is that in certain scenarios, one certain answer for a conjunctive query that is equivalent to the complete set is informative enough for users as it precisely represents the complete set and conveys the same meaning. In addition, this answer can be even computed much more efficiently as we see in Section 5.5. Also, such solutions, augmented with the additional rules stored in the source and the target can regenerate the complete set of certain answers upon request. With such flexibility, these solutions fulfil the second main requirement in a DSE setting which is to allow users to compute whenever possible and on demand a portion and/or the complete set of certain answers for a conjunctive

query. This again follows the intuition of sound and complete query re-writing in ordinary data sharing settings [7].

5.2 DSE KB-Solutions

5.2.1 Source and Target Completion

Before discussing the semantics of a DSE$_E$ setting, we begin this section by referring to the DSE$_E$ instance $\mathfrak{S}$ given in Example 5.1.2 to explain the two ways in which data in $\mathfrak{S}$ is incomplete with respect to the equivalence semantics of $\mathcal{M}$.

First, just as we explained earlier in Section 5.1.1, since $\mathcal{M}(ECOR1606, CSI1390)$ holds in $\mathfrak{S}$, then UOC course $ECOR1606$ is equivalent to the UOO course $CSI1390$. Also, since $\mathcal{M}$ $(COMP1005, CSI1390)$ holds, then UOC course $COMP1005$ is equivalent to the UOO course $CSI1390$. Therefore, we can deduce that $ECOR1606 \sim COMP1005$ with respect to the target UOO. This means, according to semantics of $\sim$, the source instance I is incomplete, since I should include the tuple $Take(Alex, COMP1005, 80)$ in order to be complete with respect to $\mathcal{M}$.

Second, since $\mathcal{M}(COMP1005, CSI1390)$ holds in $\mathfrak{S}$, then the UOC course $COMP1005$ is equivalent to the UOO course $CSI1390$ according to the semantics of $\mathcal{M}$. Also, since $\mathcal{M}(COMP1005, CSI1790)$ holds in $\mathfrak{S}$, then course $COMP1005$ is equivalent to the UOO course $CSI1790$. Therefore, we can deduce that $CSI1390 \sim CSI1790$, according to the semantics of $\mathcal{M}$. This implies that $\mathcal{M}$ is incomplete, since the fact that $\{(ECOR1606, CSI1390), (COMP1005, CSI1390), (COMP1005, CSI1790)\} \subseteq \mathcal{M}$ entails from the semantics of $\sim$ the fact that $(ECOR1606, CSI1790) \in \mathcal{M}$. Therefore, we say I and $\mathcal{M}$ are incomplete in the sense that they do not contain all the data that is implied by the semantics of $\mathcal{M}$.

In fact, it is not hard to see that the completion process we just sketched can become recursive in certain DSE instances. We show one possible recursive scenario for the completion

process in the following example.

Example 5.2.1 *(Example 5.1.2 cont.). Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE$_E$ setting. Assume that the source schema* $\mathbf{S}$ *and the target schema* $\mathbf{T}$ *in Example 5.1.2 are extended with the relational symbol $Teach(Tid, Cid)$ which specifies the teachers identifications and the list of courses they teach. Suppose that the st-mapping table in $\mathfrak{S}$ specifies for each UOC course, the list of equivalent courses in UOO such that, if the source instance I states that a teacher in UOC is able to teach one course, then the $Teach$ table in the target instance should store the list of UOO courses that this teacher, should she move to UOO, is anticipated to be able to easily teach, given her UOC teaching experience.*

Let $\mathcal{M}$ in $\mathfrak{S}$ be the st-mapping table given in Figure 5.4, and I be the source instance given in Figure 5.5. As usual, we consider the target instance J to be initially empty before the exchange process takes place. Also, let Σ_{st} consist of the following st-mapping dependencies: (a) $Teach(x, y) \wedge \mathcal{M}(x, x') \wedge \mathcal{M}(y, y') \rightarrow Teach(x', y')$.

M

S	T
ECOR1606	CSI1390
COMP1005	CSI1390
COMP1005	CSI1308
COMP1004	CSI1308
COMP1004	ITI1120
CS	CS
ENG	ENG
80	B
70	C
90	A
Peter	Peter
Scott	Scott

Figure 5.4: An instance I of $\mathbf{S}_1$

Student

Sname	Sage
Alex	18
Tom	19

Course

Cid	Cname	Pname
COMP1005	Introduction to Computer Science I	CS
COMP1004	Introduction to Computers for the Sciences	CS
ECOR1606	Problem Solving and Computers	ENG

Enroll

Sname	Cid	Cgrade
Alex	COMP1005	80
Tom	ECOR1606	70

Teach

Tid	Cid
Peter	COMP1005
Scott	ECOR1606

Figure 5.5: An instance J of $\mathbf{S}_2$

Given the st-mapping table $\mathcal{M}$ in Figure 5.4, we can deduce in a first step that $ECOR1606 \sim$

$COMP1005$ *since they are both equal to the target constant* $CSI1390$ *according to* $\mathcal{M}$ *and that* $COMP1005 \sim COMP1004$*, since, according to* $\mathcal{M}$*, they are both equal to the target constant* $CSI1308$*. But then, in a second step we can deduce that* $ECOR1606 \sim COMP1004$ *since we know that they are both equivalent to* $COMP1005$*. This reasoning allows us to anticipate that the teaching experience of professor* $Scott$ *at UOC should allow him to teach course* $ITI1120$ *at UOO.*

From what we explained so far, we conclude that the real semantics for a DSE_E setting should be based on the explicit data contained in a source instance I and an st-mapping table $\mathcal{M}$, augmented with the entailed implicit data that can be obtained by following a completion process for the source, the target, and $\mathcal{M}$. We define below a set of FO sentences, of type full tgds [1] , over a schema $(\mathbf{S} \cup \{\mathcal{M}\})$ (or $(\mathbf{T} \cup \{\mathcal{M}\})$) extended with a fresh binary relation symbol EQUAL that captures the semantics of $\sim$ in a recursive scenario. These full tgds formally define this completion process:

Definition 5.2.1 (Source and Target completion) *Let* $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ *be a DSE setting. The* source completion *of* $\mathfrak{S}$*, denoted by* Σ_s^c*, is the conjunction of the following* FO *sentences over the schema* $\mathbf{S} \cup \{\mathcal{M}, \text{EQUAL}\}$*:*

1. *For each* $S \in \mathbf{S} \cup \{\mathcal{M}\}$ *of arity* n *and* $1 \leq i \leq n$*:*

   $\forall x_1 \cdots \forall x_n (S(x_1, \ldots, x_i, \ldots, x_n) \rightarrow \text{EQUAL}(x_i, x_i))$.

2. $\forall x \forall y (\text{EQUAL}(y, x) \rightarrow \text{EQUAL}(x, y))$.

3. $\forall x \forall y \forall z (\text{EQUAL}(x, z) \wedge \text{EQUAL}(z, y) \rightarrow \text{EQUAL}(x, y))$.

4. $\forall x \forall y \forall z (\mathcal{M}(x, z) \wedge \mathcal{M}(y, z) \rightarrow \text{EQUAL}(x, y))$.

5. $\forall x \forall y \forall z \forall w (\mathcal{M}(x, z) \wedge \text{EQUAL}(x, y) \wedge \text{EQUAL}(z, w) \rightarrow \mathcal{M}(y, w))$.

6. *For each* $S \in \mathbf{S}$ *of arity* n*:*

   $\forall x_1, y_1 \cdots \forall x_n, y_n \, (S(x_1, \ldots, x_n) \wedge \bigwedge_{i=1}^{n} \text{EQUAL}(x_i, y_i) \rightarrow S(y_1, \ldots, y_n))$.

[1] Recall that full tgds are tgds that do not use existential quantication.

The target completion *of* $\mathfrak{S}$*, denoted* Σ_t^c*, is defined analogously by simply replacing the role of* $\mathbf{S}$ *by* $\mathbf{T}$ *in* Σ_s^c*, and then adding the rule* 7. $\forall x \forall y \forall z(\mathcal{M}(z,x) \wedge \mathcal{M}(z,y) \rightarrow \text{EQUAL}(x,y))$ *that defines the completion of* $\mathcal{M}$ *over the target.*

Notice that the first 3 rules of Σ_s^c make sure that EQUAL is an equivalence relation on the domain of the source instance. The fourth rule detects which source elements have to be declared equal by the implicit knowledge contained in the st-mapping table. The last two rules allow to complete the interpretation of $\mathcal{M}$ and the symbols of $\mathbf{S}$, by adding elements declared to be equivalent in EQUAL. The intuition for Σ_t^c is analogous.

Summing up, data in a DSE$_E$ scenario always consists of two modules: (1) The explicit data stored in the source instance I and the st-mapping table $\mathcal{M}$, and (2) the implicit data formalized in Σ_s^c and Σ_t^c. But this means that in the process of data exchange we also have to consider the two modules. Thus, in the context of DSE$_E$ we are essentially interested in the problem of knowledge exchange for the source knowledge base formed by I and $\mathcal{M}$, as explicit data, and by Σ_s^c, as implicit data.

5.2.2 DSE KB-Solutions

In DSE$_E$, we consider source DSE KBs of the form $((I \cup \{\mathcal{M}\}), \Sigma_s^c)$, which intuitively correspond to completions of the source instance I with respect to the implicit data in $\mathcal{M}$, and analogously target DSE KB-solutions of the form $((J \cup \{\mathcal{M}\}), \Sigma_t^c)$ where Σ_s^c (Σ_t^c) is a set of FO sentences over a schema that includes $\mathbf{S}$ ($\mathbf{T}$) and $\mathcal{M}$.

We adapt the definition of DSE KB-solutions introduced in Chapter 4 to accommodate the implicit data Σ_s^c in the source DSE KB as follows:

Definition 5.2.2 (DSE KB-solution) *Let* $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ *be a DSE setting,* I *a source instance,* $\mathcal{M}$ *an st-mapping table, and* J *a target instance. Recall that* Σ_s^c, Σ_t^c *are the source and target completions of* $\mathfrak{S}$*, respectively. Then:*

1. J *is a* DSE KB-solution *for* I *and* $\mathcal{M}$ *under* $\mathfrak{S}$*, if for every* $K \in \mathsf{Mod}((J \cup \{\mathcal{M}\}), \Sigma_t^c)$

there is $K' \in \mathsf{Mod}((I \cup \{\mathcal{M}\}), \Sigma_s^c)$ *such that the following hold: (a)* $K'_{\mathcal{M}} \subseteq K_{\mathcal{M}}$*, and (b)* $K_{\mathbf{T}}$ *is a DSE solution for* $K'_{\mathbf{S}}$ *and* $K'_{\mathcal{M}}$ *under* $\mathfrak{S}$*.*

2. *In addition,* J *is a* universal *DSE KB-solution for* I *and* $\mathcal{M}$ *under* $\mathfrak{S}$*, if* J *is a DSE solution, and for every* $K' \in \mathsf{Mod}((I \cup \{\mathcal{M}\}, \Sigma_s^c)$ *there is* $K \in \mathsf{Mod}((J \cup \{\mathcal{M}\}, \Sigma_t^c)$ *such that (a)* $K_{\mathcal{M}} \subseteq K'_{\mathcal{M}}$*, and (b)* $K_{\mathbf{T}}$ *is a DSE solution for* $K'_{\mathbf{S}}$ *and* $K'_{\mathcal{M}}$ *under* $\mathfrak{S}$*.*

As shown in [11] these notions generalize the notions of a solution and a universal solution as defined in data exchange DE [1]. In a DSE$_E$ setting, a universal DSE KB-solution reflects in a precise way the source instance I, the st-mapping table $\mathcal{M}$, and the semantics of $\mathcal{M}$. In Example 5.1.2, J = $\{St(Alex, 18, \perp_1)$, $Take\ (Alex, CSI1390, B)$, $St(Tom, 19, \perp_2)$, $Take(Tom, CSI4109, C)\}$ is a universal DSE KB-solution for I and $\mathcal{M}$ under $\mathfrak{S}$.

Same as in Proposition 4.2.2, we prove below in Proposition 5.2.3 a characterization of (universal) DSE KB-solutions in a DSE$_E$ setting which concludes that (universal) DSE solutions are, in a precise way, (universal) DSE KB-solutions.

Proposition 5.2.3 *Let* $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ *be a DSE$_E$ setting,* I *a source instance,* $\mathcal{M}$ *an st-mapping table,* J *a target instance. Then* J *is a (universal) DSE KB-solution for* I *and* $\mathcal{M}$ *under* $\mathfrak{S}$ *if and only if* $\Sigma_t^c(J)$ *is a (universal) DSE solution for* I *and* $\mathcal{M}$ *under* $\mathfrak{S}$*.*

Corollary 5.2.4 *Let* $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ *be a DSE setting,* I *a source instance and* $\mathcal{M}$ *a st-mapping table. Further, let* J *be a* (*universal*) *DSE solution for* I *and* $\mathcal{M}$ *under* $\mathfrak{S}$*. Then* J *is a* (*universal*) *DSE KB-solution for* I *and* $\mathcal{M}$ *under* $\mathfrak{S}$*.*

We define below a simple procedure $\texttt{CompUnivDSEKbSol}_{\mathfrak{S}}$ that, given a DSE$_E$ setting $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$, a source instance I and an st-mapping table $\mathcal{M}$, it generates a universal DSE KB-solution J for I and $\mathcal{M}$ under $\mathfrak{S}$.

$\texttt{CompUnivDSEKbSol}_{\mathfrak{S}}$:

Input: A source instance I, an st-mapping table $\mathcal{M}$, and a set Σ_{st} of st-tgds.

Output: A Canonical Universal DSE KB-solution J for I and $\mathcal{M}$ under $\mathfrak{S}$.

1. Apply the source completion process, Σ_s^c, to I and $\mathcal{M}$, and generate $\hat{I}$ and $\hat{\mathcal{M}}$ respectively.

2. Apply a procedure (based on the *chase* [26]) to the instance $(\hat{I} \cup \{\hat{\mathcal{M}}\})$, and generate a canonical universal pre-solution J for $\hat{I}$ and $\hat{\mathcal{M}}$.

The procedure $\texttt{CompUnivDSEKbSol}_{\mathfrak{S}}$ works as follows: step 1 applies the source completion process Σ_s^c, given in Definition 5.2.1, to I and $\mathcal{M}$, and returns as outcome the source instance $\hat{I}$ and the st-mapping table $\hat{\mathcal{M}}$ that are complete with respect to the implicit data in $\mathcal{M}$. Next, step 2 generates a canonical universal pre-solution J for $\hat{I}$ and $\hat{\mathcal{M}}$ such that $((\hat{I} \cup \{\hat{\mathcal{M}}\}), J) \models \Sigma_{st}$. By mimicking the DE terminology, we call solutions computed using this procedure as *canonical* universal DSE KB-solutions.

We can combine the fact that universal solutions in fixed data exchange settings $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ can be computed in LOGSPACE with some deep results in the computation of symmetrical binary relations [66], to show that universal DSE KB-solutions can be computed in LOGSPACE:

Proposition 5.2.5 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a fixed DSE$_E$ setting. Then computing a universal DSE KB-solution J for a source instance I and an st-mapping table $\mathcal{M}$ is in* LOGSPACE.

5.3 Minimal DSE KB-Solutions

The most compact DSE KB-solutions in a DSE$_E$ setting can be identified by the class of *minimal* universal DSE KB-solutions given in Definition 4.3.1. We illustrate MUDSE solutions in a DSE$_E$ setting in the following example.

Example 5.3.1 *(Example 5.1.2 cont.) The DSE solution J = $\{Student(Tom, 19, \perp_1)$, $Take$ $(Tom, CSI4109, C)$, $Student$ $(Alex, 18, \perp_2)$, $Take$ $(Alex, CSI1390, B)$, $Take$ $(Alex, CSI1790, B)\}$ of Example 5.1.2 is a core universal DSE solution, however it is not the most compact one since it contains the two tuples $\{Take$ $(Alex, CSI1390, B)$, $Take$ $(Alex, CSI1790, B)\}$ which convey the same meaning.*

A possible MUDSE target instance for I and $\mathcal{M}$ under $\mathfrak{S}$ in this DSE$_E$ scenario is J' = $\{Student(Tom, 19, \perp_1)$, $Take(Tom, CSI4109, C)$, $Student\ (Alex, 18, \perp_2)$, $Take\ (Alex, CSI1390, B)\}$, *as it satisfies conditions 1 and 2 in Definition 4.3.1.* □

Condition (2) in the definition of MUDSE solutions is necessary for the same reasons we explained in Chapter 4.

We define below a procedure $\texttt{CompMUDSE}_{\texttt{E}}\texttt{sol}_{\mathfrak{S}}$, that given a DSE$_E$ setting $\mathfrak{S}$, a source instance I, and an st-mapping table $\mathcal{M}$, computes a MUDSE solution J^* for I and $\mathcal{M}$ under $\mathfrak{S}$. This procedure works as follows:

$\texttt{CompMUDSE}_{\texttt{E}}\texttt{sol}_{\mathfrak{S}}$:
Input: A source instance I, an st-mapping table $\mathcal{M}$, and a set Σ_{st} of st-tgds.
Output: A Minimal Universal DSE solution J^* for I and $\mathcal{M}$ under $\mathfrak{S}$.

1. Apply the source completion process, Σ_s^c, to I and $\mathcal{M}$, and generate $\hat{I}$ and $\hat{\mathcal{M}}$ respectively.

2. Define an equivalence relation $\sim$ on $dom(\hat{\mathcal{M}}) \cap \mathsf{Const}^{\mathbf{T}}$ as follows: $c_1 \sim c_2$ iff there exists a source constant a such that $\hat{\mathcal{M}}(a, c_1)$ and $\hat{\mathcal{M}}(a, c_2)$ hold.

3. Compute equivalence classes $\{C_1, \ldots, C_m\}$ for $\sim$ over $dom(\hat{\mathcal{M}}) \cap \mathsf{Const}^{\mathbf{T}}$ such that c_1 and c_2 exist in C_i only if $c_1 \sim c_2$.

4. Choose a set of witnesses $\{w_1, \ldots, w_m\}$ where $w_i \in C_i$, for each $1 \leq i \leq m$.

5. Compute from $\hat{\mathcal{M}}$ the instance $\mathcal{M}_1$:= $\texttt{replace}(\hat{\mathcal{M}}, w_1, \ldots, w_m)$ by replacing each target constant $c \in C_i \cap dom(\hat{\mathcal{M}})$ $(1 \leq i \leq m)$ with $w_i \in C_i$.

6. Apply a procedure (based on the *chase* [26]) to the instance $(\hat{I} \cup \{\mathcal{M}_1\})$, and generate a canonical universal pre-solution J for $\hat{I}$ and $\mathcal{M}_1$.

7. Apply a procedure (based on the core [10]) to the target instance J and generate the target instance J^* that is the core of J.

The following theorem states the correctness of $\texttt{CompMUDSE}_{\texttt{E}}\texttt{sol}_{\mathfrak{S}}$.

Theorem 5.3.1 *Let $\mathfrak{S}$ be a DSE$_E$ setting, I a source instance, and $\mathcal{M}$ an st-mapping table. Suppose that J^* is an arbitrary result for $\texttt{CompMUDSE}_{\texttt{E}}\texttt{sol}_{\mathfrak{S}}(I, \mathcal{M})$. Then, J^* is a minimal universal DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$.*

In data exchange, the smallest universal solutions are known as *cores* and can be computed in LOGSPACE [10]. MUDSE solutions can also be computed in LOGSPACE.

Theorem 5.3.2 *Let $\mathfrak{S}$ be a fixed DSE$_E$ setting. There exists a* LOGSPACE *procedure that computes, for a source instance I and an st-mapping table $\mathcal{M}$, a MUDSE solution J for I and $\mathcal{M}$ under $\mathfrak{S}$.*

Also, we show in this context that MUDSE solutions are *unique*; i.e. they are isomorphic up to renaming of nulls and replacement of each constant c with another constant c' such that $c \sim c'$. We summarize this result in the proposition below.

Proposition 5.3.3 *Let $\mathfrak{S}$ be a fixed DSE$_E$ setting, I a source instance, and $\mathcal{M}$ an st-mapping table. For any two MUDSE solutions J_1 and J_2 for I and $\mathcal{M}$ under $\mathfrak{S}$,it is the case that J_1 and J_2 are isomorphic.*

5.4 Query Answering

We adopt in a DSE$_E$ setting the notions of sound certain answers and complete certain answers introduced for DSE$_U$ settings, in Chapter 4 as semantics for conjunctive query answering. Also, Definitions 4.4.2 and 4.4.3 are suitable to define complete and sound certain answers, respectively, in a DSE$_E$ setting. The only difference is that the query completion program in a DSE$_E$ setting is in the style of the target completion program given in Definition 5.2.1. We give below an example that illustrates a complete and a sound certain answer in a DSE$_E$ setting.

Example 5.4.1 *Let us refer to the DSE setting given in Example 5.1.2. Let* $Q(x, y, z)$ = $Enroll(x, y, z)$. *Then, a complete certain answer for* Q *would be:* $\mathsf{complete\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ = $\{Take(Alex, CSI1390, B), Take(Alex, CSI1790, B), Take(Tom, CSI4109, C)\}$. *A sound certain answer on the other hand would be:* $\mathsf{sound\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ = $\{Take(Alex, CSI1390, B), Take(Tom, CSI4109, C)\}$.

□

We have learned in Chapter 4 that to compute sound certain answers of a conjunctive query Q, $\mathsf{sound\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, it was necessary to re-write Q to a query Q' before applying it to a MUDSE solution in a DSE$_U$ setting. In a DSE$_E$ setting, on the other hand, MUDSE solutions possess more interesting properties where we can compute a sound certain answer for a conjunctive query Q by directly posing Q to a MUDSE solution, and thus no pre-rewriting step for Q is necessary.

Let $\mathfrak{S}$ be a DSE$_E$ setting, I be a source instance, and $\mathcal{M}$ be an st-mapping table. Also, let J be a MUDSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$. As in the case of DSE$_U$ settings, to compute the set $\mathsf{complete\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ of complete certain answers of Q we cannot simply obtain those by posing Q to a MUDSE J, since J might be incomplete with respect to the implicit data in Σ_t^c.

Therefore, we present below two possible methods for computing certain answers using DSE KB-solutions:

The first method is to complete a universal DSE KB-solution J with the implicit information entailed by $\mathcal{M}$ as a first step. This is done by applying the target completion program Σ_t^c to J (denoted as $\Sigma_t^c(J)$) to generate a complete target instance $\hat{J}$, then by applying Q to $\hat{J}$ as a second step and discarding tuples with null values.

A second method, on the other hand, is to compute certain answers of Q using MUDSE solutions. As we mentioned before, a MUDSE solution J in a DSE$_E$ setting possesses an interesting property, that is, applying Q to J returns a set of certain answers U that minimally represents the set of certain answers U' returned when Q is applied to $\hat{J} = \Sigma_t^c(J)$.

To compute the complete set complete-certain$_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ of certain answers for Q directly using a MUDSE solution J, we first apply the set of rules in Σ_t^c, excluding rule 6, to populate the binary table EQUAL. Then we complete the evaluation $Q(x_1, \ldots, x_n)$ of Q on J, and return $\hat{Q}(y_1, \ldots, y_n) = Q(x_1, \ldots, x_n) \wedge \bigwedge_{i=1}^{n} \text{EQUAL}(x_i, y_i)$ while discarding tuples with null values. Intuitively, EQUAL can be computed once and used for every computation of certain$_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ using J. We prove the correctness of the second method in the following proposition.

Proposition 5.4.1 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a fixed DSE$_E$ setting, I a source instance, $\mathcal{M}$ an st-mapping table, J a MUDSE solution, and Q a fixed CQ over $\mathbf{T}$. Then,* certain$_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ = $\hat{Q}(J)$ *where* $\hat{Q}(y_1, \ldots, y_n) = Q(x_1, \ldots, x_n) \wedge \bigwedge_{i=1}^{n} \text{EQUAL}(x_i, y_i)$

In addition, we prove in the following proposition that it is possible to compute the set of certain answers of a conjunctive query Q in LOGSPACE.

Proposition 5.4.2 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a fixed DSE$_E$ setting and Q a fixed union of conjunctive queries. There is a* LOGSPACE *procedure that computes* sound-certain$_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ *and* complete-certain$_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$*, given a source instance I and an st-mapping table $\mathcal{M}$.*

5.5 Experiments

Similar to the objectives identified in Section 4.5.3, we would like in this section to exhibit the difference in performance when storing universal DSE solutions versus minimal universal DSE KB-solutions in the target application of a DSE$_E$ instance. Also, as the case in the DSE$_U$ setting, we have implemented the algorithms we introduced in this Chapter to generate both universal DSE and MUDSE solutions, and those to compute sound and complete certain answers of conjunctive queries. We also leveraged in our implementation the exchange algorithm introduced in the state of the art ++Spicy system [15].

5.5.1 Experimental Setting

We have used the same experimental settings that we have setup for the DSE_U setting sets of experiments. We also used the same set of mapping st-tgds that we used in the DSE_U scenarios.

We used in this experiment a source instance I that contains 5000 tuples, where 500 tuples of those were courses information. Each UOC course is mapped to a maximum of 5 UOO courses and the remaining source elements like students' ids, students' names, students' grades, teachers' similar information were mapped to themselves in $\mathcal{M}$.

5.5.2 Experimental Results

We show in this section how raising levels of recursion in the $\sim$-equivalence relation in the st-mapping table $\mathcal{M}$ can be a direct cause for a rapid increase in size of universal DSE solutions, and the execution times for generating those. To enforce higher levels of recursion in the $\sim$-equivalence relation over the st-mapping table $\mathcal{M}$, we mapped larger numbers of UOC courses in the source to common target UOO courses in $\mathcal{M}$. We synthesized these levels of recursions in our experiments. In addition, we show how such an increase can lead to a noticeable deterioration in the performance when computing complete certain answers for conjunctive queries. Now, since MUDSE solutions do not contain two elements identified as equal by the semantics of the st-mapping table, as specified in Definition 4.3.1, then as we shall see later, higher levels of recursion will not highly effect the performance when computing sound certain answers.

DSE and MUDSE Solutions Generation Times

Figure 5.6 shows that as the percentage of recursion in $\sim$-equivalence relation over $\mathcal{M}$ increases, the run times to generate universal DSE and MUDSE solutions increase. We notice also in this figure that as the percentage of recursion in $\sim$-equivalence relation increases, the run times to generate universal DSE solutions increases faster than when generating MUDSE solutions. The reason behind that is, as the $\sim$ percentage increases, the number of source val-

ues (and target values) inferred to be $\sim$ increases, and thus the size of EQUAL created in Σ_s^c and Σ_t^c increases. Also, since target instances are usually larger than $\mathcal{M}$, the run time of completing the former to generate DSE solutions exceeds the time of completing the later when generating MUDSE solutions.

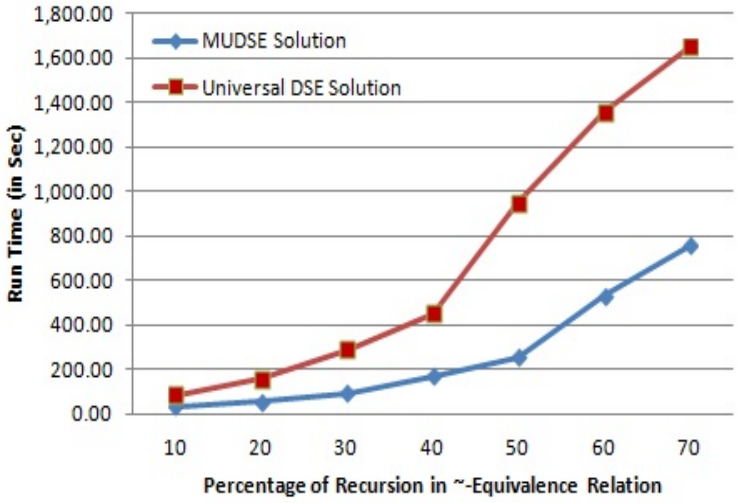

Figure 5.6: MUDSE and Universal DSE solutions Generation Times

DSE and MUDSE Solutions Change in Sizes

The consequences of the increase in percentages of recursion in the $\sim$-equivalence relation over $\mathcal{M}$ that we explained above using Figure 5.6, happen to have the same effect on the sizes of universal DSE solutions and st-mapping tables. The reason is the more recursion in the $\sim$-equivalence relation over $\mathcal{M}$ we have, the more UOO courses that would be equivalent to a UOC course, and the more UOO courses each student is considered to have completed if he/she performs a transfer to UOO. For this reason and intuitively, we can see in Figure 5.7 that as the percentages of recursion in the $\sim$-equivalence relation over $\mathcal{M}$ increases, the sizes of universal DSE solutions and the st-mapping tables increase. On the other hand, we can see that in the same circumstances the sizes of MUDSE solutions tend to decrease. The reason for that is as the the percentages of recursion in the $\sim$-equivalence relation over $\mathcal{M}$ increases, more elements are considered as equivalent in the target.

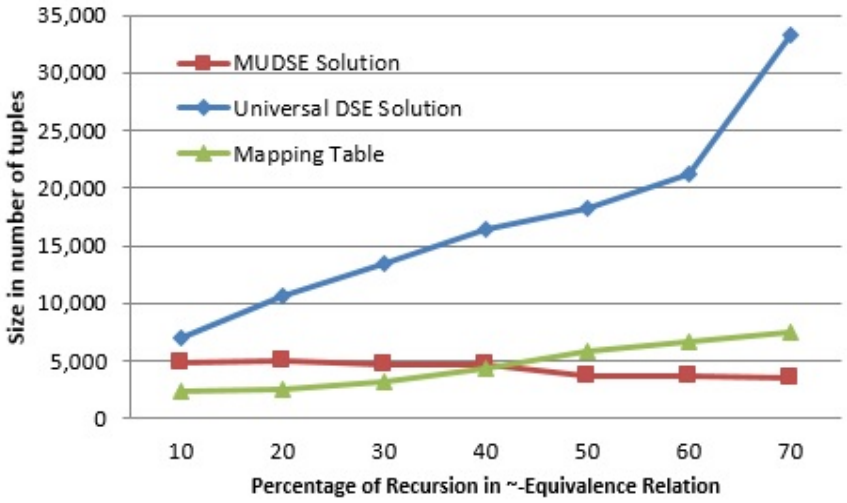

Figure 5.7: MUDSE and Universal DSE solutions sizes increase as recursion in $\sim$-Equivalence increases

Conjunctive Queries Computing Times

We have selected a set of 9 queries to compare the performance of computing complete certain answers using a universal DSE solution (following the first method introduced in Section 5.4) versus a MUDSE solution (following the second method introduced in Section 5.4). We provide the list of queries in Table 5.5.2.

We applied the list of input queries to a DSE$_E$ instance where the $\sim$ percentage is 40% and a course in the source is mapped to a maximum of two courses in the target. We chose a universal DSE solution, with a property of being a core of itself, that had around 18,000 records, and a MUDSE solution that contained around 4,900 records. Figure 5.8 shows that computing the sets of complete certain answers for the input conjunctive queries using a MUDSE solution take less run times than when computing these using a DSE solution. In addition, the deterioration in performance of query execution against the DSE solution appeared more in queries $Q2$ and $Q5$ than the remaining queries, is because both queries apply join operations to the $Take$ and $Tutor$ tables that involves a lot of elements which are inferred to be $equivalent$ by $\mathcal{M}$.

Table 5.1: List of Queries

$Q1$	*Fetch all the students names and the name of courses they have taken*
$Q2$	*Fetch the list of pairs of students ids and names that took the same course*
$Q3$	*Fetch all the students names and the grades they have received*
$Q4$	*Fetch the list of pairs of courses names that belong to the same program*
$Q5$	*Fetch for each student id the pair of courses that he has finished with the same grade*
$Q6$	*Fetch all the courses ids and their names*
$Q7$	*Fetch all the students ids and their names*
$Q8$	*Fetch the list of students names, the list of courses names that they completed, and the teachers' names that provided these courses*
$Q9$	*Fetch the list of pairs of students ids that possess the same address*

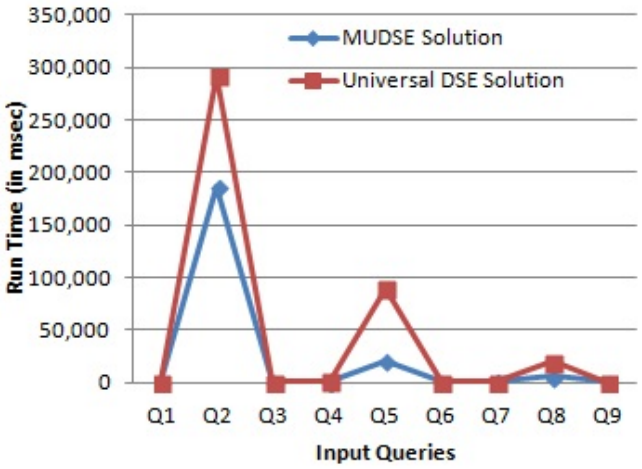

Figure 5.8: Queries run times against a Core of a universal DSE solution and a MUDSE solution

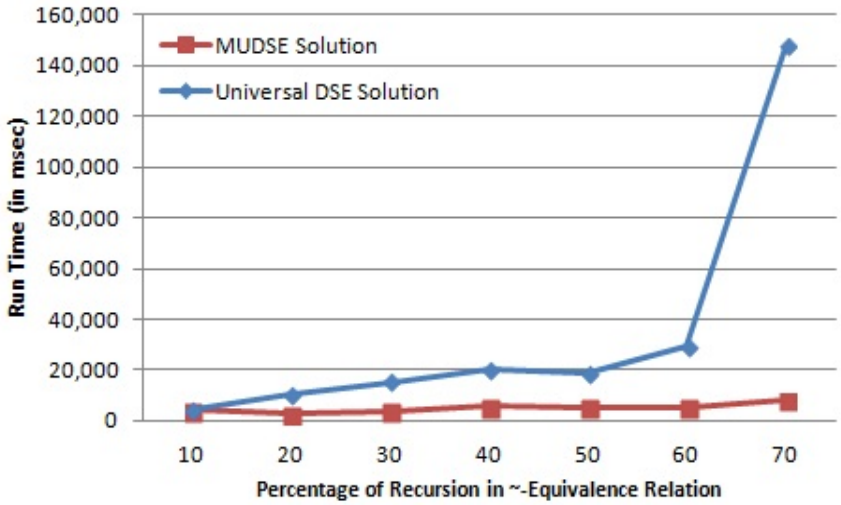

Figure 5.9: Q8 increase in run time as recursion in $\sim$-Equivalence increases

Query Computing Times at Different Levels of Recursion

We compare in this experiment the execution times to compute the set of sound certain answers for query $Q8$ from Table 5.5.2 versus the time to compute the set of complete certain answers as the percentage of recursion in the $\sim$-equivalence relation increases.

Similar to the argument we gave above about Figure 5.7 and for the same reasons, we can see in Figure 5.9 that the run times to generate the set of sound certain answers for $Q8$ using a MUDSE solution do not change with the increase in the percentage of recursion in $\sim$-equivalence relation. However, an under similar circumstances, the run time to compute the set of complete answers do increase.

5.6 Summary

This chapter has introduced a Data Sharing and Exchange setting with peculiar semantics of related data where a source element and a related target element in an st-mapping table are considered as equivalent or refer to the same object. To formalize the semantics of this setting, we defined DSE$_E$ as a knowledge exchange system with a set of explicit source and target data augmented with implicit data, in the form of rules, that are used to infer the implicit information

which should be exchanged and stored in the target. As the case in Chapter 4, we use both DSE solutions and DSE KB-solutions for the DSE_E setting, and identified MUDSE solution as the best solutions among those in terms of compactness. In addition, we defined sound and complete certain answers for conjunctive queries in a DSE_E setting and provided algorithms which compute those using both types of solutions. Finally, we provided experimental results which reflect how much it is effective to generate MUDSE solutions versus universal DSE solutions. We also explored the performance when computing a portion and/or the complete set of certain answers using both types of solutions.

Chapter 6

DSE with Incomplete Source Data

6.1 Introduction

So far we have addressed DSE settings with complete source applications that do not contain unknown data in the form of nulls. However, some scenarios of data exchange can occur in more collaborative settings where several peers interact. In such cases, it is more likely that source peers contain information that are exchanged from other different peers, and hence may contain null values.

The most intuitive instances with null values are instances that contain naïve tables. As we mentioned in Chapter 2, authors in [11] proved that target instances with positive conditional tables are the best tool to represent source instances in such data exchange scenarios. Also, the fact that a target of one data exchange instance can be the source of another, makes it possible for some sources to contain pc-tables. However, pc-tables proved as well in [11] to be good enough to represent source applications with pc-tables.

In this context, we address in this chapter the DSE problem in settings with incomplete information in source instances. We denote those DSE settings by DSE_I. More precisely, we are interested in DSE_I settings which are instances of knowledge exchange settings that exchange source knowledge bases containing explicit data with incomplete information along with a set of rules Σ_s^c and Σ_t^c with full tgds. For easier discussion in this chapter, we adopt the

semantics of mapping tables identified in DSE$_E$ settings in Chapter 5. Clearly, Σ_s^c and Σ_t^c can be easily adapted to a DSE$_I$ setting.

6.2 DSE-KB solutions

After what we discussed so far in DSE$_U$ and DSE$_E$ settings, and given the results that show that generating DSE-KB solutions in both settings is tractable, the first question that comes into place is whether this problem is still tractable in a rich setting such as DSE$_I$. To answer this question, we refer to the following two results proved by authors in [11].

Theorem 6.2.1 *[11] Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a fixed data exchange setting where Σ_{st} is a set of st-tgds. There is a polynomial time algorithm that, given a pc-table I over $\mathbf{S}$, computes a pc-table J over $\mathbf{T}$ that is a universal solution for I under $\mathfrak{S}$.*

Theorem 6.2.1 shows that besides the fact that pc-tables form a robust class for data exchange purposes, it also retains the tractability property when generating "good" solutions to materialize in the target. The second result proved in [11], with respect to knowledge bases, which we can leverage is the following:

Theorem 6.2.2 *[11] Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a fixed knowledge exchange setting where I is a complete instance of $\mathbf{S}$ and Σ_{st} is a set of st-tgds. Let Σ_I and Σ_J be a fixed source and target implicit knowledges respectively that are of type full-tgds. There exists a polynomial time algorithm that generates a target knowledge base solution (J, Σ_J) for a source knowledge base (I, Σ_I)*

Following the results in Theorem 6.2.1 and Theorem 6.2.2, we can easily deduce that there exists a tractable algorithm which generates a universal DSE KB-solution in a fixed DSE$_I$ setting. In fact applying procedure `CompUnivDSEKbSol`$_{\mathfrak{S}}$, introduced in Chapter 5, while using the chase algorithm given in [11], and mentioned in Section 2.4, generates a universal DSE-KB solution in a DSE$_I$ setting. Also, null values in the source do not interfere with the fresh null

values generated by Σ_{st} during the exchange, and thus incompleteness does not cause local problems in the source. We summarize this in the following result.

Theorem 6.2.3 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a fixed DSE$_I$ setting, I be a source instance of type naïve tables or pc-tables, $\mathcal{M}$ be an st-mapping table, and Σ_{st} be a set of st-tgds. Let Σ_s^{c} and Σ_t^{c} be fixed source and target implicit knowledges respectively that are of type full-tgds. Then, there exists a polynomial time algorithm that generates a target DSE KB-solution J for I and $\mathcal{M}$ under $\mathfrak{S}$.*

6.3 Minimal DSE KB-solutions

Most general DSE KB-solutions that we encountered in both DSE$_U$ and DSE$_E$ settings may contain naïve tables. In addition, we have identified in both DSE scenarios, in Chapters 4 and 5 respectively, MUDSE solutions as the most compact solutions in terms of numbers of tuples. In other words, given a DSE setting $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$, a source instance I, an st-mapping table $\mathcal{M}$, and a DSE KB-solution J for I and $\mathcal{M}$ under $\mathfrak{S}$, then a MUDSE solution J' under $\mathfrak{S}$ is the smallest subset DSE KB-solution of J which contain the smallest domain of constants such that $J' \equiv J$ [1].

As a first intuition, one would think that, like in DSE$_U$ and DSE$_E$ settings, given a target instance J with pc-tables in a DSE$_I$ setting $\mathfrak{S}$, the most compact target instance equivalent to J in a DSE$_I$ setting is a pc-table target instance $J' \subseteq J$. However, we show in what follow that there exist exchange instances where MUDSE solutions are not the most compact class of DSE KB-solutions under $\mathfrak{S}$.

Suppose a DSE KB-solution J with naïve tables in a DSE setting M. Since source and target instances can contain pc-tables in DSE$_I$ settings, the question that arises is whether there exists a DSE KB-solution J'' under $\mathfrak{S}$ such that J'' contains pc-tables, $J'' \equiv J'$, where J' is a MUDSE solution under $\mathfrak{S}$, and $|J''| < |J'|$.

[1] Recall that $J' \equiv J$ means that $\mathsf{Rep}(J') = \mathsf{Rep}(J)$.

We prove that, given a DSE setting $\mathfrak{S}$ where DSE KB-solutions under $\mathfrak{S}$ contain naïve tables, MUDSE solutions, introduced in Definition 4.2.1, are the most compact DSE KB-solutions under $\mathcal{M}$ and the best structure to represent MUDSE solutions are indeed naïve tables. In other words, we show in the following theorem that there does not exist pc-tables instances that are more compact, in terms of the number of tuples, than MUDSE naïve instances.

Theorem 6.3.1 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE$_I$ setting, I a source instance, $\mathcal{M}$ an st-mapping table, and J be a MUDSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$ such that J contains naïve tables. Given a target instance J' such that J' is the* core*(J). Then there does not exist a solution J'' under $\mathfrak{S}$ such that J'' contains pc-tables, $\mid J'' \mid < \mid J' \mid$ and $Rep(J'') = Rep(J)$.*

Now we ask the same question about whether Theorem 6.3.1 applies to DSE$_I$ instances where DSE KB-solutions contain pc-tables. We give below a counter example which proves that it is not the case. We show in this example that there exists a DSE KB-solution J with a pc-table R and DSE KB-solution J' with a pc-table R' such that $J \equiv J'$, $|J'| < |J|$, and $J' \not\subseteq J$.

Example 6.3.1 *Consider a DSE$_I$ setting $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ where the target schema $\mathbf{T}$ consists of the positive conditional relational symbol $R(A, B, C, Condition)$. Suppose that a DSE KB-solution J under $\mathfrak{S}$ consists of the pc-table instance below:*

R

A	B	C	$Condition$
x	y	y	$y = z$
x	x	z	$x = y$
x	a	a	$true$
b	y	b	$true$
c	c	z	$true$

By looking at J, we can see that there does not exist a DSE KB-solution J_1 under $\mathfrak{S}$ such that $J_1 \subset J$ and $J_1 \equiv J$, assuming that elements a, b, and c are three values that refer to three

different objects. In other words, we can see that J is in fact a MUDSE solution under $\mathfrak{S}$. Yet, we give below a more compact instance J' with pc-table R' such that $J' \equiv J$ and $J' \not\subset J$.

R'

A	B	C	$Condition$
x	y	z	$x = y \vee y = z$
x	a	a	$true$
b	y	b	$true$
c	c	z	$true$

Following the intuition in Example 6.3.1, MUDSE solutions might not be the most compact solutions in some DSE$_I$ instances. We denote the class of most compact solutions in terms of number of tuples in a DSE$_I$ setting by **core DSE KB-solutions**. Also, for some DSE$_I$ instances the class of MUDSE solutions and the class of core DSE KB-solutions coincide.

Notice that even though R and R' in Example 6.3.1 are equivalent, though they have different structures. Therefore, to check whether the DSE KB-solution J'' with pc-tables R' is also a DSE KB-solution under $\mathfrak{S}$ (that is $J'' \equiv J$), we need verify that both J and J'' instances have the same set of representations (i.e. $\mathsf{Rep}(R_1) = \mathsf{Rep}(R_2)$). In this context, the first result we state is the following:

Theorem 6.3.2 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a fixed DSE$_I$ setting where Σ_{st} is a set of st-tgds, I be a pc-table over $\mathbf{S}$ and J be a pc-table over $\mathbf{T}$. Consider that both , Σ_c^s and Σ_c^s are fixed sets of implicit data of type full tgds. Then, checking if J is a universal DSE KB-solution for I is DP_2^P-complete.*

Authors in [11] proved that pc-tables are the best tool to represent exchanging source instances with incomplete information. Now the question that we would like to ask is whether pc-tables are the best structures to represent the most compact DSE KB-solutions in a DSE$_I$ setting. We show in Proposition 6.3.3 below that there exists cases where the smallest instances equivalent to a pc-tables are *conditional tables* (c-table) which are more general and can include inequality formulas, along with equalities, in their conditions.

Proposition 6.3.3 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a fixed DSE$_I$ setting, I be a source instance, $\mathcal{M}$ be an st-mapping table, and Σ_{st} be a set of st-tgds. Let J be a DSE KB-solution for I and $\mathcal{M}$ under $\mathfrak{S}$. Then, the best tool to represent a core DSE KB-solution J' of J would be c-tables.*

We provide in what follow a counter example that proves Proposition 6.3.3. Consider a DSE$_I$ setting $\mathfrak{S}$ where the target schema $\mathbf{T}$ consists of the positive conditional relational symbol $R(A, B, C, Condition)$. Suppose that an instance J of $\mathbf{T}$ consists of the following pc-table instance:

R

A	B	C	$Condition$
x	y	z	$true$
w	x	1	$x = 3$
w	p	3	$y = 3$
u	v	3	$y = 2$
u	3	1	$x = 2$
v	u	1	$x = 1$

Notice that there does not exist a pc-table $R_1 \subset R$ such that $R \equiv R_1$. A possible core c-table of R is given in the following c-table instance R':

R'

A	B	C	$Condition$
x	y	z	$true$
w	3	1	$x = 3 \vee x = 2$
v'	u'	1	$x = 1$
u'	v'	3	$(x = 3 \wedge y = 2) \vee$ $(x = 2 \wedge y = 3) \vee (x = 1 \wedge y = 2)$
w	p	3	$(x = 2 \wedge y = 2) \vee$ $(x = 1 \wedge y = 3) \vee (x = 3 \wedge y = 3) \vee$ $(x \neq 3 \wedge x \neq 2 \wedge x \neq 1 \wedge y = 3) \vee$ $(x \neq 3 \wedge x \neq 2 \wedge x \neq 1 \wedge y = 2) \vee$

We have obtained the c-table R' such that $R' \equiv R$ by applying the following transformations to R:

1. We first expand the pc-table R to a c-table R_1 that is given below and that contains the set of all possible valuations of R, using the values of x and y which exist in the conditions of the tuples in R. These values are either of the following assignments: $x = 1$, $x = 2$, $x = 3$, $y = 2$, or $y = 3$.

   R_1

A	B	C	$Condition$
x	y	z	$true$
w	x	1	()
u	3	1	$(x = 2 \wedge y \neq 2 \wedge y \neq 3)$
v''	u''	1	$(x = 1 \wedge y \neq 2 \wedge y \neq 3)$
w	p	3	$x = 3 \wedge y = 3$
u'	v'	3	$x = 3 \wedge y = 2$
w_1	p_1	3	$x \neq 3 \wedge x \neq 2 \wedge x \neq 1 \wedge y = 3$
w'	p'	3	$x = 2 \wedge y = 3$
u	v	3	$x = 2 \wedge y = 2$
w''	p''	3	$x = 1 \wedge y = 3$
u''	v''	3	$x = 1 \wedge y = 2$
u_1	v_1	3	$x \neq 3 \wedge x \neq 2 \wedge x \neq 1 \wedge y = 2$

2. Then we transform R_1 and shrink it to R' by first replacing x with 3 in tuple $(w, x, 1 : (x = 3 \wedge y \neq 2 \wedge y \neq 3))$ in R_1, and then by reusing variables in values of attributes A, B, and C in different tuples in R_1 as much as possible such that the outcome of these transformations still generate c-tables equivalent to R_1.

Given a fixed DSE$_I$ setting $\mathfrak{S}$, a source instance I, and an st-mapping table $\mathcal{M}$. A problem of interest at this point is to find the data complexity of generating a smallest subset J' of a universal DSE KB-solution J for I and $\mathcal{M}$ under $\mathfrak{S}$ such that J' is also a universal DSE KB-solution under $\mathfrak{S}$. At first, it might look intuitive that given pc-table R, then generating the smallest subset R' of R such that $R' \equiv R$ is possible by applying the greedy algorithm introduced in ordinary DE settings [10].

Briefly, a greedy algorithm initially initializes R' to R, then checks for each tuple $t \in R$ whether $(R' - t) \equiv R$ holds. If so, it updates R' to be $R' - t$. However, we illustrate in Example 6.3.2 a situation where the greedy algorithm can fail to generate a smallest subset R' of R such that $R' \equiv R$.

Example 6.3.2 *Let $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ be a DSE$_I$ setting and J be a DSE KB-solution under $\mathfrak{S}$. Assume that J consists of the pc-table R given below:*

R

A	B	C	$Condition$
x	y	z	$true$
u	7	1	$x = 1$
u	5	1	$p = 2$
1	1	1	$u = 1$
2	2	2	$u = 1$
3	3	3	$u = 1$

Let $R_1 \subset R$ be the following pc-table:

R_1

A	B	C	$Condition$
x	y	z	$true$
u	7	1	$x = 1$
u	5	1	$p = 2$
3	3	3	$u = 1$

Also, Let $R_2 \subset R$ be the following pc-table:

R_2

A	B	C	$Condition$
x	y	z	$true$
u	7	1	$x = 1$
u	5	1	$p = 2$
1	1	1	$u = 1$
2	2	2	$u = 1$

Notice that $R_1 \equiv R$ *and* $R_2 \equiv R$. *However, notice that* $R - \{\langle 1,1,1 : u = 1\rangle, \langle 2,2,2 : u = 1\rangle, \langle 3,3,3 : u = 1\rangle\} \not\equiv R$. *Therefore, and intuitively, the order of removing tuples while applying the greedy algorithm [10] to* R *affects the generation of the smallest subset* R_1 *of pc-table* R *such that* $R_1 \equiv R$. *In other words, in order to check whether an arbitrary pc-table* R' *is the smallest subset pc-table of* R *and equivalent to* R, *we need to non-deterministically determine whether there exists an instance* $R'' \subset R$ *such that* $|R''| < |R'|$ *and* $R'' \equiv R$. □

In addition, following the result given in Theorem 6.3.2, and based on the fact that the order of tuples extractions from a DSE KB-solution J in a DSE$_I$ setting $\mathfrak{S}$ makes a difference when generating the smallest subset J' of J such that $J' \equiv J$, then the data complexity of the problem of generating J' becomes harder than the result given in Theorem 6.2.3. We summarize this in the following Theorem:

Theorem 6.3.4 *Let* $\mathfrak{S} = (\mathbf{S}, \mathbf{T}, \mathcal{M}, \Sigma_{st})$ *be a DSE$_I$ setting where* Σ_{st} *is a set of st-tgds,* (I, Σ_c^s) *a pc-table over* $\mathbf{S}$ *and* (J, Σ_c^t) *a pc-table over* $\mathbf{T}$. *To check whether there exists a KB-solution* (J', Σ_c^t) *for* I *and* $\mathcal{M}$ *under* $\mathfrak{S}$ *such that* $J' \subset J$ *is* $\Pi_2^{P\|}$*-complete.*

6.4 Summary

We have addressed in this chapter DSE$_E$ settings with more collaborating exchange scenarios where source instances can contain pc-tables. We have proven in this context that generating universal DSE KB-solutions is tractable. Also, we have proven the data complexity of checking whether a target instance J is a DSE KB-solution under a DSE$_I$ setting $\mathfrak{S}$ is DP_2^P*-complete.*

In addition, we have identified one type of solutions called core DSE KB-solutions, that are more compact than MUDSE solutions in some DSE_I instances. Finally, we have proved that the data complexity of checking whether there exists a DSE KB-solution J' with less number of tuples than a given DSE KB-solution J under a DSE_I setting $\mathfrak{S}$ is $\Pi_2^{P\|}$-$complete$.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Our thesis has addressed the problem of exchanging related information between applications that use different domains of constants to provide a consolidated view of integrated data using a unified set of vocabularies. To solve this problem, we introduced the data sharing and exchange setting. A DSE setting consists of a source schema $\mathbf{S}$, a target schema $\mathbf{T}$, a set of st-mapping dependencies Σ_{st}, and an st-mapping table $\mathcal{M}$. This setting complemented the schema mappings Σ_{st} adopted in DE settings with data level mappings in an st-mapping table $\mathcal{M}$ used in DC settings to solve the aforementioned problem.

Existing work in DC settings considered mapping tables with no particular interpretation of related data. Those mapping tables had general interpretations. Chapter 3 has shown one way of representing a DSE setting, with such general interpretations, by reducing it to an ordinary DE setting and augmenting the source schema with an st-mapping relation $\mathcal{M}$. Also, using the notion of universal solutions of DE settings, we defined universal DSE solutions as the most general solutions in a DSE setting. This representation however might not be the best in some cases, as we explained forward in Chapter 4. The reason is that computing the complete set of correct answers for some conjunctive queries can become expensive as the number of target elements mapped per source element increases. To go around this problem in DC settings,

authors in [5] introduced the notions of sound and complete certain answers for conjunctive queries as the semantics for conjunctive query answering. For some queries, computing the set of sound certain answers can be as informative as computing the complete set and can be less expensive in terms of run times. Therefore, and for the same reason, we adopted in a DSE setting sound and complete certain answers as the semantics for conjunctive queries answering.

In a DSE setting, we can compute complete certain answers for a conjunctive query using a universal DSE solution. However, the trickier question that arose is: what would be a good solution with which we can generate sound certain answers as well? We answered this question in Chapter 4 where we represented a DSE setting as a knowledge base exchange framework introduced in [11], while imposing the following condition on the st-mapping table $\mathcal{M}$: for each value $a \in (dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{S}})$, there exists an $a' \in (dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{T}})$ such that $\mathcal{M}(a, a')$ holds, and for no $b \in (dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{S}})$ – different than a – $\mathcal{M}(b, a')$ holds. We called this setting DSE with unique identity semantics and denoted this by DSE$_U$.

Representing a DSE$_U$ setting using the knowledge base framework makes it possible to store target instances using a portion of the set of exchanged data (explicit data) with a set of rules (implicit data), which can be used to regenerate the complete set of exchanged data when applied to the stored explicit data set. We called these combinations of explicit and implicit data universal DSE KB-solutions.

Also, following the notion of core universal solutions used in DE settings, we identified among universal DSE KB-solutions a class of most compact universal DSE KB-solutions. We called the latter minimal universal DSE KB-solutions, as they posses the minimum number of tuples and the smallest domain of constants. In addition, we have introduced an algorithm to generate such solutions in LOGSPACE. Posing a conjunctive query q to a minimal universal DSE KB-solution returns the set of sound certain answers of q. To compute the set of complete certain answers of q, on the other hand, two methods using universal DSE KB-solutions were proposed in Chapter 4.

In the final part of Chapter 4, we reported on a set of experiments which compare the growth in size and the times to generate universal DSE solutions versus minimal universal DSE

KB-solutions. We also compared the performance of both types of solutions when computing sound and complete certain answers for a sample set of conjunctive queries.

As is described in the literature [65], the main idea behind the use of a set of rules – implicit data – in a knowledge base is to complement the ability to store data with the possibility of computing additional information entailed by the semantics of the stored data and these rules at any time. Chapter 5 introduced a further data sharing and exchange setting with equivalence semantics of related data in an st-mapping table $\mathcal{M}$. We denoted it with DSE$_E$. DSE$_E$ setting turned out to be more like a knowledge base exchange setting than an ordinary data exchange setting. Pairings in $\mathcal{M}$ were interpreted as follows : a pair (a, b), where a is a source element and b is a target element, in an st-mapping table $\mathcal{M}$ means that a and b are considered by the target to be two different names that refer to the same object.

In Chapter 5, we have seen that, given a source instance I and an st-mapping table $\mathcal{M}$ in a DSE$_E$ setting $\mathfrak{S}$, I and $\mathcal{M}$ can be incomplete with respect to the equivalence semantics in $\mathcal{M}$ and additional implicit information entailed from these semantics would not exist in I and $\mathcal{M}$. For this reason, we defined the DSE$_E$ setting as a knowledge base exchange setting with a set of rules Σ_s^{c} in the source that could complete the source instant I and the st-mapping table $\mathcal{M}$ with the information entailed by the equivalence semantics. In addition, we defined a set of rules Σ_t^{c} in the target, same as in a DSE$_U$ setting, to generate complete certain answers of conjunctive queries.

We have adopted universal DSE KB-solutions introduced in Chapter 4 as semantics of a DSE$_E$ setting as well. Moreover, conjunctive query answering is based on these DSE KB-solutions. Finally, Chapter 5 also presented experimental results that compared the performance of Universal DSE solutions versus minimal universal DSE KB-solutions in a DSE$_E$ setting when computing sound and complete certain answers for conjunctive queries.

Chapter 6 addressed the data sharing and exchange problem in a collaborative setting where the source instance may contain null values. We defined the semantics of DSE$_E$ in a setting that combines the concepts of knowledge exchange and data exchange with incomplete information. We called this setting DSE with incomplete information in source instances and we

Data Complexity of Generating	DSE_U Setting	DSE_E Setting	DSE_I Setting
DSE Solution	LOGSPACE (Prop 3.3.2)	LOGSPACE (Cor 5.2.4)	PTIME (Th 6.2.3)
DSE KB-Solution	LOGSPACE (Cor 4.2.4)	LOGSPACE (Prop 5.2.5)	PTIME (Th 6.2.3)
Minimal DSE Solution	LOGSPACE (Th 4.3.3	LOGSPACE (Th 5.3.2)	$\Pi_2^{P\|\|}$-$complete$ (Th 6.3.4)
Sound Certain Answers	LOGSPACE (Th 4.4.6)	LOGSPACE (Th 5.4.2)	Open
Complete Certain Answers	LOGSPACE (Th 4.4.6)	LOGSPACE (Th 5.4.2)	Open

Table 7.1: Summary of Results

denoted it by DSE_I. The study of the main problems underlying DSE_I required applying and combining techniques from DE and KB exchange settings. We leveraged the results given in [11] to define the data complexity of generating DSE KB-solutions and that of generating a minimal one.

The main data complexity results in this thesis are summarized in Table 7.1, annotated with the corresponding Theorems, Propositions, or Corollaries. We have proven in Chapter 4 that generating DSE solutions, DSE KB-solutions, and MUDSE solutions is in LOGSPACE. This result mainly stems from the fact that applying a procedure based on the chase to generate a solution is in LOGSPACE. Also, we showed in Chapter 4 that there exists algorithms to compute sound and complete certain answers of conjunctive queries in LOGSPACE. In addition, we see in table 7.1 that this complexity does not increase in a DSE_E setting. The main reason goes to the deep result of Reinghold which show that computing the transitive closure of symmetric binary tables is in LOGSPACE. Also, we showed in Chapter 5 that there exists algorithms to compute sound and complete certain answers of conjunctive queries in LOGSPACE. In Chapter 6 we showed that generating DSE solutions and DSE KB-solutions is in PTIME in DSE_I settings. This result is based on the fact that generating KB-solutions in a KB exchange settings is in PTIME which was proved in [11]. We summarize those complexity results in Table 7.1. In addition, we summarize in Table 7.2 the semantics of each of the different DSE settings we introduced in our thesis.

Whether can generate each of the below	DSE Setting	DSE_U Setting	DSE_E Setting	DSE_I Setting
DSE Solution	Yes	Yes	Yes	Yes
DSE KB-Solution	No	Yes	Yes	Yes
Minimal DSE Solution	No	Yes	Yes	Yes
Sound Certain Answers	No	Yes	Yes	Open
Complete Certain Answers	Yes	Yes	Yes	Open

Table 7.2: A comparison of the different DSE settings and their semantics

7.2 Future Work

We propose below a list of possible extensions to the work done in this thesis.

1. **Optimizing the performance**: Our implementations of the concepts we proposed in both DSE_U and DSE_E settings did not aim at optimality in performance, but was intended to be a valuable direction of future work. It would be interesting to implement optimized algorithms which would be more efficient when generating minimal universal DSE KB-solutions and when computing complete certain answers for conjunctive queries.

2. **DSE settings with target constraints**: We have considered in this work DSE settings with no constraints in the target applications. Therefore, in such settings there always exist universal DSE solutions (or universal DSE KB-solutions) for a source instance. It would be interesting to define the semantics of DSE settings and the semantics of query answering when target constraints exist.

3. **DSE settings with incomplete information in the source**: We would like to address more the DSE setting where collaboration happens among several peers. In such settings, it is more likely that source peers contain information exchanged from other different peers, and hence may be, as seen in Chapter 6, in the form of naïve tables or pc-tables. Generating universal solutions in ordinary DE settings with incomplete source information has been proved in the literature to be a tractable problem. It would be interesting to find a way of generating a class a class of "preferable" compact DSE KB-solutions in polynomial time.

Bibliography

[1] R. Fagin, P. G. Kolaitisb, R. J. Miller, and L. Popa, "Data exchange: semantics and query answering," *Journal of the ACM*, pp. 89–124, 2005.

[2] M. Arenas, P. Barcelo, L. Libkin, and F. Murlak, "Relational and XML data exchange," *Morgan & Claypool Publishers*, 2010.

[3] M. Lawrence, R. Pottinger, and S. Staub-French, "Data coordination: Supporting contingent updates," *Proceedings of the VLDB Endowment*, pp. 831–842, 2011.

[4] P. A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu, "Data management for peer-to-peer computing: A vision," *Proceedings of the Workshop on the Web and Databases (WebDB'02)*, pp. 89–94, 2002.

[5] A. Kementsietsidis, M. Arenas, and R. J. Miller, "Mapping data in peer-to-peer systems: Semantics and algorithmic issues," *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pp. 325–336, 2003.

[6] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. J. Miller, and J. Mylopoulos, "The hyperion project: from data integration to data coordination," *ACM SIGMOD*, vol. 32, pp. 53–58, 2003.

[7] P. Rodriguez-Gianolli, M. Garzetti, L. Jiang, A. Kementsietsidis, I. Kiringa, M. Masud, R. J. Miller, and J. Mylopoulos, "Data sharing in the hyperion peer database system," *Proceeding VLDB 05 Proceedings of the 31st international conference on Very large data bases*, pp. 1291–1294, 2005.

[8] M. Masud, I. Kiringa, and A. Kementsietsidis, "Dont mind your vocabulary: Data sharing across heterogeneous peers," *Proceedings of the 2005 Confederated international conference on On the Move to Meaningful Internet Systems*, pp. 292–309, 2005.

[9] M. A. Rahman, M. M. Masud, I. Kiringa, and A. E. Saddik, "Bi-level mapping: Combining schema and data level heterogeneity in peer data sharing systems," *Proceedings of the 3rd Alberto Mendelzon International Workshop on Foundations of Data Management*, 2009.

[10] R. Fagin, P. G. Kolaitis, and L. Popa, "Data exchange: getting to the core," *ACM Trans. Database Syst*, vol. 30, no. 11, pp. 174–210, 2005.

[11] M. Arenas, J. Perez, and J. Reutter, "Data exchange beyond complete data," *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS 11)*, pp. 83–94, 2011.

[12] G. Gottlob, R. Pichler, and V. Savenkov, "Normalization and optimization of schema mappings," *The International Journal on Very Large Data Bases*, vol. 20, no. 2, pp. 277–302, 2011.

[13] R. Fagin, P. G. Kolaitis, and A. Nash, "Towards a theory of schema-mapping optimization," *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS 08)*, pp. 33–42, 2008.

[14] G. Mecca, P. Papotti, and S. Raunich, "Core schema mappings: Scalable core computations in data exchange," *Journal Information Systems*, vol. 37, no. 7, pp. 677–711, 2012.

[15] B. Marnette, G. Mecca, and P. Papotti, "++spicy: an open-source tool for second-generation schema mapping and data exchange," *The Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 1438–1441, 2011.

[16] A. Hernich, L. Libkin, and N. Schweikardt, "Closed world data exchange," *ACM Transactions on Database Systems*, vol. 36, no. 2, 2011.

[17] R. Chirkova, L. Libkin, and J. L. Reutter, "Tractable XML data exchange via relations," *Proceedings of the 20th ACM international conference on Information and knowledge management (CIKM 11)*, pp. 1629–1638, 2011.

[18] L. Popa, M. A. Hernandez, Y. Velegrakis, and R. J. Miller, "Mapping XML and relational schemas with clio," *Proceedings of the 18th International Conference on Data Engineering*, p. 498, 2002.

[19] R. Fagin, L. M. Haas, M. Hernandez, R. J. Miller, L. Popa, and Y. Velegrakis, "Clio: Schema mapping creation and data exchange," *Conceptual Modeling: Foundations and Applications*, pp. 198–236, 2009.

[20] R. Pichler and V. Savenkov, "Towards practical feasibility of core computation in data exchange," *Theoretical Computer Science*, vol. 411, no. 7-9, pp. 935–957, 2010.

[21] L. Libkin, "Data exchange and incomplete information," *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 60–69, 2006.

[22] F. Afrati, C. Li, and V. Pavlaki, "Data exchange: Query answering for incomplete data sources," *Proceedings of the 3rd international conference on Scalable information systems (InfoScale 08)*, 2008.

[23] T. Imieliski and J. Witold Lipski, "Incomplete information in relational databases," *In: Journal of the ACM*, pp. 761–791, 1984.

[24] M. Lenzerini, "Data integration: A theoretical perspective," *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 233–246, 2002.

[25] M. Friedman, A. Levy, and T. Millstein, "Navigational plans for data integration," *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 67–73, 1999.

[26] C. Beeri and M. Y. Vardi, "A proof procedure for data dependencies," *Journal of the ACM*, pp. 718–741, 1984.

[27] A. Deutsch, A. Nash, and J. Remmel, "The chase revisited," *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS 08)*, pp. 149–158, 2008.

[28] P. G. Kolaitis, J. Panttaja, and W.-C. Tan, "The complexity of data exchange," *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 30–39, 2006.

[29] R. Reiter, "On closed world databases," *Logic and Databases*, 1978.

[30] M. Arenas, "XML data exchange: consistency and query answering," *Journal of the ACM (JACM)*, vol. 55, no. 2, 2008.

[31] P. Barcelo, "Logical foundations of relational data exchange," *ACM SIGMOD Record*, vol. 38, no. 1, pp. 49–58, 2009.

[32] G. Gottlob and A. Nash, "Data exchange: Computing cores in polynomial time," *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS 06)*, pp. 40–49, 2006.

[33] R. van der Meyden, "Logical approaches to incomplete information: A survey," *Logics for Databases and Information Systems*, pp. 307–356, 1998.

[34] S. Abiteboul, R. Hull, and V. Vianu, "Foundations of databases," *Addison-Wesley*, 1995.

[35] M. Arenas, L. E. Bertossi, and J. Chomicki, "Query evaluation in almost consistent databases using residues," *Proceedings of the XVIII International Conference of the Chilean Computer Science Society (SCCC 98)*, pp. 8–14, 1998.

[36] M. Arenas, L. E. Bertossi, and J. Chomicki, "Consistent query answers in inconsistent databases," *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS 99)*, pp. 68–79, 1999.

[37] S. Abiteboul and O. M. Duschka, "Complexity of answering queries using materialized views," *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (PODS 98)*, pp. 254–263, 1998.

[38] A. Y. Levy, "Answering queries using views: A survey," *The International Journal on Very Large Data Bases*, vol. 10, no. 4, pp. 270–294, 2001.

[39] L. Bertossi and L. Bravo, "Query answering in peer-to-peer data exchange systems," *Proceedings of the 2004 international conference on Current Trends in Database Technology (EDBT 04)*, pp. 476–485, 2004.

[40] A. Fuxman, "Peer data exchange," *ACM Transactions on Database Systems*, vol. 31, no. 4, pp. 1454–1498, 2005.

[41] M. Arenas, E. Botoeva, D. Calvanese, V. Ryzhikov, and E. Sherkhonov, "Representability in dl-lite knowledge base exchange," *Description Logics*, 2012.

[42] T. Green and V. Tannen, "Models for incomplete and probabilistic information," *Proceedings of the 2006 international conference on Current Trends in Database Technology*, pp. 278–296, 2006.

[43] D. Calvanese, G. D. Giacomo, M. Lenzerini, and R. Rosati, "Logical foundations of peer-to-peer data integration," *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS 04)*, pp. 241–251, 2004.

[44] I. Tatarinov, Z. Ives, J. Madhavan, A. Halevy, D. Suciu, N. Dalvi, X. L. Dong, Y. Kadiyska, G. Miklau, and P. Mork, "The piazza peer data management project," *ACM SIGMOD Record*, vol. 32, no. 3, pp. 47–52, 2003.

[45] W. S. Ng, B. C. Ooi, K.-L. Tan, and A. Zhou, "Peerdb: A p2p-based system for distributed data sharing," *International Conference on Data Engineering (ICDE)*, pp. 633–644, 2003.

[46] G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, "On reconciling data exchange, data integration, and peer data management," *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS 07)*, pp. 133–142, 2007.

[47] L. Serafini, F. Giunchiglia, J. Mylopoulos, and P. A. Bernstein, "The local relational model: Model and proof theory," 2011.

[48] A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov, "Piazza: data management infrastructure for semantic web applications," *WWW '03 Proceedings of the 12th international conference on World Wide Web*, pp. 556–567, 2003.

[49] A. Kementsietsidis and M. Arenas, "Data sharing through query translation in autonomous sources," *Proceedings of the Thirtieth international conference on Very large data bases (VLDB 04)*, pp. 468–479, 2004.

[50] P. F. Patel-Schneider, P. Hayes, and I. Horrocks, "Ontology language," *W3C Recommendation*, 2004.

[51] P. Hayes, "RDF semantics," *W3C Recommendation*, 2004.

[52] A. Borgida, "On the relative expressiveness of description logics and predicate logics," *Artifficial Intelligence*, vol. 82, no. 1-2, pp. 353–367, 1996.

[53] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, "Data complexity of query answering in description logics," *Journal Artificial Intelligence*, vol. 195, pp. 335–360, 2013.

[54] C. Lutz, D. Toman, and F. Wolter, "Conjunctive query answering in the description logic el using a relational database system," *Proceedings of the 21st international jont conference on Artifical intelligence*, pp. 2070–2075, 2009.

[55] H. Perez-Urbina, B. Motik, and I. Horrocks, "Rewriting conjunctive queries over description logic knowledge bases," *In Proceedings of the International Workshop on Semantics in Data and Knowledge Bases*, pp. 199–214, 2008.

[56] R. Rosati, "On conjunctive query answering in el," *In Proceedings of the 2007 Description Logic Workshop (DL 2007)*, 2007.

[57] C. Lutz, D. Toman, and F. Wolter, "Conjunctive query answering in el using a database system," *The OWL: Experiences and Direction (OWLED) workshop series*, 2008.

[58] R. Kontchakov, C. Lutz, D. Toman, and F. Wolter, "The combined approach to ontology-based data access," *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*, pp. 2656–2661, 2011.

[59] T. Pankowski, "Using data-to-knowledge exchange for transforming relational databases to knowledge bases," *6th International Symposium, RuleML 2012*, pp. 256–263, 2012.

[60] M. Arenas, E. Botoeva, and D. Calvanese, "Knowledge base exchange," *Proceedings of Discription Logics*, 2011.

[61] M. Arenas, E. Botoeva, E. Sherkhonov, D. Calvanese, and V. Ryzhikov, "Exchanging description logic knowledge bases," *Proceedings of the Thirteenth International Conference (KR 2012)*, 2012.

[62] M. Arenas, P. Barcelo, and J. Reutter, "Query languages for data exchange: Beyond unions of conjunctive queries," *Proceedings of the 12th International Conference on Database Theory (ICDT 09)*, pp. 73–83, 2009.

[63] G. Gottlob, G. Orsi, and A. Pieris, "Ontological queries: Rewriting and optimization," *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering (ICDE 11)*, pp. 2–13, 2011.

[64] F. D. Pinto, D. Lembo, M. Lenzerini, and R. Mancini, "Optimizing query rewriting in ontology-based data access," *Proceedings of the 16th International Conference on Extending Database Technology*, pp. 561–572, 2013.

[65] B. Motik, I. Horrocks, and U. Sattler, "Bridging the gap between owl and relational databases," *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, no. 2, pp. 74–89, 2009.

[66] O. Reingold, "Undirected connectivity in log-space," *Journal of the ACM (JACM)*, vol. 55, no. 4, 2008.

[67] M. Y. Vardi, "The complexity of relational query languages," *In Proceedings of the fourteenth annual ACM symposium on Theory of computing, STOC*, pp. 137–146, 1982.

Appendix A

Proofs of Results

Theorem 4.1.1

Proof Let $\mathfrak{S}$ be a fixed DSE$_U$ setting and $\mathcal{M}$ be an st-mapping table. We prove in the following that generating C is in LOGSPACE. Let a be a constant value where $a \in (dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{T}})$. To show that computing elements in table C is in LOGSPACE, we show that checking whether $C(a)$ holds is in LOGSPACE.

We reduce the problem of checking whether $C(a)$ holds to the USTCON problem that has been shown by Reinghold in [66] to be solvable LOGSPACE.

The USTCON problem is a decision problem asking, for two vertices s and t in an undirected graph G, if t is reachable from s in G. We construct an undirected graph $G(C, \mathcal{M}, a)$ as follows:

- The nodes of G are a node for every element $b \in dom(\mathcal{M})$ in every data mapping in $\mathcal{M}$ (i.e. if b occurs in k different data mappings in $\mathcal{M}$, then G contains k nodes labelled b), in addition to a node labelled with a fresh element t where $t \notin (\mathsf{Const}^{\mathbf{S}} \cup \mathsf{Const}^{\mathbf{T}} \cup \mathsf{Var})$.
- There exists a edge between a node labelled $u \in (dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{S}})$ and the node labelled $v \in (dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{T}})$ in G if $\mathcal{M}(u, v)$ holds.

- There exists an edge from a node n in G where n is labelled with an element $e \in (dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{T}})$ to the node labeled t, if there does not exists a different node n' in G that is also labeled e.

Given the undirected graph G we constructed above, we say that for the element $a \in (dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{T}})$, $C(a)$ is true iff there exists a path from a node n labeled a to the node n' labeled t in G.

Proposition 4.2.2

Proof First we prove that J is a DSE KB-solution for I and $\mathcal{M}$ under $\mathfrak{S}$ if and only if $\Sigma_t^{\mathrm{c}}(J)$ is a DSE solution under $\mathfrak{S}$.

Assume first that J is a DSE KB-solution for I and $\mathcal{M}$ under $\mathfrak{S}$ and that $J' = \Sigma_t^{\mathrm{c}}(J)$. Then according to Definition 4.2.1 $(J' \cup \{\mathcal{M}\}) \in \mathsf{Mod}(J \cup \{\mathcal{M}\}, \Sigma_t^{\mathrm{c}})$ and for every $J'' \in Rep(J')$ it is the case that J'' is a solution [1] for I and $\mathcal{M}$ under $\mathfrak{S}$. Indeed, consider an arbitrary mapping ν from the set of nulls mentioned in J' into $\mathsf{Const}^{\mathbf{T}}$, such that $\nu(\perp)$ is not mentioned in $\mathcal{M}$, for each $\perp \in dom(J')$, and if $\perp_1$ and $\perp_2$ are two distinct nulls in $dom(J')$ then $\nu(\perp_1) \neq \nu(\perp_2)$. Let us define $\nu(J')$ as the target instance, without nulls, that is obtained from J' by simultaneously replacing each null $\perp \in dom(J')$ with $\nu(\perp)$. Thus, $\nu(J') \in Rep(J')$, and, hence, $\nu(J')$ is a DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$. But clearly then, for each st-mapping tgd in Σ_{st} with right-hand side $\exists w \psi(\bar{x}, \bar{w})$, whenever $\nu(J') \models \exists \bar{w} \psi(\bar{a}, \bar{w})$, for some tuple $\bar{a}$ of target constants mentioned in $\mathcal{M}$, it must also be the case that $J' \models \exists \bar{w} \psi(\bar{a}, \bar{w})$. This shows that J' is a solution [1] for I and $\mathcal{M}$ under $\mathfrak{S}$. Since also $\Sigma_t^{\mathrm{c}}(J') = J'$, we conclude that J' is a DSE solution for I and $\mathcal{M}$.

Now we prove that if $\Sigma_t^{\mathrm{c}}(J)$ is a DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$, then J is a DSE KB-solution for I and $\mathcal{M}$ under $\mathfrak{S}$. Take an arbitrary instance $(J' \cup \{\mathcal{M}'\}) \in \mathsf{Mod}(J \cup \{\mathcal{M}\}, \Sigma_t^{\mathrm{c}})$ and an arbitrary $J'' \in Rep(J')$. We will prove that J'' is a DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$. In order to prove this it is enough to prove that J' is a solution [1] for I and $\mathcal{M}$ under $\mathfrak{S}$. This is because there is a homomorphism from J' to J'' and solutions [1] are closed under

homomorphisms. Take an arbitrary st-mapping tgd $\phi(\bar{x}, \bar{y}) \wedge \mu(\bar{x}, \bar{z}) \rightarrow \exists \bar{w} \psi(\bar{z}, \bar{w})$. Assume that $(I \cup \{\mathcal{M}\}) \models \phi(\bar{a}, \bar{b}) \wedge \mu(\bar{a}, \bar{c})$ for arbitrary tuples of constants $\bar{a}$ and $\bar{b}$ in $\mathsf{Const}^{\mathbf{S}}$ and $\bar{c}$ in $\mathsf{Const}^{\mathbf{T}}$. Then since $\hat{J} := \Sigma_t^{\mathrm{c}}(J)$ is a solution [1] for I and $\mathcal{M}$ under $\mathfrak{S}$, it must be the case that $\hat{J} \models \exists \bar{w} \psi(\bar{c}, \bar{w})$. But then, since clearly $\hat{J} \subseteq J'$, it must also the case that $J' \models \exists \bar{w} \psi(\bar{c}, \bar{w})$. This finishes the proof since $(I \cup \{\mathcal{M}\}) \in \mathsf{Mod}(I \cup \{\mathcal{M}'\})$ and $\mathcal{M} \subseteq \mathcal{M}'$.

Next we prove that J is a universal DSE KB-solution if an only if $\hat{J} = \Sigma_t^{\mathrm{c}}(J)$ is a universal DSE solution. Assume first that $(J \cup \{\mathcal{M}\})$ is a universal DSE KB-solution. We know from the first part of the proof that $\hat{J}$ is a DSE solution. We prove next that it is also universal. Since $(J \cup \{\mathcal{M}\})$ is a universal DSE KB-solution, for every solution [1] J' for I and $\mathcal{M}$ under $\mathfrak{S}$, such that J' does not contain nulls, it is the case that $J' \in \mathit{Rep}(J'')$ and $(J'' \cup \{\mathcal{M}'\}) \in \mathsf{Mod}((J \cup \{\mathcal{M}\}), \Sigma_t^{\mathrm{c}})$, for some instance J'' over $\mathbf{T}$ and $\mathcal{M}' \subseteq \mathcal{M}$. Thus, $(J'' \cup \{\mathcal{M}\}) \in \mathsf{Mod}((J \cup \{\mathcal{M}\}), \Sigma_t^{\mathrm{c}})$. But since clearly $\hat{J} \subseteq J''$, it must be the case that for every solution J' for I and $\mathcal{M}$, such that J' does not contain nulls, it is the case that $J' \in \mathit{Rep}(\hat{J})$. Applying usual data exchange techniques (as in e.g. [2]), one can prove that this is equivalent to saying that $\hat{J}$ is a universal solution for I and $\mathcal{M}$ under $\mathfrak{S}$. Thus, $\hat{J}$ can be homomorphically mapped into any other solution for I and $\mathcal{M}$, and, in particular, it can be homomorphically mapped into any other DSE solution for I and $\mathcal{M}$. This shows that $\hat{J}$ is a universal DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$.

Assume finally that $\hat{J}$ is a universal DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$. From the first part of the proof we know that $(J \cup \{\mathcal{M}\})$ is a DSE KB-solution for I and $\mathcal{M}$ under $\mathfrak{S}$. We prove next that it is also universal. Consider an arbitrary $(I' \cup \{\mathcal{M}'\}) \in \mathsf{Mod}(I \cup \{\mathcal{M}\})$ and an arbitrary solution J' for I' and $\mathcal{M}'$ under $\mathfrak{S}$ such that $\mathit{dom}(J') \in \mathsf{Const}^{T}$ (i.e. J' contains no nulls). Then J' is also a solution for I and $\mathcal{M}$ (since $(I \cup \{\mathcal{M}\})$ is the minimal element in $\mathsf{Mod}(I \cup \mathcal{M})$). By applying usual data exchange techniques (as in e.g. [2]) it is possible to prove that $J' \in \mathit{Rep}(\hat{J})$, (because $\hat{J}$ is a universal solution for I and $\mathcal{M}$). Since $(\hat{J} \cup \{\mathcal{M}\}) \in \mathsf{Mod}(J \cup \{\mathcal{M}\})$ and $\mathcal{M} \subseteq \mathcal{M}'$, this finishes the proof of the proposition.

Theorem 4.3.2

Proof In the first part of this proof, we prove the correctness of the procedure $\texttt{CompMUDSE}_{\texttt{U}}\texttt{sol}_{\mathfrak{S}}$. Therefore, we prove that for an arbitrary result J^* of applying the algorithm $\texttt{CompMUDSE}_{\texttt{U}}\texttt{sol}_{\mathfrak{S}}$ procedure to a source instance I and an st-mapping table $\mathcal{M}$, J^* is a MUDSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$.

First we prove that J^* is a universal DSE solution. In order to prove this, it is sufficient to prove that J' that is the result of applying Σ_t^c to J^*, denoted by $J' = \Sigma_t^c(J^*)$, is such that $((I \cup \{\mathcal{M}\}), J') \vDash \Sigma_{st}$ in a DSE_U setting $\mathfrak{S}$. Clearly, $\Sigma_t^c(J') = J'$. Also, from the properties of relation $C(X)$, we know that each element $b \in dom(C)$, there exists a single element $a \in (dom(I) \cap \mathsf{Const}^{\mathbf{S}})$ such that $\mathcal{M}(a, b)$ holds, and for no element $c \in (dom(I) \cap \mathsf{Const}^{\mathbf{S}})$, $\mathcal{M}(c, b)$ holds. However, there can exist more than one element $d \in (dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{T}})$ that possesses the same properties as element b to uniquely identify the source element a.

Assume that the classes of C are $\{C_1, \ldots, C_m\}$ where c_1 and c_2 exist in C_i only if $\mathcal{M}(a, c_1)$ and $\mathcal{M}(a, c_2)$ hold for some element $a \in (dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{S}})$. Also, according to the definition of Σ_t^c, each element $b \in dom(C)$ is associated within the fresh binary table RELATED with each element $d \in (dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{T}})$ such that $\mathcal{M}(a, b)$ and $\mathcal{M}(a, d)$ hold for some $a \in (dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{S}})$.

Now, assume that J^* can be obtained from J' by first picking up witness $w_i \in dom(C)$ from the class C_i, for each $1 \leq i \leq m$, then by performing operation $\texttt{replace}(J', w_1, \ldots, w_m)$ and finally applying a core [10] algorithm to the result of $\texttt{replace}(J', w_1, \ldots, w_m)$.

Let J be an arbitrary universal DSE solution such that $((I \cup \{\mathcal{M}\}), J) \vDash \Sigma_{st}$. Then, clearly, $J' \subseteq J$. Intuitively, it is enough to prove that there exists a homomorphism from J to J' to prove that J' is also a universal DSE solution according to [10].

Let J_1 be the target instance obtained from J by performing operation $\texttt{replace}$ $(J, w_1, \ldots, w_m)$ and J^* be the result of applying a core [10] procedure on J_1. Since J^* is the core of J_1, it follows from [10] that there exists a homomorphism $h : J_1 \rightarrow J^*$. We now prove that h is a homomorphism also from J to J'. Let $R(a_1, \ldots, a_n)$ be an arbitrary fact in J, where for each a_i, $1 \leq i \leq n$, we have a_i is a null or $a_i \in (dom(J) \cap \mathsf{Const}^{\mathbf{T}})$. Let $R(b_1, \ldots, b_n)$ be a fact in J_1 obtained by simultaneously replacing each constant $c_i \in C_i$ with w_i, $1 \leq i \leq n$.

Now, from the definition of a core, we have that $R(h(b_1), \ldots, h(b_n)) \in J^*$. Applying Σ_t^c to J^* will generate in J' each fact that can be obtained from $R(h(b_1), \ldots, h(b_n))$ by simultaneously replacing, for each $1 \leq i \leq m$, the target constant w_i with an arbitrary target constant $c_i \in C_i$. Clearly, one of these facts is $R(h(a_1), \ldots, h(a_n))$, and, thus, $R(h(a_1), \ldots, h(a_n)) \in J'$.

Next we prove that J^* is also minimal among the class of universal DSE solutions. So, we first prove that there does not exist a proper subset J_1^* of J^* such that J_1^* is a universal DSE solution. To prove so, we first prove that $J' = \Sigma_t^c(J^*)$ is a core [10] of itself. That is, there is no homomorphism from J' to some J'' where $J'' \subset J'$.

Assume, for the sake of contradiction, that there exists a homomorphism $h : J' \rightarrow J''$, for some $J'' \subset J'$. It follows from well-known core properties [10] that (*) $dom(J'') \subset dom(J')$; that is, for each null $\perp' \in dom(J')$ there does not exist an element $d \in dom(J')$ where $d \neq \perp'$ such that $h(\perp') = d$. We prove next that there is also a homomorphism from J^* to some instance that is properly contained in J^*. This will be our desired contradiction, since J^* is a core of itself. Let us define a mapping $h' : dom(J^*) \rightarrow dom(J^*)$ as follows: (1) $h'(c) = c$, for each target constant $c \in dom(J^*)$; and (2) for each null $\perp \in dom(J^*)$ it is the case that $h'(\perp) = h(\perp)$, if $h(\perp)$ is a null, and $h'(\perp) = w_i \Leftrightarrow h(\perp) \in C_i$, for each $1 \leq i \leq m$. Clearly, h' is well defined since $dom(J') \cap \mathsf{Var} = dom(J^*) \cap \mathsf{Var}$. We prove first that h' is a homomorphism from J^* to the instance $J_1^* := \{R(t_1, \ldots, t_n) \mid R(t_1, \ldots, t_n) \in J^*$ and each t_i, $1 \leq i \leq n$, belongs to the range of $h' \}$. Assume that $R(b_1, \ldots, b_n) \in J^*$, where each b_i, $1 \leq i \leq n$, is either a null or a target constant. Then, since h is a homomorphism from J' into J' and $J^* \subseteq J'$, it must be the case that $R(h(b_1), \ldots, h(b_n)) \in J'$. Let $R(c_1, \ldots, c_n)$ be the fact that is obtained from $R(h(b_1), \ldots, h(b_n))$ by simultaneously replacing each target constant $c \in C_i$, $1 \leq i \leq m$, with w_i. It is clear that $R(c_1, \ldots, c_n) \in J^*$ (since $J' = \Sigma_t^c(J^*)$. Further, it is easy to see that $(c_1, \ldots, c_n) = (h'(b_1), \ldots, h'(b_n))$. It follows that $R(h'(b_1), \ldots, h'(b_n)) \in J_1^*$. Further, from remark (*) above, it must be the case that there is a null $\perp \in dom(J^*)$ such that $h'(\perp') \neq \perp$, for each null $\perp' \in dom(J^*)$. It follows that J_1^* is a proper subset of J^*.

Now, let us assume for the sake of contradiction that there exist a proper subset J_1^* of J^* such that J_1^* is a universal DSE solution. Then in order capture the contradiction, it is

sufficient to prove that $J'' = \Sigma_t^{\mathrm{c}}(J_1^*)$ is a proper subset of $J' = \Sigma_t^{\mathrm{c}}(J^*)$. Indeed, assume that this is the case. Then $J'' \subset J'$ and, from the definition of a universal DSE solution, it follows that $((I \cup \{\mathcal{M}\}), J'') \models \Sigma_{st}$ under $\mathfrak{S}$. This contradicts the fact we just proved above about J' being a core universal DSE solution such that $((I \cup \{\mathcal{M}\}), J') \models \Sigma_{st}$. We prove next that J'' is properly contained in J' (assuming that $J_1^* \subset J^*$). Let $R(a_1, ..., a_n)$ be a fact in $J^* \setminus J_1^*$, where for each a_i, $1 \leq i \leq n$, we have that a_i is either a null or $a_i \in (dom(J^*) \cap Const^{\mathbf{T}})$. Then it is not hard to see that $R(a_1, \ldots, a_n) \in J' \setminus J''$. Indeed, there is no way that the process Σ_t^{c} can generate the fact $R(a_1, \ldots, a_n)$ when applied on J_1^*.

We now prove that there is no universal MUDSE solution J_1^* such that $(dom(J_1^*) \cap \mathsf{Const}^{\mathbf{T}})$ is a proper subset of $(dom(J^*) \cap \mathsf{Const}^{\mathbf{T}})$. Assume for the sake of contradiction that such J_1^* exists. Then there is at least one equivalence class C_i, $1 \leq i \leq m$, such that no element from C_i belongs to $dom(J_1^*)$. Thus, no element from C_i belongs to the domain of $J_1' = \Sigma_t^{\mathrm{c}}(J_1^*)$. This implies that there is no homomorphism from J' into J_1'. This implies that J_1' is not a universal DSE solution for I and $\mathcal{M}$, which leads to a contradiction since J_1^* is a universal DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$.

Theorem 4.3.3

Proof We prove that the algorithm $\texttt{CompMUDSE}_{\texttt{U}}\texttt{sol}_{\mathfrak{S}}$ runs in LOGSPACE. Let $\mathcal{M}$ be a mapping table and a be a constant value where $a \in (dom(\mathcal{M}) \cap \mathsf{Const}^{\mathbf{T}})$.

1. As a first step, we proved in Theorem 4.1.1 that computing elements in table C is in LOGSPACE.

2. In the second step of the $\texttt{CompMUDSE}_{\texttt{U}}\texttt{sol}_{\mathfrak{S}}$ procedure, the chase [27] algorithm is applied. We have considered in this work DSE settings with no constraints in the target. Therefore, in such settings there always exists a universal DSE solution. In such a case in ordinary data exchange (DE) settings, and following the result in [2], given a fixed DE setting $\mathfrak{S}$, there is a procedure (based on the *chase* [27]) that computes a universal solution for each source instance I under $\mathfrak{S}$ in LOGSPACE.

3. Although steps 3,4 might seem non deterministic when choosing the possible set of witnesses; however, since each possible choice of witnesses leads to a MUDSE solution, we can easily deduce that these steps can be computed in LOGSPACE.

4. In the usual data exchange [10], authors introduced a simple algorithm – named *Greedy* algorithm [10] – that was proved in [2] to be computed in LOGSPACE.

5. Finally, since LOGSPACE is closed under composition [62], we conclude that the procedure $\texttt{CompMUDSE}_{\texttt{U}}\texttt{sol}_{\mathfrak{S}}$ runs in LOGSPACE.

Proposition 4.4.4

Proof Let $\mathfrak{S}$ be a DSE$_U$ setting, I a source instance, and $\mathcal{M}$ an st-mapping table. Also, let $Q(\bar{x}) = (\bar{x})\exists\bar{y}\ \phi(\bar{x},\bar{y}) \wedge \psi(\bar{y})$ be a conjunctive query over $\mathbf{T}$. Then, as we mentioned previously, $Q'(x_1,\ldots,x_n) = (\bar{x})\exists\bar{y}\exists\bar{w}\ \phi(\bar{x},\bar{y}) \wedge \psi'(\bar{y})$ where $\psi'(\bar{y})$ constitutes RELATED$(y_1, y') \wedge$RELATED(y_2, y') for each equality formula $y_1 = y_2$ in $\psi(\bar{y})$ returns a set of sound certain answers, when applied to a MUDSE solution J, which represents in a precise way the complete set. In addition, to complete the evaluation of Q' on J, we compute the answer of $\hat{Q}'(z_1,\ldots,z_n) = Q'(x_1,\ldots,x_n) \wedge \bigwedge_{i=1}^{n}$ RELATED(x_i, z_i) using J. Assume that $\mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q) = \{t_1,\ldots,t_n\}$.

To prove that $\mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q) = \hat{Q}'(J)$, we first prove that each $t_i \in \mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, we have t_i in $\hat{Q}'(J)$, $1 \leq i \leq n$. Then, we prove the opposite case. We denote by $a \sim b$ elements a and b in $\mathcal{M}$, where $\{a, b\} \subseteq \mathsf{Const}^{\mathbf{T}}$ such that $\mathcal{M}(c, a)$ and $\mathcal{M}(c, b)$ hold for some element $c \in \mathsf{Const}^{\mathbf{S}}$. Also, we denote by $t_1 \sim t_2$ two tuples such that (1) t_1 and t_2 have the same arity, say p; and (2) $t_1[r] \sim t_2[r]$, $1 \leq r \leq p$.

Case 1: We want to prove that for each tuple $t_i \in \mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, we have t_i in $\hat{Q}'(J)$, $1 \leq i \leq n$. Given that $\mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q) = \{t_1,\ldots,t_n\}$ and J is a MUDSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$. Let $J_1 = \Sigma_t^{\mathrm{c}}(J)$. Then, we can deduce that J_1 is a universal DSE solution which contain the complete set of exchanged data from I; that is

$(I \cup \{\mathcal{M}\}, J_1) \vDash \Sigma_{st}$, and $\mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q) = Q(J)$. Also, from the definition of the completion program Σ_t^{c}, we can deduce the following: for each arbitrary fact $R(a_1, \ldots, a_q)$ in J, we have a fact $R(b_1, \ldots, b_q)$ in J_1 such that, for each $a_i \in (dom(J) \cap \mathsf{Const}^{\mathbf{T}})$, $1 \leq i \leq q$, we have $b_i \in (dom(J) \cap \mathsf{Const}^{\mathbf{T}})$, $1 \leq k \leq q$, and $\mathcal{M}(c, a_i)$, $\mathcal{M}(c, b_i)$, $1 \leq i \leq q$, hold for some $c \in \mathsf{Const}^{\mathbf{S}}$. We say a_i and b_i belong to the same class C_i over C, $1 \leq i \leq r$. In addition, we have $Q(J_1) = \{t_1, \ldots, t_n\}$. This means, there exists a homomorphism $h : \phi(\bar{x}, \bar{y}) \wedge \psi(\bar{y}) \rightarrow J_1$ and $h(\bar{x}) = \{t_1, \ldots, t_n\}$. Let $h(\bar{x}, \bar{y}) = \{t'_1, \ldots, t'_n\}$ where $t_i \subset t'_i$ and $t_i = t'_i[\bar{x}]$, $1 \leq i \leq n$.

Now, assume for the sake of contradiction that $\hat{Q}'(J) \subset \mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, and that there exists a fact t_i for some i, $1 \leq i \leq n$, where $t_i \notin \hat{Q}'(J)$. Let g be a homomorphism $g : \phi(\bar{x}, \bar{y}) \wedge \psi'(\bar{y}) \rightarrow J$, where $\psi'(\bar{y})$ constitutes $\textsc{Related}(y_1, y') \wedge \textsc{Related}(y_2, y')$ for each equality formula $y_1 = y_2$ in $\psi(\bar{y})$. Then, completing $g(\bar{x}, \bar{y})$ with elements declared related in $\textsc{Related}$ to $\bar{x}$ and $\bar{y}$, will generate a set of facts $\{t'_j, \ldots, t'_k\}$, $1 \leq j, k \leq n$ and $j, k \neq i$, since $t_i \notin \hat{Q}'(J)$. This means, there exists at least one term in $\psi(\bar{y})$, $y_1 = y_2$, where y_1, y_2 are in $\bar{y}$, such that: (a) there does not exist a fact $t'_p \in g(\bar{x}, \bar{y})$ where $J \vDash \phi(t'_p) \wedge \textsc{Related}(t'_p[y_1], y') \wedge \textsc{Related}(t'_p[y_2], y')$, for some $y' \in \mathsf{Const}^{\mathbf{T}}$; in other words, the following do not hold: $\mathcal{M}(w, t'_p[y_1]) \wedge \mathcal{M}(w, t'_p[y_2]) \cdots \wedge \mathcal{M}()$ for some constant $w \in dom(I)$, (and) (b) $t'_p \sim t'_i$. Let us assume that this is the case. We know that $J \subseteq J_1$. Then, there exists a homomorphism $f : J \rightarrow J_1$. Since homomorphism is closed under concatenation, then $f(g(\bar{x}, \bar{y})) \in Q'(J_1)$. But from the definition of a universal DSE solution, we know that for each fact t such that $J_1 \vDash \phi(t[\bar{x}], t[\bar{y}]) \wedge \textsc{Related}(t[y_1], y') \wedge \textsc{Related}(t[y_2], y') \cdots \wedge \textsc{Related}()$, for some constant value $y' \in \mathsf{Const}^{\mathbf{T}}$, there exists a fact t' such that $t[\bar{x}] = t'[\bar{x}]$, and $t'[y_1] = t'[y_2]$. This directly means that, there exists a fact $t' \in h(\bar{x}, \bar{y})$ such that $J_1 \vDash \phi(t[\bar{x}], t[\bar{y}]) \wedge \psi(\bar{y})$, and $t' = t_i$.

Case 2: We want to prove that for each tuple t_i in $\hat{Q}'(J)$, we have t_i in $\mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, $1 \leq i \leq n$. Assume for the sake of contradiction that there exists a tuple $t_i \in \hat{Q}'(J)$, and $t_i \notin \mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$. Let J_1 be an arbitrary universal DSE solution such that $((I \cup \{\mathcal{M}\}), J_1) \vDash \Sigma_{st}$. Then, from the definition of certain answers, we can deduce that $Q(J_1) = \{t_1, \ldots, t_n\}$. Let g be a homomorphism $g : \phi(\bar{x}, \bar{y}) \wedge \psi(\bar{y}) \rightarrow J_1$, where $g(\bar{x}, \bar{y}) =$

$\{t'_1, \ldots, t'_n\}$ and $g(\bar{x}) = \{t_1, \ldots, t_n\}$. Let h be a homomorphism $h : \phi(\bar{x}, \bar{y}) \wedge \psi'(\bar{y}) \rightarrow J$, where $\psi'(\bar{y})$ constitutes RELATED$(y_1, y') \wedge$RELATED(y_2, y') for each equality formula $y_1 = y_2$ in $\psi(\bar{y})$, and $h(\bar{x}, \bar{y}) = \{t'_j, \ldots, t'_k\}$.

Let K be the set of tuples that resulted from applying Σ_t^c to the set of tuples in $h(\bar{x}, \bar{y})$ using RELATED. Then, K will contain for each $t \in \{t'_j, \ldots, t'_k\}$, all the possible tuples t' such that $t \sim t'$. Also, since $((I \cup \{\mathcal{M}\}), J_1) \vDash \Sigma_{st}$, then J_1 contains the complete set of exchanged data, and for each fact $R(t) \in J_1$, we have all the possible facts $R(t')$ such that $R(t) \sim R(t')$ according to $\mathcal{M}$. Let J^* be an arbitrary universal DSE solution such that $((I \cup \{\mathcal{M}\}), J^*) \vDash \Sigma_{st}$. Let f be a homomorphism $f : \phi(\bar{x}, \bar{y}) \wedge \psi(\bar{y}) \rightarrow J^*$ and f' be a homomorphism $f' : \phi(\bar{x}, \bar{y}) \wedge \psi'(\bar{y}) \rightarrow J$. Since $J \subseteq J_1$, then, for each tuple $t'' \in h(\bar{x}, \bar{y})$, we have $t'' \in f'(\bar{x}, \bar{y})$. In addition, $f'(\bar{x}, \bar{y})$ contains for each tuple t a tuple t' such that $t \sim t'$. This directly leads to the fact that $f'(\bar{x}, \bar{y}) = K$. Also, since K is complete with respect to $\mathcal{M}$, then for each tuple $t'' \in K$, there exists a tuple $t''' \in K$ such that $t'' \sim t'''$, $t''[\bar{x}] \sim t'''[\bar{x}]$, and $J \vDash \phi(t'''[\bar{x}], t'''[\bar{y}]) \wedge \psi(t'''[\bar{y}])$. Finally, this means that for each tuple $t'' \in K$, there exists a tuple $t''' \in K$ such that $t'' \sim t'''$ and $t''' \in f(\bar{x}, \bar{y})$. Also, we know that for all universal DSE solutions J' where $((I \cup \{\mathcal{M}\}), J') \vDash \Sigma_{st}$ is true, these solutions are homomorphically equivalent. Then, we can directly conclude that there exists a homomorphism $e : J^* \rightarrow J_1$, and $e(f(\bar{x}, \bar{y})) \in g(\bar{x}, \bar{y})$. Consequently, $e(f(\bar{x})) \in g(\bar{x})$. Since certain answers have facts with only constants and homomorphisms are identity on constants, then, there does not exist a fact $t_i \in \{t_1, \ldots, t_n\}$ such that $t_i \notin g(\bar{x})$. This is equivalent to saying that each fact $t_i \in \{t_1, \ldots, t_n\}$, we have $t_i \in q(J_1)$, $1 \leq i \leq n$, and this falsifies the contradiction.

Proposition 4.4.6

Proof The fact that computing the set of sound and complete certain answers, sound-certain$_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ and complete-certain$_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, of a conjunctive Q is in LOGSPACE, is based on the following facts : (1) following Corollary 4.2.4 and Corollary 4.2.3, generating a universal DSE solution is in LOGSPACE; and (2) following Theorem 4.3.3, generating a MUDSE solution is in LOGSPACE; and finally (3) it is known from [67] that the data complexity of any

FO formula is in LOGSPACE, and thus checking if a fixed conjunctive query is satisfied in a database instance is in LOGSPACE.

Proposition 5.2.3

Proof The proof of this proposition follows the same reasoning provided above in the proof of Proposition 4.2.2.

Proposition 5.2.5

Proof We prove now that the $\texttt{CompUnivDSESol}_{\mathfrak{S}}$ procedure computes a universal DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$ in LOGSPACE.

First, in step 1 of $\texttt{CompUnivDSESol}_{\mathfrak{S}}$, the source completion process Σ_s^c is applied to I and $\mathcal{M}$ and returns as outcome the source instance $\hat{I}$ and the st-mapping table $\hat{\mathcal{M}}$.

In Σ_s^c (in step 1 to step 4), we compute the transitive closures of the *symmetrical* binary table EQUAL. Computing the transitive closure of symmetrical binary relations is solvable in LOGSPACE as given by a very deep result of Reinghold [66]. Steps 5 and 6 of Σ_s^c compute the complete instances $\hat{I}$ and the st-mapping table $\hat{\mathcal{M}}$ using EQUAL. This step can be computed by applying the naive chase procedure to I and $\mathcal{M}$ using rules 5 and 6 of Σ_s^c. Following the result in [1] that a naive chase procedure runs in LOGSPACE, we can deduce that generating $\hat{I}$ and $\hat{\mathcal{M}}$ using steps 5 and 6 of Σ_s^c is in LOGSPACE.

Second, step 2 of $\texttt{CompUnivDSESol}_{\mathfrak{S}}$ procedure can be reduced to the problem of computing a universal solution for $(\hat{I} \cup \{\hat{\mathcal{M}}\})$ in a fixed DE setting $\mathfrak{S}$. Consequently, similar to steps 5 and 6 in Σ_s^c, this process works in LOGSPACE in a fixed DSE setting.

Finally, since LOGSPACE is closed under composition [62], we conclude that $\texttt{CompUnivDSESol}_{\mathfrak{S}}$ is in LOGSPACE.

Theorem 5.3.1

Proof To prove the correctness of $\texttt{CompMUDSE}_{\texttt{E}}\texttt{sol}_{\mathfrak{S}}$ procedure, we prove that for an arbitrary result J^* of $\texttt{CompMUDSE}_{\texttt{E}}\texttt{sol}_{\mathfrak{S}}(I, \mathcal{M})$, J^* is a minimal universal DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$. Let the outcome of Σ_s^c on I be $\hat{I}$ and on $\mathcal{M}$ be $\hat{\mathcal{M}}$.

First we prove that J^* is a universal DSE solution. In order to prove this, it is sufficient to prove that J' that is the result of applying Σ_t^c to J^*, denoted by $J' = \Sigma_t^c(J^*)$, is a universal pre-solution for $\hat{I}$ and $\hat{\mathcal{M}}$. Clearly, $\Sigma_t^c(J') = J'$. Thus, we only need to prove that J' is a universal pre-solution for $\hat{I}$ and $\hat{\mathcal{M}}$. It follows from the properties of the core, that it is enough to prove that J' is the core of J, where J is a canonical universal pre-solution for $\hat{I}$ and $\hat{\mathcal{M}}$. Since clearly $J' \subseteq J$, we need to prove that there is a homomorphism from J to J', but there is no homomorphism from J' to some J'' that is properly contained in J'.

Assume that the $\sim$-equivalence classes of J are $\{C_1, \ldots, C_m\}$ where c_1 and c_2 exist in C_i only if $c_1 \sim c_2$ as specified by $\hat{\mathcal{M}}$. Further, assume that J^* was obtained from J by picking up witness $w_i \in dom(J) \cap \mathsf{Const}^{\mathbf{T}}$ from the equivalence class C_i, for each $1 \leq i \leq m$, performing operation $\texttt{replace}(J, w_1, \ldots, w_m)$ and then applying a core [10] procedure on $\texttt{replace}(J, w_1, \ldots, w_m)$. Let J_1 be the result of the operation $\texttt{replace}(J, w_1, \ldots, w_m)$. Since J^* is the core of J_1, it follows from the properties of the core that there exists a homomorphism $h : J_1 \rightarrow J^*$. We now prove that h is a homomorphism also from J to J'. Clearly, the domain of h is $dom(J)$. Let $R(a_1, \ldots, a_n)$ be an arbitrary fact in J, where for each a_i, $1 \leq i \leq n$, we have a_i is a null or $a_i \in dom(J) \cap \mathsf{Const}^{\mathbf{T}}$. Let $R(b_1, \ldots, b_n)$ be a fact in J_1 obtained by simultaneously replacing each constant $c_i \in C_i$ with w_i, $1 \leq i \leq n$. Now, from the definition of a core, we have that $R(h(b_1), \ldots, h(b_n)) \in J^*$. Applying Σ_t^c to J^* will generate in J' each fact that can be obtained from $R(h(b_1), \ldots, h(b_n))$ by simultaneously replacing, for each $1 \leq i \leq m$, the target constant w_i with an arbitrary target constant $c_i \in C_i$. Clearly, one of these facts is $R(h(a_1), \ldots, h(a_n))$, and, thus, $R(h(a_1), \ldots, h(a_n)) \in J'$.

Next we prove that there is no homomorphism from J' to some J'' where $J'' \subset J'$. Assume, for the sake of contradiction, that there exists a homomorphism $h : J' \rightarrow J''$, for some $J'' \subset J'$. It follows from well-known core properties [10] that (*) $dom(J'') \subset dom(J')$; that is, for each null $\perp' \in dom(J')$ there does not exist an element $d \in dom(J')$ where $d \neq \perp'$ such that

$h(\perp') = d$. We prove next that there is also a homomorphism from J^* to some instance that is properly contained in J^*. This will be our desired contradiction, since J^* is a core of itself. Let us define a mapping $h' : dom(J^*) \rightarrow dom(J^*)$ as follows: (1) $h'(c) = c$, for each target constant $c \in dom(J^*)$; and (2) for each null $\perp \in dom(J^*)$ it is the case that $h'(\perp) = h(\perp)$, if $h(\perp)$ is a null, and $h'(\perp) = w_i \Leftrightarrow h(\perp) \in C_i$, for each $1 \leq i \leq m$. Clearly, h' is well defined since $dom(J') \cap \mathsf{Var} = dom(J^*) \cap \mathsf{Var}$. We prove first that h' is a homomorphism from J^* to the instance $J_1^* := \{R(t_1, \ldots, t_n) \mid R(t_1, \ldots, t_n) \in J^*$ and each t_i, $1 \leq i \leq n$, belongs to the range of $h'\}$. Assume that $R(b_1, \ldots, b_n) \in J^*$, where each b_i, $1 \leq i \leq n$, is either a null or a target constant. Then, since h is a homomorphism from J' into J' and $J^* \subseteq J'$, it must be the case that $R(h(b_1), \ldots, h(b_n)) \in J'$. Let $R(c_1, \ldots, c_n)$ be the fact that is obtained from $R(h(b_1), \ldots, h(b_n))$ by simultaneously replacing each target constant $c \in C_i$, $1 \leq i \leq m$, with w_i. It is clear that $R(c_1, \ldots, c_n) \in J^*$ (since $J' = \Sigma_t^{\mathrm{c}}(J^*)$. Further, it is easy to see that $(c_1, \ldots, c_n) = (h'(b_1), \ldots, h'(b_n))$. It follows that $R(h'(b_1), \ldots, h'(b_n)) \in J_1^*$. Further, from remark (*) above, it must be the case that there is a null $\perp \in dom(J^*)$ such that $h'(\perp') \neq \perp$, for each null $\perp' \in dom(J^*)$. It follows that J_1^* is a proper subset of J^*.

Next we prove that J^* is also minimal among the class of universal DSE solutions. We first prove that there does not exist a proper subset J_1^* of J^* such that J_1^* is a universal DSE solution. Assume otherwise. Then in order to get a contradiction it is sufficient to prove that $J'' = \Sigma_t^{\mathrm{c}}(J_1^*)$ is a proper subset of $J' = \Sigma_t^{\mathrm{c}}(J^*)$. Indeed, assume that this is the case. Then $J'' \subset J'$ and, from the definition of a universal DSE solution, it follows that J'' is a universal pre-solution for $\hat{I}$ and $\hat{\mathcal{M}}$ under $\mathfrak{S}$. This contradicts the fact just proved above that J' is a core of J. We prove next that J'' is properly contained in J' (assuming that $J_1^* \subset J^*$). Let $R(a_1, ..., a_n)$ be a fact in $J^* \setminus J_1^*$, where for each a_i, $1 \leq i \leq n$, we have that a_i is either a null or $a_i \in dom(J^*) \cap Const^{\mathbf{T}}$. Then it is not hard to see that $R(a_1, \ldots, a_n) \in J' \setminus J''$. Indeed, there is no way that the process Σ_t^{c} can generate the fact $R(a_1, \ldots, a_n)$ when applied on J_1^*.

We now prove that there is no universal MUDSE solution J_1^* such that $dom(J_1^*) \cap \mathsf{Const}^{\mathbf{T}}$ is a proper subset of $dom(J^*) \cap \mathsf{Const}^{\mathbf{T}}$. Assume for the sake of contradiction that such J_1^* exists. Then there is at least one equivalence class C_i, $1 \leq i \leq m$, such that no element from

C_i belongs to $dom(J_1^*)$. Thus, no element from C_i belongs to the domain of $J_1' = \Sigma_t^c(J_1^*)$. This implies that there is no homomorphism from J' into J_1'. This implies that J_1' is not a universal DSE solution for I and $\mathcal{M}$, which leads to a contradiction since J_1^* is a universal DSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$.

Theorem 5.3.2

Proof We prove that $\texttt{CompMUDSEsol}_{\mathfrak{S}}$ runs in LOGSPACE. For step 1, same as the case in Σ_s^c and following the result in [66], computing step 1 is in LOGSPACE. Also, in step 2, step 3 and step 4, the $\texttt{CompMUDSEsol}_{\mathfrak{S}}$ procedure seems to be non-deterministic since it involves choosing a set of witnesses $\{w_1, \ldots, w_m\}$ for $\{C_1, \ldots, C_m\}$. Clearly, different sets of witnesses may yield different target instances. However, each possible choice of witnesses leads to a minimal universal DSE solution. Therefore, steps 2,3 and 4 can be computed in LOGSPACE. Also, in step 5 we can replace a target constant $c \in C_i \cap dom(\hat{\mathcal{M}})$ with $w_i \in C_i$ by simply checking if EQUAL(c, w_i) holds, and this is done in LOGSPACE. In addition, according to [1, 10], applying the chase and generating cores are computed in LOGSPACE by applying the $naive$ chase and the simple $Greedy$ algorithm. Finally, since LOGSPACE is closed under composition [62], we can deduce that the procedure $\texttt{CompMUDSEsol}_{\mathfrak{S}}$ is computed in LOGSPACE.

Theorem 5.3.3

Proof We show in the following that for any two MUDSE solutions J_1 and J_2 for I and $\mathcal{M}$ under $\mathfrak{S}$, it is the case that J_1 and J_2 are isomorphic up to renaming of nulls and replacement of each constant c with another constant c' in the same equivalence class. Let the outcome of Σ_s^c on I be $\hat{I}$ and on $\mathcal{M}$ be $\hat{\mathcal{M}}$. To prove this,

First, we prove that there exists a one-to-one function $f : dom(J_1) \rightarrow dom(J_2)$ such that for each $v \in dom(J_1)$ and $v \in Const^{\mathbf{T}} \Leftrightarrow f(v) \in Const^{\mathbf{T}}$.

Let J_1' be the result of applying the rules in Σ_t^c to J_1 and J_2' be the result of applying the rules in Σ_t^c to J_2. From the definition of a universal DSE solution we provided previously,

we know that $J'_1 \in \mathsf{Mod}((J_1 \cup \{\mathcal{M}\}), \Sigma_t^{\mathrm{c}})$ and $J'_2 \in \mathsf{Mod}((J_2 \cup \{\mathcal{M}\}), \Sigma_t^{\mathrm{c}})$. Also, according to this definition, J'_1 and J'_2 are universal DSE solutions for I and $\mathcal{M}$ under $\mathfrak{S}$, and both are universal pre-solutions for $(\hat{I} \cup \{\hat{\mathcal{M}}\})$ under $\mathfrak{S}$. In other words, $((\hat{I} \cup \{\hat{\mathcal{M}}\}), J'_1) \vDash \Sigma_{st}$ and $((\hat{I} \cup \{\hat{\mathcal{M}}\}), J'_2) \vDash \Sigma_{st}$. Therefore, we can deduce that J'_1 and J'_2 are homomorphically equivalent. Also, we can deduce that $dom(J'_1) \cap Const^{\mathbf{T}} = dom(J'_2) \cap Const^{\mathbf{T}}$. Assume that the $\sim$-equivalence classes defined on J'_1 by $\hat{\mathcal{M}}$ are $\{C_1, \ldots, C_m\}$. Assume also that J_1 contains the fact $R(\ldots a \ldots)$ where $a \in dom(J_1) \cap \mathsf{Const}^{\mathbf{T}}$. Consider that $a \in C_i$, for some i $1 \leq i \leq n$. Applying Σ_t^{c} to J_1 generate in J'_1 the facts $R(\ldots c \ldots)$ for every constant $c \in C_i$. From the properties of J'_1 and J'_2 which we stated above, we can deduce that the facts $R(\ldots h(c) \ldots)$ constitute J'_2 because J'_1 and J'_2 are homomorphically equivalent. Consequently, J_2 should contain a fact $R(\ldots b \ldots)$ such that applying Σ_t^{c} to J_2 generate in J'_2 the facts $R(\ldots h(c) \ldots)$. Also, since J_1 is a MUDSE solution, then there does not exist two tuples $R(c_1 \ldots c_n)$ and $R(d_1 \ldots d_n)$ in J_1 such that $c_i \sim d_i$, $1 \leq i \leq n$, for each $c_i, d_i \in dom(J_1) \cap Const^{\mathbf{T}}$. Therefore, applying Σ_t^{c} to $R(c_1 \ldots c_n)$ and $R(d_1 \ldots d_n)$ will never generate a same tuple $R(a_1 \ldots a_n)$ where $a_i \in dom(J'_1) \cap Const^{\mathbf{T}}$, $a_i \sim d_i$, and $a_i \sim c_i$, $1 \leq i \leq n$. Thus, intuitively, we can deduce that J'_1 is a core universal DSE solution, in reference to cores in [10]. Same idea applies to J'_2. Also, cores are known as isomorphic from [10]. Therefore, $h : dom(J'_1) \rightarrow dom(J'_2)$ is one-to-one. Thus, for each $v \in dom(J'_1) \cap Const^{\mathbf{T}} \Leftrightarrow h(v) \in dom(J'_2) \cap Const^{\mathbf{T}}$. This means, if J'_1 contains a fact $R(\ldots c \ldots)$ where $c \in Const^{\mathbf{T}}$, then J'_2 should contain the fact $R(\ldots h(c) \ldots)$ and $h(c) \in Const^{\mathbf{T}}$. Consequently, if we have in J_1 $R(\ldots a \ldots)$ and $a \in \mathsf{Const}^{\mathbf{T}}$, then J_2 should contain $R(\ldots b \ldots)$ where $b = f(a)$ and $b \in \mathsf{Const}^{\mathbf{T}}$ such that applying Σ_t^{c} to $R(\ldots a \ldots)$ and $R(\ldots b \ldots)$ generate the set of facts $R(\ldots c \ldots)$ and $R(\ldots h(c) \ldots)$ in J'_1 and J'_2 respectively having the sets c and $h(c)$ in $\mathsf{Const}^{\mathbf{T}}$. In addition, since h is one-to-one, then for each $h(c)$ in $R(\ldots h(c) \ldots)$ in J'_2 and $h(c) \in \mathsf{Const}^{\mathbf{T}}$, it is that each c in $R(\ldots c \ldots)$ in J'_1, we have $c \in \mathsf{Const}^{\mathbf{T}}$. To reach this result, it should apply that if J_2 contains a fact $R(\ldots b \ldots)$ and $b \in \mathsf{Const}^{\mathbf{T}}$, then J_1 should have the fact $R(\ldots a \ldots)$ where $f(a) = b$ and $a \in \mathsf{Const}^{\mathbf{T}}$.

Now we prove that $f : dom(J_1) \rightarrow dom(J_2)$ is such that, for each $c \in dom(J_1) \cap Const^{\mathbf{T}}$

it is the case that $c \sim f(c)$. Again we assume that J_1 contains the fact $R(\ldots a \ldots)$ where $a \in dom(J_1) \cap \mathsf{Const}^{\mathbf{T}}$ and that J_2 contains the fact $R(\ldots b \ldots)$ where $b = f(a)$ and $b \in dom(J_2) \cap \mathsf{Const}^{\mathbf{T}}$. Assume for the sake of contradiction that $a \nsim b$. Therefore $a \in C_i$ and $b \in C_j$ for some i, j, $1 \leq i, j \leq n$ such that $i \neq j$. Applying $\Sigma_t^c(J_1)$ will generate in J_1' the facts $R(\ldots c \ldots)$ for every constant $c \in C_i$ and $\Sigma_t^c(J_2)$ will generate in J_2' the facts $R(\ldots h(c) \ldots)$ for every constant $h(c) \in C_j$. Since $C_i \cap C_j = \emptyset$, then for each c, $h(c) \neq c$. On the other hand, we have J_1' and J_2' are homomorphically equivalent. This means for each c, $h(c) = c$ and this falsifies our contradiction.

In what follow, we prove that if $b = f(a)$ and $b' = f(a')$ and $b \neq b'$, then $a \neq a'$. Consider that J_2 contains b and b' where $b, b' \in dom(J_2) \cap Const^{\mathbf{T}}$ and $b \neq b'$. Assume for the sake of contradiction that $a = a'$. This means, $a \sim b$ and $a \sim b'$. According to the $\sim$-equivalence relation defined on J_1' by $\hat{\mathcal{M}}$, $b \sim b'$. Let J_2'' be the result of replacing b by b' in J_2. It is clear that J_2'' is a universal DSE solution such that $dom(J_2'') \cap Const^{\mathbf{T}} \subset dom(J_2) \cap Const^{\mathbf{T}}$. This falsifies our contradiction since according to the definition of minimal universal DSE solutions, the target instance J_2'' does not exist.

Finally, we prove that for every $T \in \mathbf{T}$ of arity $k > 0$ and every elements $v_1, \ldots, v_k \in dom(J_1)$, it is the case that $T(v_1, \ldots, v_k) \in J_1 \Leftrightarrow T(f(v_1), \ldots, f(v_k)) \in J_2$. Assume that J_1 contains the fact $T(v_1, \ldots, v_k)$ where for each v_i, $1 \leq i \leq k$, we have v_i is a null or $v_i \in dom(J_1) \cap \mathsf{Const}^{\mathbf{T}}$. From the previous parts, we know that J_2 should contain the fact $T(f(v_1), \ldots, f(v_k))$ such that: if $v_i \in Const^{\mathbf{T}} \Leftrightarrow f(v_i) \in Const^{\mathbf{T}}$ and $v_i \sim f(v_i)$, $1 < i < k$. To complete this proof, we show that if $T(f(v_1), \ldots, f(v_k)) \neq T(f(v_1'), \ldots, f(v_k'))$, then $T(v_1, \ldots, v_k) \neq T(v_1', \ldots, v_k')$. Let J_2 contain the facts $T(f(v_1), \ldots, f(v_k)) \neq T(f(v_1'), \ldots, f(v_k'))$. Assume for the sake of contradiction that $T(v_1, \ldots, v_k) = T(v_1', \ldots, v_k')$, and J_1 contains $T(v_1', \ldots, v_k')$. This means, there exists at least for some i, $1 < i < k$, that $f(v_i) \neq f(v_i')$ and $f(v_i), f(v_i') \in dom(J_2) \cap Var$. An example on this: J_1 contains the fact $T(1, u)$, while J_2 contains the facts $\{T(2, v), T(2, w)\}$ where $1 \sim 2$. Let J_2''' = J_2 - $T(f(v_1'), \ldots, f(v_k'))$. It is easy to see that $J_2''' \subset J_2$ and J_2''' is a universal DSE solution. This falsifies our contradiction since according to the definition of minimal universal DSE solutions, the target instance J_2'''

does not exist.

Proposition 5.4.1

Proof Let $Q(\bar{x}) = \exists \bar{z}\ \phi(\bar{x}, \bar{z}) \wedge \psi(\bar{z})$, be a conjunctive query over $\mathbf{T}$ where: $\bar{x}$ is a set of distinguished variables, $\phi(\bar{x}, \bar{z})$ is a conjunction of predicate formulas that contain distinct variables, and $\psi(\bar{z})$ is a conjunction of equality formulas of the form $z_1 = z_2$ where $z_1, z_2 \in \bar{z}$. Also, let $\hat{Q}(y_1, \ldots, y_n) = Q(x_1, \ldots, x_n) \wedge \bigwedge_{i=1}^{n} \text{EQUAL}(x_i, y_i)$. We are given that J is a MUDSE solution for I and $\mathcal{M}$ under $\mathfrak{S}$.

Assume that $\mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q) = \{t_1, \ldots, t_n\}$. In order to prove that $\mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q) = \hat{Q}(J)$, we first prove that for each tuple $t_i \in \mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, we have t_i in $\hat{Q}(J)$, $1 \leq i \leq n$, and then visa versa.

Case 1: We want to prove that for each tuple $t_i \in \mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, we have $t_i \in \hat{Q}(J)$, $1 \leq i \leq n$. We prove this by showing that $\hat{Q}(J) \subset \mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ is a contradiction. Let J_1 be an arbitrary universal DSE solution. From our definition of $\mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, we know that $Q(\Sigma_t^c(J_1)) = \{t_1, \ldots, t_n\}$. This means, there exists a homomorphism $h : \phi(\bar{x}, \bar{z}) \wedge \psi(\bar{z}) \rightarrow \Sigma_t^c(J_1)$ and $h(\bar{x}) = \{t_1, \ldots, t_n\}$. Let $h(\bar{x}, \bar{z}) = \{t'_1, \ldots, t'_n\}$ where $t_i \subset t'_i$, $1 \leq i \leq n$.

For the sake of contradiction, we assume that there exists a fact t_i for some i, $1 \leq i \leq n$, where $t_i \notin \hat{Q}(J)$. This means, there does not exist a tuple $t \sim t_i$, and $t \in Q(J)$. Let $\{C_1, \ldots, C_m\}$ be the $\sim$-equivalence classes of J. Since J is a MUDSE solution, it is intuitive that each class C_i contains a single witness value w_i, and $w_i \in dom(\mathbf{T}) \cap \mathsf{Const}^{\mathbf{T}}$, $1 \leq i \leq m$. Also, there do not exist in J, two distinct constants w_i and w_j, such that $w_i \sim w_j$, $1 \leq i \leq m$ and $i \neq j$. We define $\{t'_r, \ldots, t'_s\} \subset h(\bar{x}, \bar{z})$ to be the set of tuples such that $t'_l \sim t'_i$, $1 \leq r, s \leq n$ and $r \leq l \leq s$. Let J_2 be the target instance obtained from the universal DSE solution $\Sigma_t^c(J_1)$ by applying the step $replace(\Sigma_t^c(J_1), w_1, \ldots, w_m)$. Also, let $f : \phi(\bar{x}, \bar{z}) \wedge \psi(\bar{z}) \rightarrow J_2$. Then, there should exist in $f(\bar{x}, \bar{z})$ a tuple $t'_q \in \{t'_r, \ldots, t'_s\}$ such that $t'_q \sim t'_i$, $1 \leq r, s \leq n$ and $r \leq q \leq s$. The reason is, if $t'_q[z_1] = t'_q[z_2]$ in $h(\bar{x}, \bar{z})$, then this is still true in $f(\bar{x}, \bar{z})$ since $t'_q[z_1]$ and $t'_q[z_2]$ will replaced by the same witness w_i, $1 \leq i \leq m$. We define a target

instance J' obtained from J_2, by applying procedure $\texttt{CompMUDSEsol}_{\mathfrak{S}}$ on J_2. Now Since J' is the core of J_2, then there exists a homomorphism $k : J_2 \rightarrow J'$. Since, homomorphism is closed under concatenation, then there exist a homomorphism $f \circ k : \phi(\bar{x}, \bar{z}) \wedge \psi(\bar{z}) \rightarrow J'$, and $f \circ k(t'_q) \in f \circ k(\bar{x}, \bar{z})$. Since certain answers are tuples of constants, then $t'_q \in f \circ k(\bar{x}, \bar{z})$, and $t_q \in f \circ k(\bar{x})$ (i.e $t_q \in q(J')$). Both J' and J are universal MUDSE solutions, $dom(J) \cap \mathsf{Const}^{\mathbf{T}}$ = $dom(J') \cap \mathsf{Const}^{\mathbf{T}}$, and there does not exist two constants w_1 and w_2 in $dom(J') \cap \mathsf{Const}^{\mathbf{T}}$ such that $w_1 \sim w_2$, then following Theorem 5.3.2, it is intuitive that J and J' are homomorphically equivalent. Let u be a homomorphism $u : J' \rightarrow J$. Then there exist a homomorphism $f \circ k \circ u : \phi(\bar{x}, \bar{z}) \wedge \psi(\bar{z}) \rightarrow J$, and $f \circ k \circ u(t'_q) \in f \circ k \circ u(\bar{x}, \bar{z})$. This leads to $t_q \in f \circ k \circ u(\bar{x})$ (i.e $t_q \in Q(J)$). It is given before that $t_q \sim t_i$, and this falsifies what we assumed earlier that there exists a tuple $t \sim t_i$, and $t \notin \hat{Q}(J)$.

Case 2: We want to prove that for each tuple $t_i \in \hat{Q}(J)$, t_i in $\mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, $1 \leq i \leq n$. Assume for the sake of contradiction that there exists a tuple $t_i \in \hat{Q}(J)$, and $t_i \notin \mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$. Let J_1 be an arbitrary universal DSE solution. Then, from our definition of $\mathsf{certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$, we can deduce that $Q(\Sigma_t^c(J_1)) = \{t_1, \ldots, t_n\}$. Let g be a homomorphism $g : \phi(\bar{x}, \bar{z}) \wedge \psi(\bar{z}) \rightarrow \Sigma_t^c(J_1)$, $g(\bar{x}, \bar{z}) = \{t'_1, \ldots, t'_n\}$, and $g(\bar{x}) = \{t_1, \ldots, t_n\}$. Let h be a homomorphism $h : \phi(\bar{x}, \bar{z}) \wedge \psi(\bar{z}) \rightarrow J$, and $h(\bar{x}, \bar{z}) = \{t'_j, \ldots, t'_k\}$. Let J^* = $\Sigma_t^c(J)$. Then, from the definition of a DSE solution, we deduce that J^* is a universal DSE solution. Let f be a homomorphism $f : \phi(\bar{x}, \bar{z}) \wedge \psi(\bar{z}) \rightarrow J^*$, then $f(\bar{x}, \bar{z}) = \{t'_1, \ldots, t'_n\}$, $f(\bar{x}) = \{t_1, \ldots, t_n\}$, and $\{t'_j, \ldots, t'_k\} \subseteq \{t'_1, \ldots, t'_n\}$ since $J \subseteq J^*$.

Let K be the result of completing $h(\bar{x}, \bar{z})$, as $h(\bar{x}, \bar{z}) \wedge \textsc{Equal}(\bar{x}, \bar{y})$, using $\hat{\mathcal{M}}$. Then, K will contain for each $t \in \{t'_j, \ldots, t'_k\}$, all the possible tuples t' such that $t \sim t'$, $t'[\bar{x}] \sim t[\bar{x}]$, and $t'[\bar{z}] \sim t[\bar{z}]$, (we know that $t'[\bar{z}] \sim t[\bar{z}]$ includes $t'[\bar{z}] = t[\bar{z}]$). Also, from the definition of a universal DSE solution, we know that for each fact $R(t) \in J^*$, we have all the possible facts $R(t')$ such that $R(t) \sim R(t')$ using $\hat{\mathcal{M}}$. Therefore, we can easily deduce that $f(\bar{x}, \bar{z}) = \{t'_j, \ldots, t'_k\} \cup \{t''\}$ for each $t'' \sim t'_l$, $j \leq l \leq k$, and $J^* \vDash \phi(t''[\bar{x}], t''[\bar{z}]) \wedge \psi(t''[\bar{z}])$. Thus, we can say that, for each tuple $t \in K$, we have a tuple $t' \in f(\bar{x}, \bar{z})$ such that $t[\bar{x}] = t'[\bar{x}]$. From our definition of a universal DSE solution, we can deduce that $\Sigma_t^c(J^*) = J^*$ and that J^*

and $\Sigma_t^{\mathrm{c}}(J_1)$ are homomorphically equivalent. Thus, there exists a homomorphism $e : J^* \rightarrow \Sigma_t^{\mathrm{c}}(J_1)$, and $e(f(\bar{x}, \bar{z})) \in g(\bar{x}, \bar{z})$. Consequently, $e(f(\bar{x})) \in g(\bar{x})$. Since certain answers have facts with only constants and homomorphisms are identity on constants, then there does not exist a fact $t_i \in \{t_1, \ldots, t_n\}$ such that $t_i \notin g(\bar{x})$. This is equivalent to saying that each fact $t_i \in \{t_1, \ldots, t_n\} \in Q(\Sigma_t^{\mathrm{c}}(J_1))$, and this falsifies our contradiction.

Proposition 5.4.2

Proof Let Q be a fixed union of conjunctive queries. The fact that computing the set of sound and complete certain answers of Q, $\mathsf{sound\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ and $\mathsf{complete\text{-}certain}_{\mathfrak{S}}((I \cup \{\mathcal{M}\}), Q)$ respectively, is in LOGSPACE, is based on the following facts: (1) following Proposition 5.2.5, generating a universal DSE solution is in LOGSPACE; and (2) following Theorem 5.3.2, generating a MUDSE solution is in LOGSPACE; and finally (3) it is known from [67] that the data complexity of any FO formula is in LOGSPACE, and thus checking if a fixed conjunctive query is satisfied in a database instance is in LOGSPACE.

Theorem 6.3.1

Proof Assume that the target instance J contains a naïve table R and that J' contains a naïve table R_c. Since J' is the core of J [10], then according to [10], $\mathsf{Rep}(R) = \mathsf{Rep}(R_c)$ and there exists a homomorphism from R_c to R.

Assume for the sake of contradiction that there exists a target instance J'' such that J'' contain a pc-table S, $\mathsf{Rep}(J) = \mathsf{Rep}(J'')$, and $|J''| < |J'|$. Let $\mathsf{Const}^{\mathbf{S}}$ be the set of constants in S and $\mathsf{Var}^{\mathbf{S}}$ be the set of variables in S.

Now, let us consider the valuation ρ of S such that each null value $\perp \in \mathsf{Var}^{\mathbf{S}}$ is such that $\rho(\perp) = c$ where c is a fresh constant and $c \notin \mathsf{Const}^{\mathbf{S}}$. In this case, for each tuple $\langle t : C \rangle \in S$, then $\rho(C) = false$. Consequently, each tuple $\langle t' : C' \rangle \in S$ such that $\rho(t') \in \rho(S)$, it is that the condition C' of t' is a tautology; that is C' is a rule that for every valuation $\rho'(S)$, $\rho'(C')$ always evaluates to $true$. Given this fact, generating from S a pc-table S' such that for each

tuple $\langle t'' : C'' \rangle \in S$ and condition C'' that has the property of C', $\langle t'' : true \rangle$ is placed in S', and for all the remaining tuples $\langle t : C \rangle \in S$, then $\langle t : C \rangle$ also exist in S',we have $S \equiv S'$.

Given the fact that $S \equiv R_c$ and $S \equiv S'$, then we can deduce that $S' \equiv R_c$. Also, since R_c is the core of R then according to the definition of core solutions in [10], there does not exist an instance R'' with naïve tables where $|R''| < |R_c|$ and $R'' \equiv R$. Then, the smallest instances that belong to $\mathsf{Rep}(R_c)$ have the same number of tuples, n, of R_c.

Now if $|S'| < |R_c|$, then applying a valuation ρ' to S' such that each condition C – that is not a tautology – is such that $\rho'(C) = false$ will generate an instance $\rho'(S')$ where $|\rho'(S')| < n$. And this fact falsifies our contradiction that S exists, since $\rho'(S') \notin \mathsf{Rep}(R_c)$.

Theorem 6.3.2

Proof Following Theorem 6.2.3, there exists a polynomial time algorithm that generates a universal DSE KB-solution in a DSE$_I$ setting. Assume that J' is a generated universal DSE KB-solution for I and $\mathcal{M}$ under $\mathfrak{S}$. To check whether J is a universal DSE KB-solution for I and $\mathcal{M}$ under $\mathfrak{S}$, it is enough to prove that $J \equiv J'$.

To prove that $J \equiv J'$, we should show that $\mathsf{Rep}(J) = \mathsf{Rep}(J')$. We show in what follow that checking whether two pc-tables T_1 and T_2 are equivalent is $DP_2^p - complete$. With this result, we can deduce that the problem of checking whether J is a universal DSE KB-solution for I and $\mathcal{M}$ under $\mathfrak{S}$ is DP_2^p-complete.

Let Var_1 be the set of variables in T_1 and Var_2 be the set of variables in T_2. Also, let Const_1 be the set of constants in T_1 and Const_2 be the set of constants in T_2. Let C_1 be a set of constants such that C_1 is disjoint from Const_1 and $|C_1| = |\mathsf{Var}_1|$. Also, let C_2 be a set of constants such that C_2 is disjoint from Const_2 and $|C_2| = |\mathsf{Var}_2|$. We define the $dom(T_1) = \mathsf{Const}_1 \cup C_1$ and $dom(T_2) = \mathsf{Const}_2 \cup C_2$. We define $\mathsf{Rep}(T_1)$ to be all the possible valuations of T_1 using constants in $dom(T_1)$ and $\mathsf{Rep}(T_2)$ to be all the possible valuations of T_2 using constants in $dom(T_2)$.

To prove that $\mathsf{Rep}(T_1) = \mathsf{Rep}(T_2)$, then for each valuation $\rho(T_1)$ using constants in dom_1, we need to guess a valuation $\rho'(T_2)$ using constants in dom_2 such that $\rho(T_1) = \rho'(T_2)$. Also,

for each valuation $\rho_2(T_2)$ using constants in dom_2, we need to guess a valuation $\rho_1(T_1)$ using constants in dom_1 such that $\rho_2(T_2) = \rho_1(T_1)$. Therefore, for each valuation $\rho(T_1)$, we need to call an NP Oracle to guess a valuation $\rho'(T_2)$ such that $\rho(T_1) = \rho'(T_2)$. In addition, same case is applied for $\rho_1(T_1)$ and $\rho_2(T_2)$. The total number of different valuations $\rho(T_1) = |dom_1|^{|\mathsf{Var}_1|}$ and the total number of different valuations $\rho'(T_2) = |dom_2|^{|\mathsf{Var}_2|}$.

We now prove that checking if $T_1 \equiv T_2$ is DP_2^p-hard by reducing $\forall * \exists * 3SAT - \forall * \exists * UN - 3SAT$ which is DP_2^p-complete. $\forall * \exists * 3SAT - \forall * \exists * UN - 3SAT$ problem is as follows: given two sentences $\phi = \forall X \exists Y \psi(X, Y)$ and $\theta = \exists U \forall V \delta(U, V)$, the problem is to decide whether ϕ and θ are true. We say that $T_1 \equiv T_2$ only iff ϕ and θ are true.We use ϕ and θ to construct the conditional tables T_1 and T_2. Given $\phi = \forall X \exists Y C_1(X, Y) \wedge ... \wedge C_n(X, Y)$ and $\theta = \exists U \forall V C'_1(U, V), ..., C'_m(U, V)$, we first define S_ϕ to be the set of possible formulas over $C_1 \wedge ... \wedge C_n$ such that all possible negations of the elements in Y are considered. So, for example, assume that $\psi(X, Y) = C_1(x_1 \vee x_2 \vee y_1) \wedge C_2(\neg x_1 \vee \neg y_1 \vee y_2)$. Then, the set $C_1(X, Y_l) \wedge ... \wedge C_n(X, Y_l)$ = $\{C_1(x_1 \vee x_2 \vee y_1) \wedge C_2(\neg x_1 \vee \neg y_1 \vee \neg y_2), C_1(x_1 \vee x_2 \vee y_1) \wedge C_2(\neg x_1 \vee y_1 \vee y_2), C_1(x_1 \vee x_2 \vee y_1) \wedge C_2(\neg x_1 \vee y_1 \vee \neg y_2), C_1(x_1 \vee x_2 \vee \neg y_1) \wedge C_2(\neg x_1 \vee y_1 \vee \neg y_2), C_1(x_1 \vee x_2 \vee \neg y_1) \wedge C_2(\neg x_1 \vee y_1 \vee y_2), C_1(x_1 \vee x_2 \vee \neg y_1) \wedge C_2(\neg x_1 \vee y_1 \vee \neg y_2), C_1(x_1 \vee x_2 \vee \neg y_1) \wedge C_2(\neg x_1 \vee \neg y_1 \vee \neg y_2)\}$. Also, the same concept applies to the set S_ψ with respect to the set U.

Assume, without loss of generality, that $S_\phi = \{F_1, ..., F_r\}$, and that $S_\psi = \{F'_1, ..., F's\}$.

Now, we construct T_1 to be $T_1 = \{(0, 1, 0 : F_1), ..., (0, 1, 0 : F_r), (1, 1, 1 : \neg(\psi \bigvee_{i=1}^{i=s} F'_i))\}$
Also, we construct T_2 to be $T_2 = \{(0, 1, 0 : F'_1), ..., (0, 1, 0 : F'_s), (1, 0, 0 : \neg(\phi \bigvee_{i=1}^{i=r} F_i))\}$

We say, $T_1 \equiv T_2$ only if both ϕ and θ are true. We below show that our reduction is correct:
$\rightarrow$ Assume that both ϕ and θ are true. Then, for any assignment, ρ_1, of the variables $X \cup Y$ in ϕ, $\rho_1(X \cup Y)$ = True, and for any assignment of the variables $U \cup V$, ρ_2, in θ, $\rho_2(U \cup V)$ = True. Also, we have $\rho_1(T_1) = \{(0, 1, 0)\}$ and $\rho_2(T_2) = \{(0, 1, 0)\}$. Therefore, we can say that $\mathsf{Rep}(T_1) = \mathsf{Rep}(T_1)$.

$\rightarrow$ Assume that either ϕ or θ is false. This can occur in the following three cases:

1. ϕ is true and θ is false: In this case, we have for any assignment of the variables $X \cup Y$ in ϕ, $\rho_1(X \cup Y)$ = True, and for any assignment of the variables $U \cup V$ in θ, $\rho_2(U \cup V)$ = False. Therefore, we will have $\rho_1(T_1) = \{(0,1,0),(1,1,1)\}$ and $\rho_2(T_2) = \{\emptyset\}$. Therefore, $\mathsf{Rep}(T_1) \neq \mathsf{Rep}(T_2)$.

2. ϕ is false and θ is true: In this case, we have for any assignment of the variables $X \cup Y$ in ϕ, $\rho_1(X \cup Y)$ = False, and for any assignment of the variables $U \cup V$ in θ, $\rho_2(U \cup V)$ = True. Therefore, we will have $\rho_1(T_1) = \{\emptyset\}$ and $\rho_2(T_2) = \{(0,1,0),(1,0,0)\}$. Therefore, $\mathsf{Rep}(T_1) \neq \mathsf{Rep}(T_2)$.

3. ϕ is false and θ is false: In this case, we have for any assignment of the variables $X \cup Y$ in ϕ, $\rho_1(X \cup Y)$ = False, and for any assignment of the variables $U \cup V$ in θ, $\rho_2(U \cup V)$ = False. Therefore, we will have $\rho_1(T_1) = \{(1,1,1)\}$ and $\rho_2(T_2) = \{(1,0,0)\}$. Therefore, $\mathsf{Rep}(T_1) \neq \mathsf{Rep}(T_2)$.

Theorem 6.3.4

Proof Given a universal DSE KB-solution (J, Σ_c^t). Assume without loss of generosity that J consists of n tuples. If there exists a DSE KB-solution (J', Σ_c^t) for I and $\mathcal{M}$ under $\mathfrak{S}$ such that $J' \subset J$, then J' should have the following property: for each solution $J_1 \in \mathsf{Rep}(J')$, their exists a solution $J_2 \in \mathsf{Rep}(J)$ such that $J_2 \subseteq J_1$.

To check if J' exists, it is enough to perform n parallel calls to a procedure that verifies whether there exists an instance $J_i = J - \langle t_i, c_i \rangle$, $1 \leq i \leq n$, where $\langle t_i, c_i \rangle$ is a tuple with a condition in J, such that J_i possesses the same properties of J' mentioned above. That is: for each solution $J_1 \in \mathsf{Rep}(J_i)$, their exists a solution $J_2 \in \mathsf{Rep}(J)$ such that $J_2 \subseteq J_1$.

In order to prove this Theorem, we leverage the check-solution problem for pc-tables in data exchange settings that was proved in [11] to be $\Pi_2^P - hard$. The check-solution problem for pc-tables in data exchange settings is as follows: Given an pc-table instance M over a schema $\mathbf{S}_1$, a pc-table instance N over a schema S_2, and a set of st-tgds Σ_{st}^1. Check whether N is a solution of M, in other words, check whether $(M, N) \vDash \Sigma_{st}^1$.

Now we reduce the check-solution problem for pc-tables to each problem in the n parallel procedures as follows: First we define Σ_{st}^2 to contain mapping st-tgds of the form:

$$\forall \bar{x} R(\bar{x}) \rightarrow \forall \bar{x} R(\bar{x})$$

for each relation R in J. Then, for each instance J_i, where $J_i = J - \langle t_i, c_i \rangle$ and $1 \leq i \leq n$, J_i is a DSE KB-solution under $\mathfrak{S}$ if and only if $(\Sigma_c^t(J_i), \Sigma_c^t(J)) \vDash \Sigma_{st}^2$.