

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT COMPUTERWETENSCHAPPEN
AFDELING INFORMATICA
Celestijnenlaan 200 A — B-3001 Leuven

TOWARDS A LOGICAL RECONSTRUCTION OF A THEORY FOR LOCALLY COMPLETE DATABASES

Promotoren:
Prof. Dr. Marc Denecker
Prof. Dr. ir. Maurice Bruynooghe

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Álvaro Cortés Calabuig

December 2008



KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT COMPUTERWETENSCHAPPEN
AFDELING INFORMATICA
Celestijnenlaan 200 A — B-3001 Leuven

TOWARDS A LOGICAL RECONSTRUCTION OF A THEORY FOR LOCALLY COMPLETE DATABASES

Jury:
Prof. Dr. ir. Hugo Hens, voorzitter
Prof. Dr. Marc Denecker, promotor
Prof. Dr. ir. Maurice Bruynooghe, promotor
Prof. Dr. Daniel de Schreye
Prof. Dr. Bettina Berendt
Prof. Dr. Ofer Arieli (Academic College of Tel Aviv)
Prof. Dr. Jan Chomicki (University at Buffalo)

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Álvaro Cortés Calabuig

U.D.C. 681.3*I23

December 2008

©Katholieke Universiteit Leuven – Faculteit Ingenieurswetenschappen
Arenbergkasteel, B-3001 Heverlee (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotocopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the publisher.

D/2008/7515/123
ISBN 978-94-6018-015-6

Abstract

A relational database stores information about the world. In order to describe how reliable the content of a database is with respect to a certain domain, two cardinal concepts are generally used: correctness and completeness. Correctness refers to whether the database stores only true facts about its intended domain or if it possibly contains false data. Completeness refers to whether the database stores all the information that is true in the world under consideration or not.

In this work we study databases that store correct but possibly incomplete information about the world. In particular, we investigate what we call locally complete databases. These are databases that may be complete about certain parts of the world, while being incomplete in general. Such databases consist of a standard relational database supplemented with a number of specifications, called local closed world assumptions, of the areas in the real world where the database contains all true tuples.

Our study of locally complete databases is divided into three parts. In the first part we investigate how to represent the fact that a relational database is locally complete. For this purpose we extend the idea of a window of expertise of a database predicate to the first-order case. At the reasoning level, we study two important reasoning tasks: computing answers to queries and determining whether a database has complete knowledge on a query. In the second part of the thesis we study methods to perform efficient computation of the reasoning tasks described above. In particular, we present approximate methods for query answering. We prove that for a useful class of queries and locally complete databases these methods are tractable and complete. The third and last part of the thesis deals with extending the query language from standard SQL to queries with modalities. This extension will allow us to retrieve useful information about the knowledge endorsed by a locally complete database. In this part, we also present a prototype of this extension, based on the model expansion system IDP.

Acknowledgements

This work would not have been finished without the support of many people, for many years and at many levels. These pages are now my opportunity to say thank you to those people.

The most important person in this work has been my co-supervisor Marc Denecker. His support during all these years has been absolutely amazing. I can not remember a single time Marc refused to attend to me when I knocked on his door seeking his scientific (and sometimes non-scientific) advice. Very often after those meetings I felt a motivational boost about my research, with the feeling of having had a lesson on clarity of thinking and precision. This thesis has certainly very much benefitted from his ideas and technical involvement. Sharing extra-office time with him (for instance, trekking in the Patagonian Andes) has been great fun as well. Thanks, Marc.

My co-supervisor Maurice Bruynooghe was the person I first contacted to apply for a PhD in Leuven. Since that very first email long ago, the support of Maurice, ranging from the financial to the scientific order, has been continuous, effective and, very often, silent. Thanks, Maurice.

I'd like to thank Ofer Arieli for many years of fruitful scientific collaboration. I'd also like to thank him and his family for their hospitality during my visit to Tel Aviv in 2005.

I am honored to have a world-class group of scientists as members of my PhD jury. Besides my supervisors, I'd like to thank Danny de Schreye, Bettina Berendt, Ofer Arieli, Jan Chomicki and jury chair Hugo Hens for kindly agreeing to be members of my doctoral commission. Their detailed comments on the preliminary draft of the thesis turned out to be very valuable. I am specially grateful to Bettina Berent, who accepted to be part of the committee (and also the secretary!) on a short notice before the preliminary defense.

During the first four years of my stay in Leuven I've been funded by a scholarship of the international office of the K.U. Leuven. I'd like to thank the university for this monetary support. I also thank GOA on Inductive Knowledge Bases for additional financial support during the last four years.

For all his diligence with the administration of the scholarship and various resident permits, thanks go to Edmund Guzmán. For similar reasons, I'd also like to express gratitude to Karin Michiels and Denise Brahms.

Maarten Mariën has been my partner in adventures right from the beginning of this PhD. I'd like to thank him for so many hours of fun travelling, philosophical arguments, photography trips, running events, etc, etc, etc. Perhaps most important of all, I'd like to thank him just for being a good friend.

During these years there were also difficulties. In the troubled days two persons made a big different for me: Anissa Goffart and Anneleen van Assche. They provided me with a comforting light in the middle of those otherwise dark days. Thanks a lot, ladies.

I'd like to thank Fabian Güiza for connecting me with my Latin American roots. If so far I am still able to speak my native language it is because of our frequent and always amenable chats. Para que te digo que no si sí.

Many thanks go to Joost Vennekens and Stephen Bond for countless relaxing evenings of DVD watching sessions, jazz attendance, beer, and the unutterable hobby ;).

I'd also like to say thanks to other members of the KR&R group for numerous pleasant coffee breaks together: Ping Hou, Johan Wittocx, Hanne Vlaeminck and Mai Ajspur. I am also grateful to ML members Daan Fierens and Leander Schietgat for their sportive companionship in, respectively, climbing and tennis events.

During the past five years many other people have made a positive impact in my life. I would like to express my deep gratitude to them: Veerle Coupé, Monica Caniupán, Alex Gurovich, An Buys, Loreto Bravo, Marcello di Bello, Brechtje Vandergraesen, Jorge González, Lien Van Besien, Fien Decaluwe, Hilla Karas, Eva, Leopoldo Bertossi, and the Roggen Family. Thank you all.

Even from 16.000km away I've felt, as if they were all right here, the constant support of my family members. I'd like to thank my parents María Teresa and Manuel, and Joaquín, Gonzalo, Francisca and Isabel for a lifetime of care and support, and for cheering me up from the distance all these years.

In the fourth year of my PhD I met someone who since then has irradiated my days with happiness, 'surprise' and joy. She is my girlfriend Lien Roggen, and just for being who she is my biggest thanks go to her.

Álvaro
December, 2008

I have approximate answers, and possible beliefs, and different degrees of certainty about different things, but I'm not absolutely sure of anything, and in many things I don't know anything about, such as whether it means anything to ask why we're here, and what the question might mean.

Richard Feynman (1918-1988),
The pleasure of finding things out.

Contents

Abstract **i**

Acknowledgements **ii**

Contents **vii**

1 Introduction **1**
1.1 Context 1
1.1.1 The Closed-World Assumption (CWA) 3
1.1.2 The Case of Incomplete Databases 5
1.1.3 The Local Closed-World Assumption 9
1.2 Goals and Contributions of this Thesis 10
1.2.1 Methodology 12
1.3 Organization of the Text 15
1.3.1 Dependency of Chapters 16
1.4 Bibliographical Note 16

2 Technical Background **17**
2.1 Basics of Set Theory 17
2.2 Enumerability 18
2.3 Order and Lattice Theory 18
2.3.1 Lattices as Ordered Sets 19
2.4 Computational Complexity 20
2.5 Classic Logical Formalisms 22
2.5.1 Propositional Logic 22
2.5.2 First-order Logic 24
2.6 Nonmonotonic Logical Formalisms 28
2.6.1 Circumscription 29
2.6.2 Logic Programming 31
2.6.3 ID-Logic 33
2.7 Bibliographical Note 35

3 The Local Closed-World Assumption **37**
3.1 Introduction . . . 37
3.2 Relational Databases . . . 38
3.3 Local Closed-World Assumptions . . . 41
3.3.1 Locally Closed Databases . . . 46
3.3.2 Relations to CWA and OWA . . . 49
3.4 Further Representation Considerations Regarding the LCWA . 50
3.4.1 LCWA with Multiple Objects . . . 50
3.4.2 Expressing Complete Knowledge on an Object . . . 51
3.4.3 LCWA with Several Databases . . . 52
3.4.4 Boolean Combinations of LCWAs . . . 52
3.4.5 LCWA and the Soundness Assumption . . . 53
3.4.6 A Circumscriptive Approach to the LCWA . . . 54
3.5 Queries in Locally Closed Databases . . . 56
3.5.1 Query Answering and Complete Information . . . 58
3.5.2 Complexity Results . . . 61
3.5.3 Summary of the Results . . . 65
3.6 Related Work . . . 66
3.6.1 Local Forms of CWA in Databases . . . 66
3.6.2 Deductive Databases and Logic Programming . . . 74
3.7 Related Topics . . . 75
3.7.1 Safety . . . 75
3.7.2 An Application of the LCWA to Mediator-Based Systems 76
3.8 Bibliographical Note . . . 79

4 Approximative Reasoning for the LCWA **81**
4.1 Introduction . . . 81
4.2 Approximative Reasoning . . . 82
4.2.1 Three-Valued Structures . . . 83
4.2.2 Approximations by Three-Valued Structures . . . 88
4.3 Constructing Approximations by Fixpoint Computations . . . 90
4.3.1 Simulating Approximate Query Answering by Fixpoint Queries . . . 93
4.3.2 The logic LFP^{simult} . . . 94
4.3.3 Query Approximation by Expressions in LFP^{simult} . . . 96
4.3.4 Hierarchically Closed Databases . . . 101
4.3.5 Towards Approximate Query Answering with Integrity Constraints . . . 105
4.3.6 Summary of the Results on Approximations . . . 107
4.4 Accuracy of Approximate Query Answering . . . 107
4.4.1 Introduction . . . 107
4.4.2 Motivational Examples . . . 108
4.4.3 Modularity Properties of Locally Closed Databases . . . 110
4.4.4 Squared Queries . . . 112

4.4.5 Proving Squaredness of Queries 114
4.4.6 Proving Partial Optimality of $\mathcal{C}_{\mathfrak{D}}$: 116
4.4.7 Summary of the Results on Accuracy of Approximations 117
4.5 Domain-Independent and Safe Queries 119
4.6 Conclusions 120
4.7 Proofs for the Accuracy Section 122

5 Modal Queries and a Prototype for the LCWA 127
5.1 Introduction 127
5.2 A Modal Language for the LCWA 130
5.2.1 The Logic of $KFOPCE$ 130
5.2.2 $KFOPCE$ and Three-Valued Logic 134
5.2.3 $KFOPCE$ and Locally Complete Databases 134
5.2.4 Discussion on $KFOPCE$ Queries 137
5.3 An IDP-Based Prototype of the Local Closed-World Assumption 138
5.3.1 System Overview 138
5.4 Summary of the Results 142
5.5 Bibliographical Note 143

6 Conclusions 145

A Apendix 149
A.1 Perl Script Output File 149
A.2 IDP Execution Output File 150

References 153

List of Symbols 161

Index 164

Publication List 165

Biography 167

Chapter 1

Introduction

1.1 Context

A relational database stores information about the world. As with other media or sources of information, the accuracy of the data will determine whether or not we can trust the database to retrieve our knowledge about the world. In order to describe how reliable the content of the database is with respect to a certain domain, two cardinal concepts are used: correctness and completeness[1].

Correctness refers to whether the database stores *solely* true facts about its intended domain or if it possibly contains false data. If the former is the case the database is branded as *correct*, and we can be confident that all the information in the repository is true in the real world.

Completeness points out whether the database stores *all* the information that is true in the world under consideration or whether there is possibly missing information. A database is labeled as *complete* if it belongs to the first case, and *incomplete* otherwise.

In a database that is incorrect or incomplete (or both), there exists a mismatch between the intended reality to be captured by the database and the database itself. In an ideal world there is no mismatch, and the database remains correct and complete at all time points. In such case we can trust that all queries will be answered with exact information.

Unfortunately, as the following example illustrates, all too often we face situations in which the database is incomplete, incorrect, or both.

Example 1.1 *Consider a database storing information about trains and their departure times from the train station of the city of Leuven. The following table*

[1] A notion somehow related to these two concepts is *consistency*. In this thesis we do not address this issue. The reader interested in the matter is referred to (Subrahmanian 1992; Bertossi 2006; Arieli et al. 2007) for different treatments of consistency in relational databases.

is an extract of such a database for trains that leave the station between 8 and 9 o'clock in the morning. Suppose we are informed by the personnel at the station

Train Time Table	
Destination	*Time*
Brussel Centraal	8:03
Antwerpen Centraal	8:05
Gent St. Pieters	8:13
Brugge	8:22

Table 1.1: A train schedule database.

that due to a technical failure the train with destination Antwerpen Centraal has been cancelled. In its place, a new train with destination Mechelen and connection to Antwerpen Centraal has been scheduled at 8:34. The database, however, has not immediately been updated with this information. As a consequence the database becomes incorrect, since it stores a train to Antwerpen Centraal at 8:05 that is not actually stopping at the station. The database is not complete anymore either, because it does not register the data of the newly scheduled train to Mechelen. Observe, however, that travellers with destination Antwerpen Centraal that rely on the information in the official database and that are not informed of the new state of affairs on time risk waiting in vain and would likely loose the train to Mechelen and the additional connection to Antwerpen Centraal. Travellers with destinations other than Antwerpen Centraal would not be affected by the lack of accuracy of the database, since the faulty information is circumscribed to that station. In order to reflect the actual state of affairs, the database should have been updated as follows:

Train Time Table	
Destination	*Time*
Brussel Centraal	8:03
Gent St. Pieters	8:13
Brugge	8:22
Mechelen (Connect. Ant)	8:34

Table 1.2: Updated train schedule database.

In this updated table the erroneous tuple with information to Antwerpen Centraal has been deleted and an additional tuple with data regarding the train to Mechelen has been added.

At a given time point, a relational database table is in one of the following accuracy states: (a) Correct and complete, or *accurate* (b) Complete but incorrect

(c) Correct but incomplete or (d) Incorrect and incomplete.
In the following we briefly describe each accuracy state.

(a) **The database is correct and complete, or accurate**. In this state the database is an *exact* abstract image of the external world. All the information that has to be in the database, is included. Moreover, the database does not contain any information that is false in the domain of discourse. Queries posed to such database are always answered with exact information about its intended world.

(b) **The database is complete but incorrect**. The database contains all the information in the intended domain but it possibly contains some additional tuples storing unsound data. Information retrieved from such database is always complete but also possibly incorrect.

(c) **The database is correct but incomplete**. This state corresponds to the dual of case (b). This is the status of a database that stores only part of the information in the intended domain. This part, however, is always sound.

(d) **The database is both incorrect and incomplete**. In this state a database stores only part of the intended domain plus additional tuples with false information. Answers retrieved from such databases can be incomplete, incorrect, or both. This is the case of the database of Example 1.1 before the update is performed.

A relational database presents itself as a set of tuples arranged in a number of tables. Without additional (external) knowledge, the user can not determine solely from this information in which of the four states (a)-(d) the database is. As a consequence, the user has to make a certain *assumption* in order to give an interpretation of the tuples in the database and of the answers that are retrieved from it. The assumption, if sensible, has to be adopted based on additional *meta knowledge* about the contents of the database. This knowledge can be acquired in different ways. For instance, by the user based on his/her/its own experience interacting with the database. Another way is by relying on the experience of other users of the database, or on information provided by an expert on the database. Irrespective of the grounds in which the assumption is made, it will determine what is true and false according to the database.

In this thesis we will concentrate on assumptions for databases that are believed to be in states (a) or (c) of the diagram above, i.e., databases that contain true but possibly incomplete information about the world. The well-known closed-world assumption is one of such assumptions.

1.1.1 The Closed-World Assumption (CWA)

The closed-world assumption was introduced by Raymond Reiter (Reiter 1978) to give an account of relational databases that are considered as completely

representing all the information about a certain world. In its paradigmatic form the closed-world assumption presupposes the following:

- All tuples in the database are *true* in the real world.
- Every tuple not in the database is assumed to be *false*.
- Different names in the database represent different objects in the real world.
- All objects in the domain are represented in the database.

Databases coupled with the closed-world assumption are called *closed-world databases*. In such databases there is no uncertainty about the domain, and every answer to a closed query[2] is either true or false. The following example illustrates what is believed according to a database in which this assumption is made.

Example 1.2 *Consider again the updated database of Table 1.2. Suppose the closed-world assumption is adopted and the following queries are posed to it:*

- *Query 1: Is there a train to Antwerpen Centraal before 8:15?*
  Answer: No.
- *Query 2: Is there any train with a connection to Antwerpen Centraal before 9 o'clock?*
  Answer: Yes.
- *Query 3: Retrieve the departure times of all trains stopping at the train station of Leuven between 8 and 9 o'clock.*
  Answer: {8:03,8:13,8:22,8:34}.

The answers in this example are all correct and complete in the sense that they correspond exactly to the situation at the Leuven train station between 8 and 9 o'clock. As the following example shows, wrong conclusions can be derived when the closed-world assumption is adopted on databases that are not complete or correct.

Example 1.3 *Lirias is a newly implemented centralized institutional repository of scientific publications of the K.U. Leuven. The central database is structured into communities and collections, representing groups, departments and divisions. The data in Lirias is automatically imported from local data sources of research centers and faculties of the university. The migration is currently carried out in stages, and every new record has to be validated by at least one of the authors of each publication. The following table presents the results of a query about publications of people with last name 'Cortés'*[3]*.*

[2] A query that contains no free variables.

[3] Please forgive the self reference: Cortés was the only person the author of this thesis had *complete knowledge* about his publications.

An inspection of the database of the computer science department (the original source of data of Lirias for the author 'Cortés'), reveals that these results are highly incomplete: some publications for the year 2006 are not in display, and all publications for 2007 and 2008 are missing. For years earlier than 2006 the database shows complete results.

The Lirias database

Date	*Title*	*Contributors*
2004	Data integration using ID-Logic	Van Nuffelen et al.
2005	On the LCW assumption of data sources	Cortés et al.
Aug-2006	Partial knowledge in LC databases	Cortés et al.

Table 1.3: The Lirias database.

Consider the following queries and their answers if the CWA is adopted in the database of Table 1.3:

- *Query 1: Retrieve all publications of 'Cortés' in year 2006.*
  Answer 1: 'Partial knowledge in LC databases'.
- *Query 2: Are there publications of 'Cortés' in year 2007?*
  Answer 2: No.
- *Query 3: Are there publication of 'Cortés' in year 2008?*
  Answer 3: No.

Answers to queries 1 and 2 are both wrong. Answer 1 is incorrect since there exist other publications for 2006 than the one in the answer. Query 2 is also incorrect because there is at least one publication in 2007 in which 'Cortés' is a co-author. Finally, Query 3 is the only correct one; there are no publications of this author in 2008 yet.

This example shows how the closed-world assumption fails to capture the fact that there is missing information about the domain, and that as a consequence incorrect answers can be retrieved. For databases that are incomplete it is therefore necessary to adopt another assumption, one that gives a different treatment to tuples that are not stored in the database. In the following section we discuss such an assumption, the *open*-world assumption, in connection with databases that are believed to be incomplete.

1.1.2 The Case of Incomplete Databases

Incompleteness is a very common phenomenon in relational databases. In broad terms, it refers to information that one normally expects to be present in the database. As observed by Levesque (Levesque 1982), saying that a database

is incomplete is rather different from the obvious fact that a database cannot capture all the aspects of some world and therefore unknown information will always exist. Reasons leading to incompleteness are broad and varied. In what follows we briefly comment on a few of them.

- **Ignorance about the domain.** This is perhaps the most frequent source of database incompleteness. It refers to the fact that information about the domain itself is unknown and therefore there is no data to fulfill the intended content of the database. Paradigmatic cases of databases in this situation are data repositories in the bio-informatics domain. In this area of research it is often the case that the results of experiments are incrementally added to general data banks. Until *all* the information is obtained from the experiments – something unfeasible in many cases – the repository remains incomplete. In the meantime the database remains incomplete, and its information is queried and shared with other repositories of data.

- **Lack of proper maintenance.** In this case, instead of lacking knowledge about the world, the lack is in human resources, machine power or just interest to update the database. Incompleteness due to poor maintenance often arises in databases that are no longer in use, or that are used rarely. In some other cases incompleteness surges because the domain changes rapidly and the database is not updated on time. Example 1.1 illustrates such a situation: the database is not updated according to the new schedule of trains with destination Antwerpen Centraal.

- **Incomplete migration of data.** Changes in technology favor the migration of data from old repositories to faster, more reliable systems. During this process, some information may get lost. Failures in the power supply, lack of physical space or improper design of the target database may all contribute to an incomplete migration. Typical cases of incomplete migration are library archives that are upgraded to modern systems (see example 1.3 above).

- **Security and privacy.** In certain scenarios, it is necessary to leave out of the database sensitive information, because of security or privacy reasons. In this case the incompleteness is due to the intentional action of an agent. For instance, the administrative database of a commercial entity may omit the information about the salary of high rank associates. Similarly, the personal telephone numbers of political authorities are not available in telephone directories, in which case the numbers are tagged as 'unlisted' or 'private', and not displayed.

- **Accidental or malignant deletion of tuples.** Human and machine errors occur and sometimes by mistake information from a database is

erased. During the period of time that the deletion is committed and the missing data is recovered, the database remains incomplete.

Malignant deletion of tuples refers to the intentional elimination of data by some agent interested in a mismatch between the real world and the information in the database.

- **Mediator-based systems.** A mediator-based system consists of a number of heterogeneous databases integrated by a global interface – usually called global schema – by means of semantic information. In such a system the knowledge about the domain is distributed among the data sources that collaborate with the mediator. As the global schema stores no information, the system deploys the query and relies on the local data sources for answers. Each of the data sources is then regarded as incomplete with respect to the mediator's domain.

- **Peer-to-Peer systems.** A peer-to-peer system is a decentralized network of data resources which have agreed to collaborate and share information about a common domain. The user poses a query to one of the peers, which in turn distributes the query among other peers that may provide answers. As in mediator-based systems, every peer stores only part of the information about the common domain, and they are therefore regarded as incomplete databases.

1.1.2.1 The Open-World Assumption

Under the *Open-World Assumption* (OWA) (Abiteboul and Duschka 1998; Grahne 2002), every atom that is not in the database is regarded as unknown, and no standard truth value (true or false) in the strict sense is assigned. Let's illustrate the OWA with a simple example:

Example 1.4 *Consider again the Lirias database of Example 1.3. The following correspond to the answers to queries 1, 2 and 3 when the open-world assumption is adopted:*

- *Query 1: Retrieve the publications of 'Cortés' in 2006.*
  Answer 1: 'Partial knowledge in LC databases'.

- *Query 2: Are there publications of 'Cortés' in 2007?*
  Answer 2: Unknown.

- *Query 3: Are there publication of 'Cortés' in 2008?*
  Answer 3: Unknown.

Queries 2 and 3 are answered as 'unknown', since there is no information in the database to rule them as true (they are not stored in the database as such), and they are not known to be false either (the database stores only true facts about

the world). The answer to query 1 is the same as the one obtained when the closed world-assumption is adopted. In this case, however, the interpretation of the answer changes: it is everything the database knows and not the complete picture of the world.

As this example shows, the open-world assumption does not distinguish between tuples that are not in the database because they are false and tuples that are not in the database because they are unknown. All what is not in the database is simply regarded as unknown.

The closed and open-world assumptions represent two extremes. On then one hand, the closed-world assumption rules as false all the tuples that are not in the database. On the other hand, the open-world assumption regards all tuples that are not in the database as unknown. In many situations, nonetheless, it is the case that a database, while in general incomplete about its intended domain, stores some information that is complete. The following examples illustrate two such cases.

Example 1.5 *Consider a database of a computer science (CS) department that stores information about the telephone numbers of the department's members and collaborators. The database is represented in Table 1.4.*

Telephone	
Name :	*Telephone*
Lien Desmet	6531421
Lien Desmet	0923314
Bart Delvaux	5985625
Tom Demans	5845213

Deparment	
Name :	*Department*
Bart Delvaux	Computer Science
Lien Desmet	Philosophy
Tom Demans	Computer Science
David Finner	Biology

Table 1.4: A database of contact phone numbers for a CS department.

It is the case that this database is complete with respect to all computer science department members, but possibly incomplete regarding its external collaborators. Thus, appropriate answers for $Telephone(Bart\ Delvaux, 3962836)$ *and* $Telephone(Lien\ Desmet, 3212445)$ *are "no" and "unknown", respectively. If completeness of the database is taken for granted, then the answer for both queries is "no". For the query 'does David Finner have a telephone number?', under the CWA the answer is "no", but as the database is complete only with respect to the CS department, one cannot exclude the possibility that David Finner has a phone number, so the correct answer is "unknown".*

Example 1.6 *Consider now a distributed traffic tax administration system, in which there is one database for each county, maintaining a database of car owners in that county. There is a protocol among the different counties that whenever a car owner leaves one county* $\mathcal{A}$ *to live in another county* $\mathcal{B}$*, county* $\mathcal{A}$

Car Owners		
Name :	*Model*	*CarID*
Peter Steward	Mercedes 320	Qn-5452
John Smith	Volvo 230	Bx-5242
Mary Clark	BMW 550	Bx-5462

Location	
Name :	*Residence*
Peter Steward	Queens
Mary Clark	Bronx
John Smith	Bronx

Table 1.5: A database of the traffic tax administration system.

immediately transfers its information to county $\mathcal{B}$, while still preserving a record of the car owner and its current status for a certain period of time to handle all running tax demands. By the nature of the protocol, we may assume that each database has complete knowledge about all car owners in its county, but in general it has more information than that. Part of the tables of a particular county, say the Bronx, is represented in Figure 1.5.
By the nature of the distributed system, this database has expertise on car owners of the Bronx. This meta-knowledge allows to derive further information that is not explicitly stated in the database, e.g., that all people that are recorded in the table Location as residents of the Bronx, are actually all the car owners from that county. However, as the information about car owners in Queens is not complete in this database, one should not rely only on the tables of this source for making further conclusions of that kind related to that county.

Examples 1.5 and 1.6 illustrate situations in which database information is only *locally* complete, and so applying the closed-world assumption is not the right approach. As already shown, it might even lead to wrong conclusions. The other extreme approach, the open-world assumption, in which there is no closure at all, does not allow to express locally closed information so some valuable information may be lost. In the examples above, for instance, one cannot state a full knowledge regarding the phone numbers of the CS department members or about the car owners of the Bronx. The database is then in a state of *local completeness*.

1.1.3 The Local Closed-World Assumption

One way to deal with locally closed databases [4] is to augment the databases with *Local Closed-World Assumptions* (LCWAs), which are expressions of the following form:

$$\mathcal{LCWA}(P(\bar{x}), \Psi[\bar{x}])^5,$$

where P is the name of a database relation and $\Psi[\bar{x}]$ a first-order formula with free variables $\bar{x}$. Intuitively, this expression states that the database table of P

[4] In the course of this thesis we use equivalently the words closed and complete.
[5] In Section 3 we will provide a full account of local closed-world assumptions.

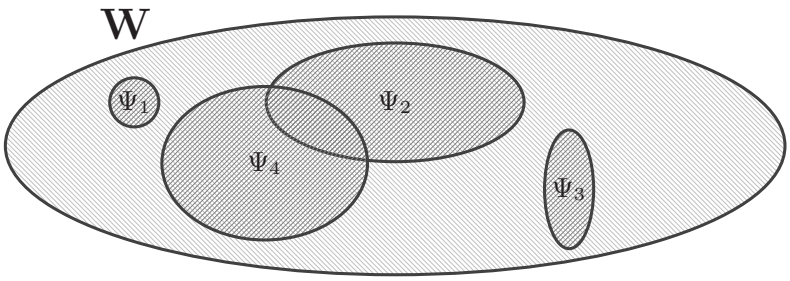


Figure 1.1: Graphical representation of the areas $\Psi_1, \ldots, \Psi_4$ where a locally closed database is complete.

is complete for all tuples $\bar{x}$ such that $\Psi[\bar{x}]$ holds in the domain of discourse. Or, for such tuples, $P(\bar{x})$ holds if and only if $P(\bar{x})$ is represented in the database. The formula Ψ is called a *window of expertise* of the database predicate P. In Example 1.5, for instance, the assumption that the table of telephone numbers is complete for all members of the CS department would be specified by

$$\mathcal{LCWA}(Telephone(p,t), Department(p, CS)),$$

and in Example 1.6 the local closed-world assumption would be expressed as:

$$\mathcal{LCWA}(CarOwners(p,c,i), Location(p, Bronx)).$$

The situation of locally closed databases can be depicted as in Figure 1.1. Areas $\Psi_1 \ldots \Psi_4$ represent the parts of the domain in which the database contains all true tuples, i.e. windows of expertise. In other areas of the domain the database, as a whole, remains incomplete.

1.2 Goals and Contributions of this Thesis

In this thesis, we study locally closed databases from both a representational and a computational perspective. At the declarative level, the contributions of this thesis are as follows:

- We present a first-order generalization of Levy's approach (Levy 1996) for representing partial completeness in relational database systems and

define its semantics. We show that with these local closed-world assumptions, we are able to capture both Reiter's CWA and the OWA. In addition, we investigate extensions such as local closed-world assumptions in a multi-source setting (as in mediator-based systems), or to express *full knowledge* about the domain elements of the database.

- We study the relationship of our notion of LCWA with other non-monotone formalisms for representing incomplete knowledge. Towards this goal, an equivalent representation is given based on second-order circumscription (McCarthy 1990). This provides the basis for extending the concept of LCWA from relational to so-called disjunctive databases, and allows to compare our proposal with other approaches for LCWA, expressed in terms of higher-order languages.

At the reasoning level, we study the basic reasoning tasks in locally closed databases: computing certain and possible answers of queries and determining whether the database has complete information on a query. We study the following topics:

- We analyze the computational complexity of computing certain and possible answers to queries. Not surprisingly, these problems turn out to be intractable in general.

- These intractability results then provide the motivation for the development of a set of efficient *approximate methods* for query answering based on three-valued logic. Approximate reasoning has recently emerged in many domains of computational logic. In databases, approximate query answering provides tractable lower and/or upper approximations to queries in deductive databases (Chaudhuri 1993; Chaudhuri and Kolaitis 1994), databases with NULL values (Libkin 1998), semistructured databases (Grahne and Thomo 2001) and integrated databases (Grahne and Mendelzon 1999b). More recently, approximate reasoning has been introduced to description logics in order to keep complexity of query answering over web ontologies under a certain threshold (Stuckenschmidt and van Harmelen 2002; Pan and Thomas 2007). There are three main parts to our contribution on approximate reasoning methods:

  – First, we describe a polynomial fixpoint procedure to compute a three-valued structure approximating *all* models of a locally closed database. We prove that such a structure can be used to compute overestimations of possible answers and underestimations of certain answers for all queries.

  – Second, we analyze the *precision* of our method. We show that, for a practically useful class of queries and locally closed databases, the approximate structure is *optimal*, and possible and certain answers

extracted from it are *exact*. In other words, for these queries and databases, *approximate* and *standard* query answering coincide. An interesting corollary is that we determine a class of databases and queries for which query answering is tractable after all.

- A weakness of the above method is the cost of the computation of the three-valued database, which, although polynomial, may be unacceptable for large databases. Worse, when the underlying database is modified, the three-valued structure must be updated as well, making this method impractical for databases with non-persistent data. We address this problem by presenting a method in which the computation of this three-valued database can be avoided. This method takes as input a query for possible or for certain answers and a collection of local closed-world assumptions, and *compiles* them into a new query in the form of a fixpoint query that can be run directly against the database. We will prove that this compiled query returns the same answers as the first, naive fixpoint method. As an alternative technique, we present an algorithm that, for a more restricted class of so called *hierarchically closed databases*, transforms the query with the local closed-world assumptions into a standard first-order query. Since these smarter methods compute the same answers as the naive method, it follows that they are *optimal* exactly where the naive methods are.

- In order to permit the combination and nesting of certain and possible modalities, we extend the basic query language with *modal* operators. To answer such modal queries efficiently we rely on the same approximation principles developed for answering standard queries.

Finally, at the implementation level we make the following contributions:

- We present a prototype based on in IDP (a model generator system for ID-Logic[6] theories) of the approximative methods described above. The system handles locally closed databases in conjunction with certain types of integrity constraints. An extension that allows the definition of deductive databases is also provided. The system receives as input a database, a set of LCWAs, and a query from the user. The system then extracts answers to the original query based on the input knowledge.

1.2.1 Methodology

This thesis is situated at the crossroads of three research domains: foundations of databases, knowledge representation and artificial intelligence (see figure

[6]ID-Logic is an extension of first-order logic with inductive definitions. We give a short introduction to it in Section 2.6.3.

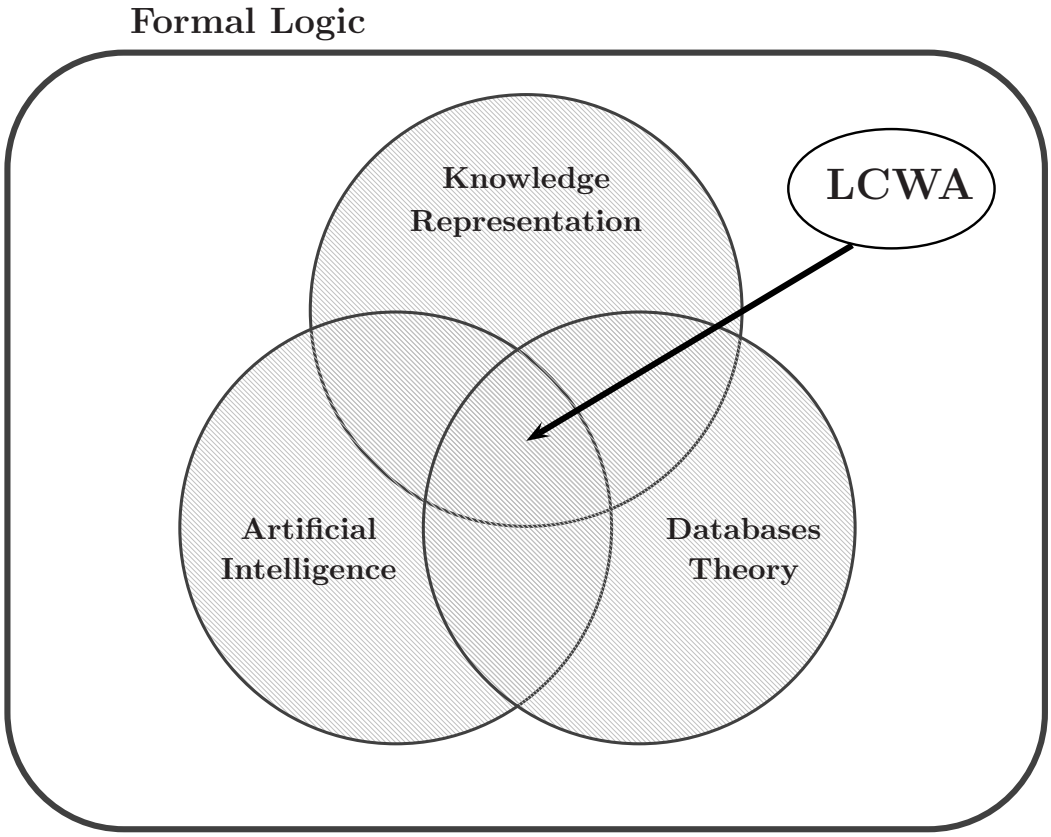


Figure 1.2: The local closed-world assumption touches three areas of knowledge.

1.2). We use tools from and make contributions to the three of them. The connection with databases has already been established in this introduction. The link with the other two fields deserve some further explanation.

- **Knowledge representation.** This area of research has been defined as the field of study concerned with using formal symbols to represent a collection of propositions believed by some putative agent (Brachman and Levesque 2004). In this thesis we study how to represent what a locally closed database *knows about the world*. In particular, we address how to formally *represent* the areas in the real world in which the database – the putative agent – stores all true tuples.
- **Artificial Intelligence.** Broadly defined, artificial intelligence is the study and design of intelligent agents (Russell and Norvig 2002). An important topic in this thesis is how to reason in the presence of locally closed databases. In our context the reasoning task takes the form of query answering, which corresponds to reasoning of the deductive kind.

1.2.1.1 The Role of Formal Logic

For the development of this thesis we rely on the tools of formal logic. As it has been widely acknowledged, logic provides a very useful if not indispensable tool in computer science in general and in the areas of databases, knowledge representation and artificial intelligence in particular:

> "..logic has permeated through computer science during the past thirty years much more than it has through mathematics during the past one hundred years. Indeed, at present concepts and methods of logic occupy a central place in computer science, insomuch that logic has been called the calculus of computer science..." (Halpern et al. 2001)

Throughout this work we will rely on several logics.

- At the representational level (Chapter 3), we define the semantics of the local closed-world assumption and of locally closed databases in terms of *first-order logic*. Towards the comparison of our approach with other proposals to represent partial knowledge we provide alternative semantics based on *second order circumscription*.
- In the query answering section (Chapter 4) we rely on *Kleene's three-valued logic* to approximate all models of a locally complete database. To simulate approximative query answering we use tools from *fixpoint logics*. These logics are extensions of first-order logic with fixpoint constructions.
- In Chapter 5 we study a modal query language for the local closed-world assumption. This language is a variant of the well-known S45 flavor of *modal logic* (Chellas 1980).

- The prototype implementation relies on IDP, a model generator system for *ID-Logic* theories. ID-Logic is an extension of first-order logic with inductive definitions.

A full introduction to each of these logics is provided in Chapters 2 and 4.

1.3 Organization of the Text

This thesis is organized in six chapters.

- In Chapter 2 we introduce the technical content that will serve as a background for the rest of the text.[7] We present a short introduction to several logical formalisms and some basic concepts of set and lattice theory.

- In Chapter 3 we formally introduce the concept of a locally closed database. We proceed with defining its semantics and the study of different reasoning tasks. Next we provide a number of complexity results that motivate the use of approximations for query answering. A comparison of our approach with other formalisms for representing local completeness follows. We conclude this chapter proposing some extensions of the local closed-world assumption to more general settings.

- In Chapter 4 we propose several techniques for query answering in locally closed databases. We study these techniques both from a computational and semantical perspective. In particular, we present the logical foundations of three-valued logic for query approximations. To tackle important drawbacks of computing a three-valued approximation for query answering, we provide an algorithm based on fixpoint logic that allows to retrieve answers while leaving the three-valued structure implicit. An analysis of the optimality of approximative query answering is also developed.

- In Chapter 5 we extend the query language with modalities. We provide the semantics of the new formalism in terms of Levesque's variant of the modal logic S45.

  We conclude this chapter with a prototype implementation of the LCWA. The system makes use of a model generator system for ID-Logic theories.

- We conclude this thesis in Chapter 6 with a discussion of the obtained results and by proposing possible topics for future research.

[7] Less general formalisms are introduced in the specific chapters where they are applied.

1.3.1 Dependency of Chapters

The text is intended to be read sequentially. Nonetheless, the reader already familiar with mathematical logic and knowledge representation may skip Chapter 2. Readers interested in using the implementation that is described in the text can directly move to Chapter 5.

1.4 Bibliographical Note

Some parts of this text have been published before. The following list contains the key articles. A complete publication list of the author can be found at the end of this thesis.

- **Chapter 3: The Local Closed-World assumption**
  - A. Cortés-Calabuig, M. Denecker, Ofer Arieli, B. Van Nuffelen, and M. Bruynooghe, *On the local closed-world assumption of data-sources*, 8th International Conference on Logic Programming and Non-Monotonic Reasoning, LPNMR 2005, Proceedings (C. Baral et al. eds.), vol 3662, Lecture Notes in Artificial Intelligence, pp. 145-157, Springer, 2005.
  - A. Cortés-Calabuig, M. Denecker, Ofer Arieli, and M. Bruynooghe, *Representation of partial knowledge and query answering in locally closed databases*, 13th International Conference on Logic for Programming Artificial Intelligence and Reasoning, LPAR 2006, Proceedings (M. Hermann et al. eds.), vol 4246, Lecture Notes in Computer Science, pp. 407-421, Springer, 2006.
- **Chapter 4: Query answering under the LCWA**
  - A. Cortés-Calabuig, M. Denecker, Ofer Arieli, and M. Bruynooghe, *Approximate query answering in locally closed databases*, 22nd National Conference on Artificial Intelligence, AAAI 2007, Proceedings, pp. 397-402, AAAI Press, 2007
  - A. Cortés-Calabuig, M. Denecker, Ofer Arieli, and M. Bruynooghe, *Efficient fixpoint methods for approximate query answering in locally closed databases*, 1st Symposium on Logic in Databases, LID 2008, Informal Proceedings (G. Gottlob general chair), 2008.
  - A. Cortés-Calabuig, M. Denecker, Ofer Arieli, and M. Bruynooghe, *Accuracy and efficiency of fixpoint methods for approximate query answering in locally complete databases*, 11th International Conference on Principles of Knowledge Representation and Reasoning, KR 2008, Proceedings, pp. 81-91, AAAI press, 2008.

Chapter 2

Technical Background

In this chapter, we present the technical background necessary to read through the rest of this thesis. The reader familiar with the basic ideas of formal logic and nonmonotonic reasoning may safely skip this section. A bibliographical note is presented at the end of the chapter.

2.1 Basics of Set Theory

We begin this section with presenting some basic concepts from set theory and lattice theory. A set, in this thesis always denoted by a capital letter, is a collection of 'elements'. The symbol $\in$ denotes membership and $\notin$ its negation. To denote that x and y are the same object we use $= y$, and we read it as 'x is equal to y'. The equality relation may be extended to sets. $A = B$ means that for any object a it is true that $a \in A$ if and only if $a \in B$.

The *empty* set is denoted by $\emptyset$ and it has no members. $A \subseteq B$, read as 'A is a subset of B', indicates that all members of A are also members of B. For any set A the empty set is a subset of A. $A \cup B$, read as 'A union B', is the set of all elements that are in A, B or both. Two sets are *disjoint* if they have no member in common. The *intersection* $A \cap B$ is the set of all elements that are members of A and B.

An *ordered pair* is a collection of two distinct objects, where one object is identified as the first coordinate and the other as the second coordinate. A *relation* R is a set of ordered pairs. The *cross product* of A and B, denoted $A \times B$, is the set of all pairs (x, y) for which $x \in A$ and $y \in B$. A^n is the set of all n-tuples of elements of A. A n-ary relation over a set A is a subset of A^n. An arbitrary relation R over a set A may have certain properties:

- R is *reflexive* over A iff $(x, x) \in R$ for every x in A.
- R is *symmetric* iff whenever $(x, y) \in R$, $(y, x) \in R$.

- R is *transitive* iff whenever both $(x, y) \in R$ and $(y, z) \in R$, $(x, z) \in R$.
- R is an *equivalence relation* on A iff R is a binary relation on A which is reflexive, symmetric and transitive.

2.2 Enumerability

We say that a set is *denumerable* if all its members can be enumerated. Intuitively, we can enumerate a set if we can arrange it in a single list consisting of a first element, a second element and so on. The empty set is by definition denumerable. Denumerable sets can be finite or infinite. The set P of natural numbers can be enumerated by the list $1, 2, 3, 4, \ldots$. The following list is not accepted as a list of positive integers: $1, 3, 5, \ldots, 2, 4, 6, 8$ because in an acceptable list each item must appear as the nth entry for a finite n.

2.3 Order and Lattice Theory

An order is a relation of a specific kind, which is defined by two essential properties: *transitivity* and *antisymmetry*. More formally,

Definition 2.1 (Order or Partial Order) *Let P be a set. An* order *(or* partial order*) on P is a binary relation $\leq$ on P such that for all $x, y, z \in P$,*

$$\begin{aligned} &x \leq x && \textit{(Reflexivity)}\\ &x \leq y \text{ and } y \leq x \text{ implies } x = y && \textit{(Antisymmetry)}\\ &x \leq y \text{ and } y \leq z \text{ implies } x \leq z && \textit{(Transitivity)} \end{aligned}$$

If P satisfies these three properties we say P is an *ordered set* or simply a *poset*.

Example 2.1 *The set of natural numbers $\mathbb{N}$ forms a partial order with $\leq$.*

An ordered set P is a *chain* (alternatively *linearly ordered set* or *totally ordered set*) if for all distinct $x, y \in P$, either $x \leq y$ or $y \leq x$. I.e., if all the elements of the set are comparable with each other.

An ordered set P has a *bottom* element if there exists Bottom element$\bot \in P$ such that $\bot \leq x$ for all $x \in P$. Alternatively, P has a *top* element if there exists $\top \in P$ such that $x \leq \top$ for all $x \in P$. A finite chain has both a bottom and a top element.

Let P be an ordered set. We say that $a \in Q \subseteq P$ is the *maximal* element of Q if $a \leq x$ and $x \in Q$ imply $a = x$. The maximal elements of Q are denoted by Max_Q. If Q has a top element, $\top_Q$, then $Max_Q = \{\top_Q\}$. In such case, $\top_Q$ is called the *greatest* element of Q. The *minimal* elements of $Q \subseteq P$ and the *least* element of Q are defined by reversing the order.

If P is a finite ordered set, then any non-empty subset of P has at least one maximal element and, for each $x \in P$, there exists $y \in Max_P$ with $x \leq y$. A subset Q of an ordered set P may have none, one or several maximal elements.

2.3.1 Lattices as Ordered Sets

Let P be an ordered set and let $S \subseteq P$. An element $x \in P$ is an *upper bound* of S if $s \leq x$ for all $s \in S$. A *lower bound* is defined inversely. The set of all upper bounds of S is denoted by S^u, and the set of lower bounds by S^l. The least element of S^u, if it exists, is called the *least upper bound* of S. The dual of the least upper bound is the *greatest lower bound* of S. If they exist, least upper bounds and greatest lower bounds are always unique. The least upper bound of S is also called the *supremum* of S, and it is denoted by $SupS$. The greatest lower bound, denoted by $InfS$, is also called *infimum* of S. In the following we denote $x \vee y$ the least upper bound of $\{x, y\}$ and $x \wedge y$ the greatest lower bound of $\{x, y\}$.

Definition 2.2 *Let P be a non-empty ordered set. If $x \wedge y$ and $x \vee y$ exist for all members of P, then P is called a* lattice. *If $\bigwedge S$ and $\bigvee S$ exist for all $S \subseteq P$, then P is called a* complete *lattice.*

A way to check whether a non-empty ordered set P is a lattice, consists in proving that $x \wedge y$ and $x \vee y$ exist in P for all non-comparable elements of P. The well-known sets of natural, rational, integers and real numbers are all lattices under their usual order. A well-known property of lattices is that every finite lattice is complete.

Certain properties of monotone functions over lattices will prove useful in the query answering section of this text.

Definition 2.3 *Let $F : P \to Q$ be a mapping on ordered sets P and Q. F is* monotone *if $x \leq y$ implies $F(x) \leq F(y)$ for all elements of P. F is anti-monotone if $x \leq y$ implies $F(y) \leq F(x)$ for all elements of P.*

In this work we will mostly define mappings from a poset P to itself. We say that $x \in P$ is a *fixpoint* of a mapping F if $F(x) = x$, a *pre-fixpoint* if $F(x) \leq x$ and a *post-fixpoint* if $x \leq F(x)$. The least and the greatest fixpoints of F (if they exist) will be denoted by $lfp(F)$ and $gfp(F)$. A well-known class of ordered sets for which every monotone map has a least fixpoint are the chain-complete posets.

Definition 2.4 *A poset P is* chain-complete *if and only if every chain has a supremum.*

The following proposition formalizes the property stated in the paragraph above:

Proposition 1 *If $F : P \to P$ is a monotone function on a chain-complete set P, then F has a least fixpoint.*

The following theorem states the importance of chain-complete posets for having fixpoints on monotone function.

Theorem 1 *Let P be an ordered set. Every monotone map $F : P \to P$ has a least fixpoint if and only if P is chain complete.*

The following definition describes how to compute least fixpoints of mappings.

Definition 2.5 *The* ordinal powers *of F starting from x are defined as follows:*

$$F^0(x) = x$$
$$F^{\alpha+1}(x) = F(F^\alpha(x)) \text{ for a successor ordinal } \alpha + 1$$
$$F^\alpha = \bigvee_{\beta<\alpha} F^\beta(x) \text{ for a limit ordinal } \alpha$$

We can then use ordinal powers to compute fixpoints:

Proposition 2 *There exists an ordinal ∞ such that $F^\infty(\bot) = lfp(F)$.*

2.4 Computational Complexity

In this section, we present the basic concepts of computational complexity that we use along the text. A *vocabulary* Σ is a finite set of symbols. The set of all (finite) strings over Σ is denoted by Σ^*. A *language* $\mathcal{L}$ is a subset of Σ^*.

Definition 2.6 *A decision problem is a par $(\mathcal{L}, \mathcal{Y})$ of a language $\mathcal{L}$ and a set $\mathcal{Y} \subseteq \mathcal{L}$ of yes-instances. A string $s \in \mathcal{L}$ is called an instance of a problem.*

A class is a set of decision problems. The complement $\mathsf{co}\mathcal{C}$ of a class $\mathcal{C}$ is defined as

$$\mathsf{co}\mathcal{C} = \{(\mathcal{L}, \mathcal{L} - \mathcal{Y}) | (\mathcal{L}, \mathcal{Y}) \in \mathcal{C}\}$$

To study complexity, the standard computational model is a Turing machine. Intuitively, a Turing machine consists of a read-write head and an infinite tape. Initially, the head is located at the start of an input string and the machine is in a pre-defined initial state. At every step of the computational process, the Turin machine reads the input symbol at the current position of the head. Depending on the state, the machine then writes a new symbol at the current position, changes to a new state and moves the head to the left or right.

Definition 2.7 *A Turing machine M is a tuple $\langle Q, \Sigma, \delta, q_0, F \rangle$, where:*

- *Q is a finite set of states,*

- Σ *is a finite vocabulary,*
- $q_0 \in Q$ *is the initial state,*
- $F \subseteq Q$ *is a set of final states, and*
- $\delta : (Q - F) \times \Sigma \rightarrow Q \times \Sigma \times \{-1, +1\}$ *is a transition function.*

A Turing machine with a non-deterministic transition function is a Turing machine that is non-deterministic. In that case, in every step of the computation the machine may have different choices how to proceed. An input string is accepted by a non-deterministic Turing machine if there exists a computation which ends in a final state.

We now define the important class of polynomial time computable decision problems.

Definition 2.8 *The class P of polynomial time computable decision problems consists of all decision problems $(\mathcal{L}, \mathcal{Y})$ for which there exists a deterministic Turing machine which for every instance $s \in \mathcal{L}$ decides in polynomial time in the size of s if $s \in \mathcal{Y}$. The class NP is defined in a similar way but using non-deterministic instead of deterministic Turing machines.*

We now introduce the notion of complete problems. These are the 'most difficult' problems for a given class since all other problem in the same class can be 'reduced' to them. The notion of reduction is made precise in the following definition.

Definition 2.9 *A polynomial transformation from a problem $\Pi_1 = (\mathcal{L}_1, \mathcal{Y}_1)$ to a problem $\Pi_2 = (\mathcal{L}_2, \mathcal{Y}_2)$, denoted with $\Pi_1 \propto \Pi_2$, is a function $f : \mathcal{L}_1 \rightarrow \mathcal{L}_2$ which is computable by a polynomial time Turing machine and for every $x \in \mathcal{L}_1$, $x \in \mathcal{Y}_1$ if and only if $f(x) \in \mathcal{Y}_2$.*

Next, we formalize the notion of complete decision problem.

Definition 2.10 *A decision problem Π is complete for a complexity class $\mathcal{C}$ if $\Pi \in \mathcal{C}$ and $\Pi' \propto \Pi$ for all $\Pi' \in \mathcal{C}$.*

In databases theory, two complexity measures for query languages are generally used: *data complexity* and *expression complexity*. Data complexity is the complexity of evaluating a query in the language as a function of the size of the database. Expression complexity is the complexity of evaluating a query in the language as a function of the size of the expression defining the query (Vardi 1982). In this text, unless stated otherwise, complexity results will be presented in terms of data complexity.

2.5 Classic Logical Formalisms

Logical expressions allow us to manipulate syntactical objects that refer to certain aspects of a reality or *domain*. We will consider logic as formal languages consisting of two essential parts: one syntactical and the other semantical. The meaning of the syntactical representation is evaluated with respect to the formal interpretation, i.e. the semantics, that is assigned to them. In this section we present the logical formalisms that are used in the remainder of the text. We start with the simplest of all, propositional logic.

2.5.1 Propositional Logic

We start with a finite set of distinct elements which are called symbols. We describe the elements of this set as follows:

- *Logical connectives*: $\neg, \wedge, \vee, \supset, \equiv, .$
- *Punctuation symbols*: $,)($.
- A fixed set P of *propositional variables*

Normally the set P is assumed to be finite, although P may also be infinite denumerable, for instance $p_1, p_2, p_3, \ldots$. Logical and punctuation symbols are fixed in every propositional language while the propositional variables may vary. Since a propositional language depends on its propositional variables, we will refer to it as a function of the variables. For instance, if the set of variables in the language is P, the propositional language will be denoted by $\mathcal{L}(P)$.

We proceed now to formally define a formal propositional language based on variables P.

Definition 2.11 *The propositional language $\mathcal{L}(P)$ is formed exactly from those formulas obtained by a finite application of the following rules:*

- *Every propositional variable is a* formula*. These formulas are named* atomic *formulas.*
- *If ϕ is a formula, then $\neg\phi$ is also a formula.*
- *If ϕ and ψ are formulas, then $(\phi * \psi)$ is also a formula. Here $*$ represents any of the logical connectives $\wedge, \vee, \supset, \equiv$.*

When clear from the context we will omit the symbols) and (. We now define the two-valued semantics of the propositional language $\mathcal{L}(P)$ by assigning *truth values* to the propositional variables of the language. We assume each propositional variable takes one and only one of the truth values $\mathbf{t}$ and $\mathbf{f}$ [1].

[1] Throughout this thesis truth values $\mathbf{t}$ and $\mathbf{f}$ are associated and are interchangeable with the numbers 1 and 0.

Definition 2.12 *A* truth assignment *(sometimes referred to as* valuation*) is a function σ of P to the set $\{\mathbf{t}, \mathbf{f}\}$.*

Observe that there are 2^n truth assignments for a set P of n propositional variables. Note also that every propositional variable maps to one and only one of the defined truth values. Later in this text we will define languages for which a third value *unknown* may be assigned to atomic sentences.

Definition 2.13 *Consider a propositional language $\mathcal{L}(P)$ and a valuation $\sigma : P \rightarrow \{\mathbf{t}, \mathbf{f}\}$. We define $\sigma : \mathcal{L}(P) \rightarrow \{\mathbf{t}, \mathbf{f}\}$ as follows: given an arbitrary formula $\psi \in \mathcal{L}(P)$, the truth value $\sigma(\psi)$ is:*

1. *Base case: If $\psi \in P$ (i.e. ψ is an atomic formula), the $\sigma(\psi)$ corresponds to the truth valued originally assigned to ψ.*
2. *Inductive steps:*
   - *If ψ is $\neg\phi$, then $\sigma(\psi) = 1 - \sigma(\phi)$.*
   - *If ψ is $(\phi_1 \vee \phi_2)$ then $\sigma(\psi) = max\{\phi_1, \phi_2\}$.*
   - *If ψ is $(\phi_1 \wedge \phi_2)$ then $\sigma(\psi) = min\{\phi_1, \phi_2\}$.*
   - *If ψ is $(\phi_1 \supset \phi_2)$ then $\sigma(\psi) = \mathbf{f}$ if $\sigma(\phi_1) = 1$ and $\sigma(\phi_2) = \mathbf{f}$. Otherwise $\sigma(\psi) = 1$.*
   - *If ψ is $(\phi_1 \equiv \phi_2)$, then $\sigma(\psi) = 1$ if $\sigma(\phi_1) = \sigma(\phi_2)$. Otherwise, $\sigma(\phi_2) = \mathbf{f}$.*

Two formulas are *logically equivalent* if exactly the same valuations make them true. We say a formula $\psi \in \mathcal{L}(P)$ is a *tautology* if and only if for every valuation $\sigma : P \rightarrow \{\mathbf{t}, \mathbf{f}\} : \sigma(\psi) = \mathbf{t}$. A formula $\psi \in \mathcal{L}(P)$ is *satisfiable* iff there exists a valuation $\sigma : P \rightarrow \{\mathbf{t}, \mathbf{f}\}$ such that $\sigma(\psi) = \mathbf{t}$.

Finally, we a say a propositional formula ψ is *unsatisfiable* (or *inconsistent*), if ψ is not satisfiable.

The concept of valuation in formulas can be further extended from single formulas to a set of formulas (sometimes called *theories*). Suppose Γ is a set of formulas of a propositional language $\mathcal{L}(P)$. We define the truth value of Γ with respect to a valuation σ as:

$$\sigma(\Gamma) = min\{\sigma(\psi) \mid \psi \in \Gamma\}.$$

We say σ *models* Γ, denoted by $\sigma \models \Gamma$, iff for every $\psi \in \Gamma$, $\sigma(\psi) = 1$ or equivalently, if $\sigma(\Gamma) = 1$.

A set of formulas Γ, can be tautogically true, satisfiable or inconsistent depending on whether respectively all, some, or any valuation(s) satisfy Γ.

With the notion of satisfiability of theories in order, we proceed to define the concept of entailment (or logical consequence), which semantically relates formulas and theories.

Definition 2.14 *Let Γ be a set of formulas of a propositional language $\mathcal{L}(P)$, and $\psi \in \mathcal{L}(P)$. We say ψ is* entailed *by (or a logical consequence of) Γ, if for every valuation σ, $\sigma \models \Gamma$ implies $\sigma \models \psi$. This is denoted by $\Gamma \models \psi$.*

To check whether a formula is entailed by a set of formulas, we need to verify the truth value of the formula with respect to those valuations that make truth the original set of formulas.

2.5.2 First-order Logic

First-order logic is an extension of propositional logic that permits quantifications over the elements of the domain.

A first-order vocabulary Σ consist of the following symbols:

- Punctuation symbols: ,)(.
- Logical connectives: $\neg, \wedge, \vee \subset, equiv, \forall, \exists$.
- Variable symbols: $x_1, \ldots x_n$.
- Function symbols: $f_1, \ldots f_n$. Function symbols of arity 0 are called *constants*, and their set is denoted by $\mathcal{C}(\Sigma)$.
- A set $\mathcal{R}(\Sigma)$ of predicate or relation symbols: $R_1, \ldots, R_n$.
- Fixed interpreted symbols: $\mathbf{t}, \mathbf{f}$ and $=$.

Each relation and function symbol has an associated arity. When clear from the context, the arity of predicates will be omitted. Primitive elements referring to domain elements are called terms.

Definition 2.15 *The set T of* terms *over a given first-order vocabulary Σ is the smallest set such that:*

- *Any constant $a \in \Sigma$ is in T.*
- *Any variable $x \in \Sigma$ is in T.*
- *If f/n is a function symbol in Σ and $t_1, \ldots t_n \in T$ then $f(t_1, \ldots, t_n) \in T$.*

We now define the set of well-formed formulas of a first-order language based on Σ.

Definition 2.16 *Let T be the set of terms over Σ. The set F of well-formed formulas is the smallest set such that:*

- *If R is a predicate symbol in Σ and $t_1, \ldots, t_n \in T$ then $R(t_1, \ldots, t_n) \in T$.*
- *If ψ_1 and $\psi_2 \in F$, then so are $\neg\psi_1$, $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$, $\psi_1 \supset \psi_2$ and $\psi_1 \equiv \psi_2$.*

- *If $\psi \in F$ and x is a variable in Σ, then $\forall x\psi$ and $\exists x\psi \in F$.*

Since the formulas of a first-order language are determined solely on their vocabulary Σ, we will often refer to first-order languages by the vocabularies they are based on.

Formulas of the form $R(t_1, \ldots, t_n)$ are called *atomic* formulas, or *atoms*. The occurrence of a variable x in a formula ψ is *bound* if either the occurrence of x follows directly after a quantifier or if it appears inside the subformula which follows after $\forall x$ or $\exists x$. Otherwise the occurrence is *free*. A formula with no free variables is called a *closed formula* or a *sentence*. A formula or term that contains no variables is called *ground*. The *universal closure* of a formula ψ with free variables $x_1, \ldots x_n$ is the formula $\forall x_1, \ldots x_n\psi$. Dually, the existential closure of ψ is the formula $\exists x_1, \ldots x_n\psi$.

Logical sentences may be true or false in a certain world, called the *domain of discourse*. The mathematical abstraction of a world is called a *structure*.

Definition 2.17 *A* Σ-structure *(sometimes called an* interpretation*) I of a first-order vocabulary Σ consists of a non-empty set $dom(I)$ called the domain of I and a mapping that associates:*

- *each constant $c \in \mathcal{C}(\Sigma)$ with an element $c^I \in dom(I)$.*
- *each n-ary function symbol $f \in dom(I)$ with a function $f^I : dom(I)^n \to dom(I)$.*
- *each n-ary predicate symbol $R \in \mathcal{R}(\Sigma)$ with a relation $R^I \subseteq dom(I)^n$.*

The first two of these mappings allow us to associate a domain element t^I with a variable free term t. Denoting a tuple of terms by $\bar{t}$ and a tuple of domain elements by $\bar{t}^I$, and setting $\mathbf{f} \leq \mathbf{t}$ and $\mathbf{f}^{-1} = \mathbf{t}$ and $\mathbf{t}^{-1} = \mathbf{f}$, the truth value of a variable free formula ψ in a structure I, denoted ψ^I, is defined recursively as follows:

$$
\begin{array}{l}
P(\bar{t})^I = \mathbf{t} \text{ if } \bar{t}^I \in P^I \text{; otherwise } P(\bar{t})^I = \mathbf{f};\\
(\psi \wedge \phi)^I = Min_{\leqslant}(\psi^I, \phi^I);\\
(\psi \vee \phi)^I = Max_{\leqslant}(\psi^I, \phi^I);\\
(\neg\psi)^I = (\psi^I)^{-1};\\
(\forall x\psi[x])^I = Min_{\leqslant}\{(\psi[a])^I \mid a \in Dom^I\};\\
(\exists x\psi[x])^I = Max_{\leqslant}\{(\psi[a])^I \mid a \in Dom^I\}.
\end{array}
$$

We denote $I \models \varphi$ if $\varphi^I = \mathbf{t}$.

Some important concepts of propositional logic have their equivalents in first-order logic. A structure I *models* a set of closed formulas Γ (called a *theory* Γ) iff every formula of Γ is true in I. We say Γ is *satisfiable* if it has at least one model. We now define formally the notion of logical consequence for the first-order case.

Definition 2.18 *Let Γ be a set of sentences. A sentence ψ is called a* logical consequence *of Γ, denoted by $\Gamma \models \psi$, iff ψ is true in every model of Γ.*

Proposition 3 *Let Γ be a set of first-order sentences and ψ a formula. Then, $\Gamma \models \psi$ iff $\Gamma \cup \{\neg\psi\}$ is unsatisfiable.*

Two first-order sentences are *logically equivalent* iff both sentences have the same truth value for all interpretations.

When studying query answering mechanisms for locally closed databases we will use the notion of *substitution*.

Definition 2.19 *A* substitution *θ is a finite set of pairs of terms $\{x_1/t_1, \ldots, x_n/t_n\}$, where each t_i is a term and each x_i is a variable such that $x_i \neq t_i$ and $x_i \neq x_j$, if $i \neq j$. The* application *$x\theta$ of a substitution θ to a variable x is defined as follows:*

$$x\theta = \begin{cases} \mathbf{t} & \text{if } x/t \in \theta. \\ x & \text{otherwise .} \end{cases}$$

The application of a substitution θ to a term or formula ψ, denoted by $\psi\theta$ is the term or formula obtained by simultaneously replacing t_i for every free occurrence of x_i in θ.

The concepts of tautology, logical consequence and satisfiability for the propositional case have also their equivalents in the first-order case.

Definition 2.20 *A first-order sentence is* valid *iff it is true in every first-order interpretation. A sentence is* satisfiable *iff is true in at least one first-order interpretation.*

The concept of model is now defined for the first-order case.

Definition 2.21 *A* model *M of a theory Γ based on vocabulary Σ is an interpretation such that for each formula ψ of Γ, $M \models \psi$.*

An important class of structures (resp. models) we will consider in this thesis are the so-called *Herbrand interpretations* (resp. models).

Definition 2.22 *Let Σ be a first-order vocabulary. The* Herbrand universe *$HU(\Sigma)$ is the set of all ground terms of Σ. The* Herbrand base *$HB(\Sigma)$ is the set of all ground atoms of Σ. A* Herbrand interpretation *HI based on vocabulary Σ is an interpretation such that:*

- *The domain of HI is $HU(\Sigma)$.*
- *For every constant c, c^{HI} is interpreted as itself (c).*
- *Every n-ary function symbol f is interpreted as follows:*

$$f^{HI}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n), \text{ with } t_i \in HU(\Sigma) \ .$$

- *Every predicate symbol P of arity n is interpreted as a subset of HU^n.*

Since in every Herbrand interpretation based on a vocabulary Σ the Herbrand universe and Herbrand base remain constant, it suffices to specify a particular Herbrand interpretation by a subset of the Herbrand base.

A Herbrand interpretation that models a theory is called Herbrand model. Formally:

Definition 2.23 *A Herbrand model of a set of (closed) formulas is a Herbrand interpretation which is a model of every formula in the set.*

Example 2.2 *Consider the first-order theory*

$$\Gamma = \{P(a,b), \forall y.R(a,y) \supset Q(y,b)\}.$$

In this thesis, whenever the vocabulary of the theory is not explicitly defined, as in the current case, we will assume that it consists solely of the constants, functions and relations that are included in the theory. In this case,

$$\begin{aligned} HU &= \{a,b\} \\ HB &= \{P(a,a), P(a,b), P(b,a), P(b,b), \\ &\quad R(a,a), R(a,b), R(b,a), R(b,b) \\ &\quad Q(a,a), Q(a,b), Q(b,a), Q(b,b)\} \end{aligned}$$

The Herbrand interpretation $I_1 = \{P(a,b), R(a,b), Q(a,b)\}$ is a structure that does not model Γ. The Herbrand structure $I_2 = \{P(a,b), R(a,b), Q(b,b), Q(a,a)\}$ is a Herbrand model of Γ.

An important property of Herbrand models is that we can reduce satisfiability of first-order theories into satisfiability in Herbrand models.

Theorem 2 (Herbrand theorem) *A set of universal sentences[2] is satisfiable iff it has a Herbrand model.*

Propositional and first-order logic share the aforementioned property of monotonicity: previous conclusions are not retracted when new information is added to a theory. Many applications of artificial intelligence, specially those where so-called *commonsense* reasoning is needed, require logical formalisms which support reasoning of a nonmonotonic nature. In the following section we describe a number of logics that support nonmonotonic reasoning.

[2] A universal sentence is a sentence that consists of universal quantifiers followed by a quantifier-free formula.

2.6 Nonmonotonic Logical Formalisms

Nonmonotonic reasoning deals with the derivation of feasible conclusions from a certain set of statements about the world. The conclusions are fallible in the sense that they may be retracted if new information has been added to an initial set of statements. We can illustrate the idea of nonmonotonic reasoning with a simple example:

Example 2.3 *Consider the sentence 'Trees have green leaves'. Under a common sense assumption we do not want to imply with this sentence that* all *trees are green. What we really mean, and it is how we expect that a charitable interlocutor would understand the sentence, is that* in general *trees are green. So if we are presented with a tree, based solely in the information that trees have green leaves, we could assume, perhaps defeasibly, that the tree is green. There are of course many exceptions to the general statement: some trees have red leaves, others yellow leaves in autumn, and some others (in winter, for instance) do not have leaves at all. So if we got to know that the tree is of special kind, for instance a maple in autumn, we may retract our conclusion.*

This kind of reasoning is non deductive, but plausible: it takes into consideration external assumptions rather than infallible conclusions. In general, nonmonotonic reasoning deals with statements of the form:

It is often the case that A holds.

Usually, A is true.

Unless something special happens, we assume A be default.

All these sentences follow the same pattern:

In the absence of knowledge to the contrary, assume 'A'.

This type of reasoning is sometimes referred to as *commonsense reasoning*: anyone with some commonsense would in principle assume 'A'.

First-order logic, precisely because of its monotonic nature, is not suitable for capturing the subtleties of nonmonotonic reasoning. Suppose we want to capture in first-order logic the commonsense interpretation of the sentence 'Trees have green leaves'. A naive attempt to capture the meaning of this sentence is through a material implication:

$$\forall x.Tree(x) \supset Green(x).$$

It is not difficult to see that this sentence does not do the job: it just says that *all* trees are green, and no exceptions to the rule are allowed. That is, of course, not the real meaning of our natural language sentence.

A second alternative is to include into the formula the exceptions to the rule:

$$\forall x.Tree(x) \wedge \neg Yellowleaves(x) \wedge \neg Redleaves(x), \cdots \supset Green(x).$$

In this case, to derive that a tree is green we need additional *explicit* information about the tree, i.e., that it has no yellow leaves, no red leaves and so on. Of course, in having to make explicit all these assumptions the whole idea of reasoning by default vanishes.

An alternative to the last formalization is to group all the exceptions to the rule in one single predicate of 'abnormality'. In the working example,

$$\forall x.Tree(x) \wedge \neg Ab(x) \supset Green(x).$$

Although more compact, this attempts suffers from the same problem of the previous formalization: we need to define explicitly for each element of the domain whether it is abnormal or not.

The shortcomings of first-order logic to support nonmonotonic reasoning have motivated the introduction of several non-standard logical formalisms. In the remainder of this section we present a number of them that will turn out to be related to the local closed-world assumption of databases.

2.6.1 Circumscription

Circumscription (McCarthy 1986; Reiter 1978; Lifschitz 1994) is a type of nonmonotonic reasoning that is based on but extends first-order logic. In its paradigmatic form, circumscription was introduced by J. McCarty (McCarthy 1986) to cope with the difficulties of representing dynamic domains using standard first-order logic.

In the words of McCarthy, the main idea of circumscription is that "the objects that can be shown to have certain property P by reasoning from certain facts A are all the objects that satisfy P" (McCarthy 1986). More formally, circumscription is a transformation of a logical sentence A into a sentence A^* such that the models of A^* are exactly the minimal models of A. In other words, under circumscription we consider as valid only the models that satisfy a certain minimality condition.

Example 2.4 *Consider again example 2.3 and our last attempt in first-order logic to capture the meaning of 'Trees have green leaves'.*

$$\forall x.Tree(x) \wedge \neg Ab(x) \supset Green(x).$$

Under first-order logic semantics this rule says that if an element of the domain is a tree and it is not abnormal, then it has green leaves. Suppose we additionally know that $Tree(pine)$*. Under standard first-order logic semantics we do*

not obtain as a logical consequence that Green(pine), since we know nothing about whether Ab(pine) holds or not, so we can not make the antecedent of the rule true. However, if we circumscribe the predicate Ab, i.e. we accept only a minimal interpretation for it -in this case the empty one- we obtain that Green(pine).

Let ψ be a first-order sentence from a language Σ that contains the predicate P. Denote by $\psi(P')$ the result of replacing all occurrences of P in ψ by P'.

Definition 2.24 *The* circumscription *of P in $\psi(P)$ is the sentence schema*

$$\psi(P') \wedge \forall \bar{x}.(P'(\bar{x}) \supset P(\bar{x})) \supset \forall \bar{x}.(P(\bar{x}) \supset P'(\bar{x})).$$

Informally, this formula states that the only tuples that satisfy P are those that have to when ψ is true. The first conjunct of the antecedent expresses the assumption that P' plays the role of P in ψ and the second conjunct that the set of elements that satisfy P' is a subset of the set of elements satisfying P. The consequent expresses that the elements satisfying P are a subset of those satisfying P', i.e., that the assumption implies that P and P' are equivalent predicates.

The formal meaning of a circumscribed theory is based on the notion of minimal entailment. A first-order sentence ψ is *minimally entailed* by a formula ϕ if ψ is true in all minimal models of ϕ. This idea is formalized in definitions 2.25 and 2.26.

Definition 2.25 *Let M_1 and M_2 be two models of a first-order sentence ϕ. M_1 is a submodel of M_2, in P, denoted as $M_1 \leq_P M_2$, if M_1 and M_2 have the same domain, all predicate symbols in ϕ except for P have the same extensions in M_1 and M_2 and the extension of P in M is a subset of its extension in M_2. A model M of ϕ is* minimal *in P if $M' \leq_P M$ only if $M' = M$.*

Example 2.5 *Consider again the formula*

$$\phi \equiv \forall x.Tree(x) \wedge \neg Ab(x) \supset Green(x).$$

Suppose M_1 and M_2 are two interpretations that have the same domain and interpret the predicates as follows $Tree^{M_1} = Tree^{M_2} = \{pine\}$, $Green^{M_1} = Green^{M_2} = \{pine\}$, $Ab^{M_1} = \emptyset$ and $Ab^{M_2} = \{pine\}$. In this case M_1 is a submodel of the model M_2, since $Ab^{M_1} \supset Ab^{M_2}$ and all the other predicates are interpreted identically. Moreover, the model M_1 is minimal in Ab.

Definition 2.26 *A first-order sentence ϕ minimally entails a formula ψ with respect to a predicate P iff ψ is true in all models of ϕ that are minimal in P. This is denoted by $\phi \models_P \psi$.*

Example 2.6 *Consider the formula ϕ of example 2.5 in conjunction with the atom $Tree(pine)$. This is,*

$$\delta \equiv Tree(pine) \wedge (\forall x.Tree(x) \wedge \neg Ab(x) \supset Green(x)).$$

The atom $Green(pine)$ is minimally entailed by δ with respect to Ab.

2.6.2 Logic Programming

The central idea behind logic programming (Lloyd 1987) is to create a formal specification about a certain domain and to use an automated system to draw conclusions from the specification. The logic programmer describes the domain by means of declarative statements called rules.

Definition 2.27 *A rule is a formula of the form*

$$P(t_1, \ldots, t_n) \leftarrow \phi$$

where p is a user defined predicate, $t_1, \ldots, t_n$ are terms and ϕ is a formula.

The atom $P(t_1, \ldots, t_n)$ is called the *head* of the rule and the formula ϕ the *body*. If the body of the rule is empty, the implication arrow is omitted, and the head is called a *fact*. A *logic program* is a finite set of rules. A *definite logic program* is a logic program in which all the rules are positive formulas. If the body contains negative literals, it is called a normal rule. A program is called *normal* if it contains normal rules. Function-free logic programs are called Datalog programs (Ullman 1988).

Definition 2.28 *Let P be a definite logic program based on a first-order vocabulary Σ. The* semantics *of P corresponds to the least Herbrand model of P.*

Example 2.7 *Consider the following logic program P:*

$$\begin{aligned} &Vegetable(x) \leftarrow Tree(x). \\ &Tree(Pine). \\ &Vegetable(Rafflesia). \end{aligned}$$

and the Herbrand interpretations I_P^1 and I_P^2:

$$\begin{aligned} I_P^1 =&\{Vegetable(Rafflesia), \\ &\quad Tree(Pine), Vegetable(Pine)\} \\ I_P^2 =&\{Vegetable(Rafflesia), Tree(Rafflesia), \\ &\quad Tree(Pine), Vegetable(Pine)\} \end{aligned}$$

I_P^1 corresponds to the least Herbrand model of this program. The Herbrand interpretation I_P^2, although a Herbrand model of P (if P is regarded as a first-order theory with "$\leftarrow$" as implication), is not minimal *and so it is not a model of P.*

The least Herbrand model of a definitive logic program can be constructed by iterating over the *immediate consequence operator*:

Definition 2.29 *Let P be a definitive logic program based on vocabulary Σ. The immediate consequence operator T_p is a function on Herbrand interpretations of P such that:*

$$T_p(I) = \{A_0 \mid A_0 \leftarrow A_1, \ldots, A_m \in ground(P) \wedge \{A_1, \ldots, A_m\} \subseteq I\},$$

where $ground(P)$ corresponds to the set of all ground instances of clauses in P.

The sequence of interpretations constructed from P is usually denoted as follows:

$$\begin{aligned} T_P \uparrow 0 &= \emptyset \\ T_P \uparrow (i+1) &= T_P(T_{Pi} \uparrow i) \\ T_P \uparrow \omega &= \bigcup_{i=0}^{\infty} T_P \uparrow i \end{aligned}$$

As expected, the minimal Herbrand model of a definitive program P and $T_P \uparrow \omega$ coincide:

Theorem 3 *(van Emden and Kowalski 1976) Let P be a definitive logic program based on first-order vocabulary Σ. Let M_P be the least Herbrand interpretation of P. Then:*

- *M_P is the least Herbrand interpretation such that $T_P \uparrow \omega(M_P) = M_P$.*
- *$T_P \uparrow \omega = M_P$.*

For logic programs that contain negation in the bodies of the rules, the least Herbrand model not always exists. This fact has motivated the introduction of several semantics that try to capture the intuitions behind logic programs with negation (called normal programs). We briefly recall the one that will be used in connection with locally complete databases: the *stable model semantics*.

2.6.2.1 Stable model semantics

Definition 2.30 *(Gelfond and Lifschitz 1988) Let P be a normal logic program based on vocabulary Σ. Assume that each rule of the program is grounded. From any set M of atoms from P, we denote by P_M the logic program resulting from the following transformation.*

- *Delete each rule that has a negative literal $\neg R$ in its body with $R \in M$;*
- *Delete the negative literals in the bodies of the remaining rules.*

Since P_M is a definite logic program it has a least Herbrand model. If the Herbrand model of P_M coincides with M, then M is called a stable set.

Since any stable set of a program P is a *minimal Herbrand model* of P, stable sets are also called *stable models* of P. The *Gelfond-Lifschitz* operator $GL_P(I)$ is defined as $lfp(T_{P_I}(\emptyset))$. An interpretation I is a *stable model* for the propositional program P if and only if it is a fixpoint of $GL_P(I)$.

2.6.2.2 Well-founded semantics

The well-founded semantics (Van Gelder et al. 1991) is a three-valued semantics. The operator $A_P(I)$ is defined as the operator $GL_P(GL_P(I))$. The well-founded model is defined as the interpretation assigning **t** to the atoms that are true in the under-estimate $lfp(A_P(\emptyset))$, **f** to the atoms that are false in the over-estimate $gfp(A_P(\emptyset))$ and undefined to all other atoms. The following two theorems are well-known:

Theorem 4 ((Van Gelder et al. 1991)) *Each logic program has a unique well-founded model.*

Theorem 5 ((Van Gelder et al. 1991)) *The well-founded model is two valued if and only if there exists a unique stable model.*

2.6.3 ID-Logic

We introduce ID-Logic in connection with the ID-Logic-based query answering implementation of the local closed-world assumption that we present in Chapter 5.

ID-Logic is an extension of first-order logic with definitions. A general ID-Logic theory consists of a set of first-order sentences, called assertional knowledge, and a set of definitions, called definitional knowledge.

Example 2.8 *Consider the following ID-Logic theory:*

$$\Delta = \left\{ \begin{array}{ll} Ancestor(x,y) & \leftarrow Parent(x,y) \\ Ancestor(x,y) & \leftarrow \exists z Ancestor(x,z) \wedge Ancestor(z,y) \end{array} \right\}$$

$$\mathcal{A} = \{\forall x \neg Ancestor(x,x)\}.$$

The first two rules in Δ correspond to the definition of the relation ancestor in terms of the relations Parent and Ancestor itself. The declarative reading of the definition of Ancestor is as follows: a person x is an ancestor of a person y

iff either x is a parent of y or there exists a person z such that x is an ancestor of z and z is an ancestor of y.

The second component of an ID-Logic theory is the assertional knowledge, that in this example corresponds to $\mathcal{A}$. In general, $\mathcal{A}$ corresponds to a standard first-order set of sentences expressing properties of the intended domain. Here, the only sentence in $\mathcal{A}$ expresses that no person is ancestor of him/herself.

Definition 2.31 *Let Σ be a first-order vocabulary. A definition D is a set of rules:*

$$R(t_1, \ldots, t_n) \leftarrow \phi,$$

where $R(t_1, \ldots, t_n)$ is an atom and ϕ a first-order formula. An ID-Logic theory *Γ based on Σ is a pair $(\Delta, \mathcal{A})$, where Δ is a set of definitions and $\mathcal{A}$ is a set of first-order sentences.*

The *defined predicates* of a definition D, denoted by $Defined(D)$, correspond to predicates occurring in the head of a rule. All the other predicates are called *open predicates*, and are denoted by $Open(D)$. In Example 2.8, the sets of defined and open predicates are respectively $Defined(D) = \{Ancestor\}$ and $Open(D) = \{Parent\}$.

The semantics of ID-Logic hinges upon the semantics of definitions.

Definition 2.32 *An interpretation I is a* model of a definition *D iff there exists an interpretation I_0 of open(D) such that I is the two valued well-founded model D extending I_0. By extension, M is a model of a set of definitions $\Delta = \{D_1, \ldots, D_n\}$ iff M is a model of each definition $D \in \Delta$.*

The semantics of definitions is now extended to full ID-Logic theories.

Definition 2.33 *An interpretation I is a model of an ID-Logic theory $\Gamma = (\Delta, \mathcal{A})$ iff M is a model of Δ and satisfies all formulas of $\mathcal{A}$.*

As in the case of first-order logic, an ID-Logic theory may have none, one or many models. In such cases we say the theory is *inconsistent*, *complete*, or *incomplete*, respectively. The notions of satisfiability and entailment of a formula with respect to an ID-Logic theory are defined in the same fashion as their first-order counterparts.

Definition 2.34 *Let Γ be an ID-Logic theory based on vocabulary Σ, and ψ a closed first-order formula. We say ψ is a* logical consequence *of Γ, denoted by $\Gamma \models \psi$, iff every model I of Γ satisfies also ψ.*

Example 2.9 *Consider again the ID-Logic theory of example 2.8. Assume the following two facts are added to the assertional knowledge*

$$\{Parent(Francisco, Teresa), Parent(Teresa, Gonzalo)\}.$$

The following ID-Logic interpretation I is a model of this theory:

$$Dom(I) = \{Francisco, Teresa, Gonzalo\},$$

$$\begin{aligned} Francisco^I &= Francisco \\ Teresa^I &= Teresa \\ Gonzalo^I &= Gonzalo \end{aligned}$$

$$\begin{aligned} Parent^I = \{&(Francisco, Teresa), \\ &(Teresa, Gonzalo)\} \\ Ancestor^I = \{&(Francisco, Teresa), \\ &(Teresa, Gonzalo), \\ &(Francisco, Gonzalo)\}. \end{aligned}$$

Additionally, it is not difficult to see that $Ancestor(Francisco, Gonzalo)$ is a logical consequence of Γ.

In this text, we will use the symbol $\models$ to denote logical consequence in the propositional, first-order and ID-Logic case. This syntactic overloading poses no problem since propositional logic is generalized by first first-order logic, which in turn is generalized by ID-Logic.

2.7 Bibliographical Note

The material concerning set theory, enumerability, order and lattices has been composed and adapted from the excellent introductory books by Enderton (Enderton 1972), Boolos et al. (Boolos and Jeffrey 1999) and Davey et al. (Davey and Priestley 2002). In this introductory text, we have only presented the concepts of each of these topics that are relevant for the development of this thesis; the reader interested in additional insights is invited to check the original sources.

An excellent introductory book on the theory of computational complexity is the book by Garey et al. (Garey and Johnson 1979).

The literature in propositional and first-order logic is vast and varied, ranging from the philosophical to the purely technical presentation. In this text we have tried to find a compromise between both approaches based on the treatment of formal logic presented by Enderton (Enderton 1972) and Bertossi (Bertossi 1996). Additional good references in mathematical and philosophical logic are (Mendelson 1972) and (Boolos and Jeffrey 1999).

A very good introduction to knowledge representation in general and to nonmonotonic reasoning in particular is the textbook by Brachman et al.

(Brachman and Levesque 2004). The general discussion on nonmonotonic reasoning is based on this book and on Reiter's critical discussion of several formalisms for nonmonotonic reasoning (Reiter 1987). For the text on circumscription we have relied on the original paper by McCarthy (McCarthy 1980; McCarthy 1986) complemented with Lifschitz' review on the topic (Lifschitz 1994). The standard reference on logic programming is (Lloyd 1987). For this chapter we have also relied on the intuitive treatment of Nilsson et al. (Nilsson and Maluszyński 2000). The section on stable models is adapted from (Gelfond and Lifschitz 1988). For the introduction of ID-Logic we have based the treatment on (Denecker 2000), (Denecker et al. 2001) and (Denecker and Ternovska 2007).

Chapter 3

The Local Closed-World Assumption

3.1 Introduction

In standard database systems, a relational database can be viewed as a collection of tuples that stores information about a certain domain in the real world. As we have already noted in the introduction, it is not uncommon that the information stored in the database is not complete with respect to its intended domain. The reasons for this fact may be diverse: ignorance about the domain, lack of proper maintenance, incomplete migration, accidental deletion of tuples, etc. In other contexts, incompleteness is intrinsic in the nature of the database. A typical example of this are the so called mediator-based (or data integration) systems (Levy et al. 1996; Grahne and Mendelzon 1999a; Grahne 2002; Lenzerini 2002; Van Nuffelen et al. 2004) and peer-to-peer systems (Halevy et al. 2003; Franconi et al. 2003; Calvanese et al. 2004; Bertossi and Bravo 2004), where the information is distributed over several independent databases; each one of them usually has only partial knowledge about the domain of discourse. Yet, it is often the case that a particular database (or a data source) is an expert about a specific part of the overall domain and has complete knowledge about it.

- In Example 1.3, for instance, the database stores all the information about publications of 'Cortés' before 2006, but it is incomplete after that year.
- In Example 1.5, the database is not aware of *every* collaborator of the computer science department, but it has complete knowledge about the members of the computer science department under consideration.
- In Example 1.6, a certain source may have full knowledge about the car

owners in a specific county, although the information may not be complete with respect to other counties.

In order to draw correct conclusions from a given database system, this kind of 'meta information' should be explicitly represented in the system, and its meaning should be well-defined. Below, we formalize this idea. In particular, we extend standard relational database theory in order to accommodate locally closed knowledge. In the next section we discuss our assumptions on relational databases.

3.2 Relational Databases

In what follows we assume the existence of a function-free database vocabulary Σ consisting of a finite set of predicate symbols $\mathcal{R}(\Sigma)$, and an infinite set $\mathcal{C}(\Sigma)$ of constants to represent potential objects of the domain of discourse.

Definition 3.1 (Database instance) *A* database instance D *over* Σ *is a Herbrand* σ_D*-structure, for some vocabulary* $\sigma_D \subseteq \Sigma$ *with* $\mathcal{R}(\sigma_D) = \mathcal{R}(\Sigma)$ *and finite domain* $Dom^D = \mathcal{C}(\sigma_D) \subseteq \mathcal{C}(\Sigma)$.

A database instance D interprets only a finite set of constants of Σ. The set Dom^D is called the *active domain* of the database instance and contains at least all constants in the tables of D (and often only those). For some variable-free atomic formula A, we write $A \in D$ to denote that $A = P(\bar{d})$ where $\bar{d} \in P^D$.

A database instance D is completely determined by the pair $(Dom^D, \{A \in HB(\sigma_D) \mid A \in D\})$. Thus, in the examples below, we will often specify a database instance D by a domain and a set of atoms. If the domain of D is not explicitly mentioned, it consists of the set of constants that appear in these atoms.

Example 3.1 *The database of Example 1.5 consists of relations* $Tel(\cdot,\cdot)$ *(between people and their telephone number) and* $Dept(\cdot,\cdot)$ *(between people and the department they belong to). It can be abbreviated as follows:*

$$D = \left\{ \begin{array}{ll} Tel(LD, 6531421), & Dept(BD, CS), \\ Tel(LD, 0923314), & Dept(LD, Phil), \\ Tel(BD, 5985625), & Dept(TD, CS), \\ Tel(TD, 5845213), & Dept(DF, Bio) \end{array} \right\}$$

Thus,

$$Tel^D = \left\{ \begin{array}{l} (LD, 6531421), (LD, 0923314) \\ (BD, 5985625), (TD, 5845213) \end{array} \right\}$$

$$Dept^D = \{(BD, CS), (LD, Phil), (TD, CS), (DF, Bio)\}$$

and

$$Dom^D = \left\{ \begin{array}{l} LD, BD, TD, CS, Phil, DF, Bio, \\ 6531421, 0923314, 5985625, 5845213 \end{array} \right\}$$

As in standard databases, the order in which the atoms appear in D is irrelevant. Observe also that if we add, for instance, the constant BP (Bart Peeters) to Dom^D, then even if BP does not appear in D, the resultant instance is a different *database than the original one.*

Structures are a convenient way to formalize the *content* of a database: the collection of named objects and tables of the database. Structures are also the standard way to formalize *semantics*, as they serve as a formal description of a *possible state* of the *domain of discourse*. When interpreted in this way, a database instance, which is a structure, represents the *unique* possible state of the real world, the domain of discourse. A database instance thus represents *complete knowledge* of the domain, specifying what objects exist in the domain of discourse and in what relations they occur. In this thesis, we will consider incomplete databases representing *partial knowledge* on the domain of discourse. A database instance will still formalize the content of the database, but cannot any longer be viewed as the (unique) possible state of the world.

An alternative formalization of a database is as a first-order theory consisting of ground atoms, augmented with the *closed-world assumption* (Reiter 1978). This view can be formalized in first-order logic. Before recalling this theory, we first introduce a convenient notation.

Notation 1 *Let D be a database instance. For any $P \in \mathcal{R}(\Sigma)$ and tuple $\bar{t}$ of terms, the formula $\bigvee_{\bar{a} \in P^D} (\bar{t} = \bar{a})$ expresses that $\bar{t}$ belongs to P^D. This formula is denoted by $P(\bar{t}) \in D$.*

In Example 3.1, for instance, the expression $Dept(x, y) \in D$ is a shorthand for the formula

$$\begin{array}{c} \big((x = BD) \wedge (y = CS)\big) \vee \big((x = LD) \wedge (y = Phil)\big) \vee \\ \big((x = TD) \wedge (y = CS)\big) \vee \big((x = DF) \wedge (y = Bio)\big). \end{array}$$

Definition 3.2 (Semantics of a database instance) *(Reiter 1982) The* logical semantics *of a database instance D is a set $\mathcal{M}_{st}(D)$ of formulas, consisting of the following sentences:*

Soundness*:*

$$\bigwedge_{A \in D} A$$

Completeness*:*

$$\bigwedge_{P \in \mathcal{R}(\Sigma)} \forall \overline{x} \left(P(\overline{x}) \supset (P(\overline{x}) \in D) \right)$$

Domain Closure Axiom *(*$\mathsf{DCA}(Dom^D)$*):*

$$\forall x \, (\bigvee_{C \in Dom^D} x = C)$$

Unique Name Axiom *(*$\mathsf{UNA}(Dom^D)$*):*

$$\bigwedge_{C \neq C' \in Dom^D} C \neq C'$$

In the following we discuss each of these axioms:

- **Soundness.** This axiom formalizes the assumption that every atom that is in the database instance is true in the domain of discourse. It formally captures the natural language statements 'It is true that $A_1 \in D$', *and* 'It is true that $A_2 \in D$', and so on.
- **Completeness.** This axiom is constructed by the conjunction of formulas of the type $\forall \overline{x} \, \big(P(\overline{x}) \supset (P(\overline{x}) \in D)\big)$. This formula states that for every tuple $\overline{x}$ of the domain of discourse, if $P(\overline{x})$ is true, then $P(\overline{x})$ is one of the atoms of the database. In other words, the database stores all the tuples that are true in the domain of discourse.
- **Domain Closure Axiom.** The argument of this axiom is the active domain of the database. It states that the elements of the domain Dom^D of the database are all the objects in the domain of discourse. This is, then, a completeness axiom for the elements of the domain of discourse.
- **Unique Name Axiom.** This axiom states that every element in the active domain of the database denotes a different object in the domain of discourse. I.e. each element in the domain of discourse has exactly one name.

Example 3.2 *Consider the database instance D of Example 3.1. The following formulas correspond to the axioms of the logical semantics of D:*
Soundness axiom Δ_1:

$$\begin{array}{l} Tel(LD, 6531421) \wedge Tel(LD, 0923314) \wedge \\ Tel(BD, 5985625) \wedge Tel(TD, 5845213) \wedge \\ Dept(BD, CS) \wedge Dept(LD, Phil) \wedge \\ Dept(TD, CS) \wedge Dept(DF, Bio) \end{array}$$

Completeness axiom Δ_2:

$$\forall x, y \big(Tel(x, y) \supset (Tel(x, y) \in D)\big) \bigwedge \forall x, y \big(Dept(x, y) \supset (Dept(x, y) \in D)\big)$$

Domain closure axiom Δ_3:

$$\begin{aligned}\forall x \quad & (x = LD) \vee (x = BD) \vee (x = TD) \vee (x = CS) \vee \\ & (x = Phil) \vee (x = DF) \vee (x = Bio) \vee (x = 6531421) \vee \\ & (x = 0923314) \vee (x = 5985625) \vee (x = 5845213)\end{aligned}$$

Unique name axiom Δ_4:

$$\begin{aligned}& LD \neq BD \wedge LD \neq BD \wedge \cdots \wedge LD \neq 5845213 \wedge \\ & BD \neq TD \wedge BD \neq CS \wedge \cdots \wedge BD \neq 5845213 \wedge \\ & \vdots \\ & \cdots \wedge 0923314 \neq 5845213 \wedge 5985625 \neq 5845213\end{aligned}$$

and $\mathcal{M}_{st}(D) = \{\Delta_1, \Delta_2, \Delta_3, \Delta_4\}$.

The first-order formalization of the CWA consists of the Soundness and Completeness axioms, the Domain Closure Axiom and the Unique Name Axiom.

Viewing a database as a structure or as a theory under the CWA is equivalent, as follows from the following well-known proposition:

Proposition 4 *Each model of $\mathcal{M}_{st}(D)$ is isomorphic to D.*

In this thesis, a *query* is a first-order formula $\mathcal{Q}[\bar{x}]$ with free variables among $\bar{x}$. We will always assume that all constants in this query are contained in the domain Dom^D of the database instance (and hence the need to make explicit the domain of the database). It follows from the above proposition that for each query $\mathcal{Q}[\bar{x}]$ and ground terms $\bar{t}$,

$$D \models \mathcal{Q}[\bar{t}] \text{ iff } \mathcal{M}_{st}(D) \models \mathcal{Q}[\bar{t}]. \tag{3.1}$$

In Section 3.3, we study databases that have complete knowledge on the (finite) domain, but not on the predicate tables. In Section 3.7, we come back to this issue and discuss finite, domain independent queries in the context of locally closed databases.

3.3 Local Closed-World Assumptions

A local closed-world assumption expression specifies formally for which elements in the real world a database table contains all the tuples. Any such expression should then contain the following two components:

(a) A formal specification circumscribing where in the real world the database predicate is complete.

(b) The specific predicate of the database that contains all the information for the elements described in (a).

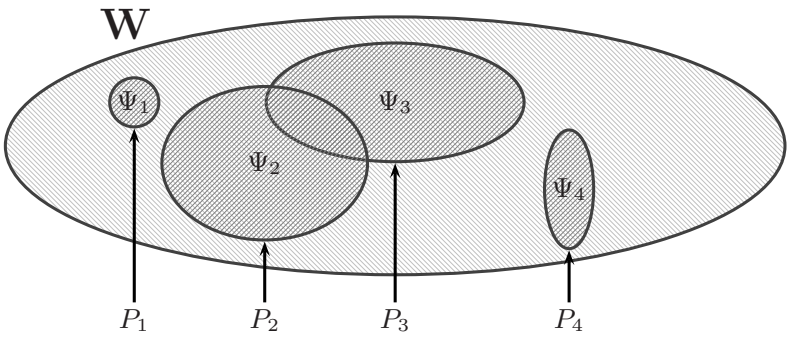


Figure 3.1: W represents the domain of discourse of the database. Areas specified by $\Psi_1, \ldots, \Psi_4$ represent the regions in the real world where the database predicates $P_1, \ldots P_4$ are complete.

For the specification (a), since we seek a specification about *real world* elements, we use a first-order expression $\Psi[\bar{x}]$. For (b), we name the database predicate that is complete for the domain elements described by $\Psi[\bar{x}]$. Figure 3.1 illustrates the situation.

With this intuition at hand we now introduce the formal syntax and semantics of local closed-world assumptions.

Definition 3.3 (LCWA) *A* local closed-world assumption *(LCWA) is an expression of the form*

$$\mathcal{LCWA}(P(\bar{x}), \Psi[\bar{x}]),$$

where $P \in \mathcal{R}(\Sigma)$ is a predicate symbol, called the LCWAs object *and $\Psi[\bar{x}]$, called the LCWAs* window of expertise*, is a first-order formula over Σ with free variables among $\bar{x}$.*

The intuitive reading of the expression in Definition 3.3 is the following: "for all objects $\bar{x}$ such that $\Psi[\bar{x}]$ holds in the *real world*, if an atom of the form $P(\bar{x})$ is true in the real world, then $P(\bar{x})$ occurs *in the database*". Note that in $P(\bar{x})$ the value of the variables $\bar{x}$ are constrained by Ψ. For this reason we call Ψ a *window of expertise* of the predicate P.

Example 3.3 *Some local closed-world assumptions for the database of Example 3.1 are the following:*

1. *The expression* $\mathcal{LCWA}(Dept(x,d)), d = CS)$ *states that the database contains all members of computer science. That is, for every member* m *of that department,* $Dept^D$ *contains the pair* (C_m, CS) *where* C_m *is the name of* m*.*

2. *The expression* $\mathcal{LCWA}(Tel(x,y), Dept(x,CS))$ *states that all telephone numbers of all members of the computer science department are known and occur in the database. That is, for every telephone number* n *owned by a member* m *of the computer science department, the atom* $Tel(C_m, n)$ *appears in the database.*

3. $\mathcal{LCWA}(Tel(x,y), x = LD)$ *expresses that* D *contains all telephone numbers of Lien Desmet.*

As the next example shows, windows of expertise may contain complex formulas, and might be closed.

Example 3.4 *Consider again the database schema of Example 1.6 and the following windows of expertise:*

1. *The expression:*

$$\mathcal{LCWA}(CarOwner(x,y,z), Location(x, Bronx))$$

states that the database predicate $CarOwner$ *is complete for people who live in Bronx.*

2. *The local closed-world assumption*

$$\mathcal{LCWA}(CarOwner(x,y,z), CarOwner(x, Volvo, z) \wedge \neg Location(x, Manh))$$

represents completeness about the Car Owners and the car ID of people that own a Volvo and that are not residents of Manhattan $(Manh)$*.*

3. *The expression:*

$$\mathcal{LCWA}(CarOwner(x,y,z), \neg\exists x CarOwner(x, FerrariF40, z))$$

states that the database is complete for those who do not own a $FerrariF40$*. In particular, if no one owns a* $FerrariF40$ *(a rare, exclusive model), then the database predicate CarOwner is complete.*

A local closed-world assumption expression states a logical relationship between a database table and facts of the real world, and hence, its meaning necessarily depends on the database. In the context of a given database, this relationship can be expressed by a first-order formula.

Definition 3.4 (Semantics of a LCWA) *Let $\theta = \mathcal{LCWA}(P(\bar{x}), \Psi[\bar{x}])$ be a local closed-world assumption and D a database over vocabulary Σ. The logical semantics of θ in D is the formula*

$$\mathcal{M}_D(\theta) = \forall\bar{x}\Big(\Psi[\bar{x}] \supset \big(P(\bar{x}) \supset (P(\bar{x}) \in D)\big)\Big).$$

A σ_D-structure M is a model of θ iff M is a model of $\mathcal{M}_D(\theta)$.

Observe that the logical semantics of a local closed-world assumption contains the subformula $P(\bar{x}) \in D$ that depends on the database instance. We cannot characterize its meaning in a way that is independent of the content of the database. Stated differently, changing the database instance modifies the logical semantics of a local closed-world assumption. It follows that a local closed-world assumption is a *non-monotonic* construct.

Example 3.5 *Given the expression $\mathcal{LCWA}(Tel(x, y), x = LD)$, the database of Example 3.1 implies that* 016311560 *is not one of the telephone numbers of Lien Desmet, whereas the updated database obtained by adding $Tel(LD,$* 016311560*), implies that* 016311560 *is a telephone number of Lien Desmet.*

Example 3.6 *Applying Definition 3.4 on the LCWA of item (2) of Example 3.3, one obtains*

$$\begin{aligned}\mathcal{M}_D(\theta) = \forall x\forall y\big(&Dept(x, CS) \supset \big(Tel(x, y) \supset \\ &\big((x = LD \wedge y = 6531421) \vee (x = LD \wedge y = 0923314) \vee \\ &(x = BD \wedge y = 5985625) \vee (x = TD \wedge y = 5845213)\big)\big)\big).\end{aligned}$$

Because $Dept(LD, CS)$ does not hold, this simplifies to the following formula.

$$\begin{aligned}\forall x\forall y\big(Dept(x, CS) \supset (Tel(x, y) \supset &(x = BD \wedge y = 5985625) \vee \\ &(x = TD \wedge y = 5845213))\big)\end{aligned}$$

Observe the similarity in the structure of the meaning of local closed-world assumption and the completeness axiom of definition 3.2. The difference here is that the new antecedent Ψ is not present in the complete database. Or in other words, the formula Ψ is always true. It is then possible to represent complete knowledge on a predicate using a LCWA expression. Also, it is possible to represent the somehow redundant statement that a predicate is incomplete. These are the two extreme cases of local closed-world assumptions, and they are represented as follows:

- LCWA with window of expertise that contains all tuples of the domain:

$$\mathcal{LCWA}(P(\bar{x}), \mathbf{t}).$$

This LCWA expresses that when $P(\bar{x})$ is true in the real world, then it belongs to the database. In other words, this LCWA expresses that D has complete knowledge on P. Indeed,

$$\forall\bar{x}\Big(\mathbf{t} \supset \big(P(\bar{x}) \supset (P(\bar{x}) \in D)\big)\Big) \equiv \forall\bar{x}(P(\bar{x}) \supset (P(\bar{x}) \in D),$$

which is equivalent to the completeness axioms of definition 3.2.

- LCWA with empty window of expertise:

$$\mathcal{LCWA}(P(\bar{x}), \mathbf{f}).$$

This LCWA does not express any closure. In fact, $\mathcal{M}_D(\mathcal{LCWA}(P(\bar{x}), \mathbf{f}))$ is tautologically true.

A useful modularity property of the local closed-world assumption is that a finite set of LCWA expressions for the same object predicate can be composed into one (disjunctive) LCWA expression for the same predicate. Conversely, one may split a LCWA with a disjunctive window of expertise over the disjunction and preserve equivalence. This is expressed in the following proposition.

Proposition 5 *Consider the following LCWA expressions:*

$$\theta = \mathcal{LCWA}(P(\bar{x}), \bigvee_{i=1}^{n} \Psi_i[\bar{x}_i]) \text{ and}$$
$$\theta_i = \mathcal{LCWA}(P(\bar{x}), \Psi_i[\bar{x}_i]),$$

$i = 1, \ldots, n$. *Then* $\{\theta\}$ *and* $\{\theta_1, \ldots, \theta_n\}$ *are equivalent in every database* D, *i.e.,* $\mathcal{M}_D(\theta)$ *and* $\bigwedge_{i=1}^{n} \mathcal{M}_D(\theta_i)$ *are equivalent for every* D.

Proof: Indeed,

$$\bigwedge_{j=1}^{n} \mathcal{M}_D(\theta_j) \equiv \bigwedge_{j=1}^{n} \forall\bar{x}\Big(\Psi_j[\bar{x}] \supset \big(P(\bar{x}) \supset \big(P(\bar{x}) \in P^D\big)\big)\Big)$$
$$\equiv \forall\bar{x}\Big(\bigvee_{j=1}^{n} \Psi_j[\bar{x}] \supset \big(P(\bar{x}) \supset \big(P(\bar{x}) \in P^D\big)\big)\Big) \equiv \mathcal{M}_D(\theta).$$

□

Example 3.7 *Consider again the expressions:*

$$\mathcal{LCWA}(CarOwner(x, y, z), Location(x, Bronx))$$

and

$$\mathcal{LCWA}(CarOwner(x, y, z), CarOwner(x, Volvo, z) \wedge \neg Location(x, Manh))$$

With the help of proposition 5 we can combine them in a single local closed-world assumption:

$$\mathcal{LCWA}(CarOwner(x,y,z), Location(x,Bronx) \vee (CarOwner(x,Volvo,z) \wedge \neg Location(x,Manh)))$$

The new expression states that the database table CarOwner stores all tuples for residents of the Bronx or for people not living in Manhattan and that own a Volvo.

In the rest of this thesis we will assume without a loss of generality that each predicate symbol in $\mathcal{R}(\Sigma)$ is the object of exactly one LCWA expression. For predicates P in which no closure is expressed at all, the expression $\mathcal{LCWA}(P(\bar{x}), \mathbf{f})$ will be assumed. For a predicate P, Ψ_P will denote the windows of expertise of P in this combined LCWA. In the last example, for instance,

$$\Psi_{CarOwner} = Location(x,Bronx) \vee (CarOwner(x,Volvo,z) \wedge \neg Location(x,Manh))$$

3.3.1 Locally Closed Databases

In a locally closed database we need to drop the completeness axiom, and replace it by a new axiom that reflects the real state of affairs. Such a database consists of a database instance augmented with a finite set of local closed-world assumptions, one for each predicate in $\mathcal{R}(\Sigma)$.

Definition 3.5 (locally closed database) *A* locally closed database $\mathfrak{D}$ *over* Σ *is a pair* $(D, \mathcal{L})$ *of a database instance* D *over* Σ *and a finite set* $\mathcal{L}$ *of local closed-world assumptions such that* Dom^D*, the domain of* D*, contains all constants in* $\mathcal{L}$*. We denote* $\sigma_{\mathfrak{D}} = \sigma_D$ *and* $Dom^{\mathfrak{D}} = Dom^D$*.*

Observe that in this definition the constants appearing in the local closed-world assumption do not necessarily appear in the database, hence the need of a database vocabulary extending the constants in the database, as defined in 3.1. This extension is motivated by the need of expressing closure expressions over domain elements that not necessarily appear in the database instance. For example, in the third item of Example 1.6 in the expression

$$\mathcal{LCWA}(CarOwner(x,y,z), \neg\exists x CarOwner(x, FerrariF40, z)),$$

the object of the assumption includes the constants $FerrariF40$, which is not present in the original database of car owners.

In the following definition we introduce the important class of predicates called *base*. In base predicates every tuple in the domain is known to be true

or false. I.e., in base predicates there exists no uncertainty about their truth value, and they can be treated as predicates of standard two-valued databases. Their importance will be made clear in the query answering section.

Definition 3.6 (Base predicates) *The* base predicates *of a locally closed database $\mathfrak{D} = (D, \mathcal{L})$ are $\mathbf{t}, \mathbf{f}$ and all predicates P such that $\mathcal{L}$ contains the local closed-world assumption $\mathcal{LCWA}(P(\bar{x}), \mathbf{t})$.*

We illustrate the intuition behind this definition with an example:

Example 3.8 *Consider the following LCWA expressions:*

- $\mathcal{LCWA}(Location(x, y), \mathbf{t})$ *and*
- $\mathcal{LCWA}(CarOwners(x, y, z), Location(x, y))$.

The predicate Location is base, since its window of expertise is the base predicate $\mathbf{t}$. Hence, we know that every tuple that is not in the database instance of Location should be false. The object of the second expression, although not a base predicate itself, depends on the base predicate Location. This, of course, does not entail complete knowledge on $CarOwners$, but it implies that the tuples of the windows of expertise of $CarOwners(x, y, z)$ can be determined solely from the data available in the table Location, and without any appeal to the external domain of discourse.

In the examples below, we will often use domain independent predicates such as $=$ and arithmetical relations $<, >, \leq, \geq$. For the sake of simplicity, we will assume here that these 'built-in' predicates are *base* predicates of the database. Thus, for each of these predicates, $\mathcal{L}$ contains the local closed-world assumption expressing complete knowledge about it, and D contains a (finite) database table representing the (projection of the) predicate in Dom^D. In particular, D contains the identity relation on Dom^D as a table for $=$, and the projections of the standard natural number relations onto the set of natural numbers in the domain of D. For instance, $<^D$ is $\{(n, m) \mid n, m \in Dom^D \cap \mathbb{N} \wedge n < m\}$.[1]

The semantics of a locally closed database $\mathfrak{D} = (D, \mathcal{L})$ can now be formalized in a way similar to the semantics of standard databases, as given in Definition 3.2. All we need to do is to replace the database completeness axioms by the logical semantics of the local closed-world axioms.

[1] This approach is not entirely complication-free. Certain queries may have unintended answers when solved with respect to these projected relations. For instance, the query $x < 100$ would return the numbers less than hundred that occur in Dom^D, which is potentially different than the numbers less than 100. In the next chapter we will investigate additional conditions on queries and Dom^D that guarantee correctness.

Definition 3.7 (The semantics of a locally closed database) *Let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database over Σ. A model M of $\mathfrak{D}$ is a Herbrand $\sigma_{\mathfrak{D}}$-structure satisfying each atom $A \in D$ and each $\theta \in \mathcal{L}$. The* logical semantics *of $\mathfrak{D}$ is a set $\mathcal{M}(\mathfrak{D})$ of formulas, consisting of the axioms for the built-in predicates and the following sentences:*

Soundness*:*	$\bigwedge_{A \in D} A$
Local Completeness*:*	$\bigwedge_{\theta \in \mathcal{L}} \mathcal{M}_D(\theta)$.
Domain Closure Axiom *(*$\mathsf{DCA}(Dom^{\mathfrak{D}})$*):*	$\forall x(\bigvee_{C \in Dom^{\mathfrak{D}}} x = C_i)$
Unique Name Axiom *(*$\mathsf{UNA}(Dom^{\mathfrak{D}})$*):*	$\bigwedge_{C \neq C' \in Dom^{\mathfrak{D}}} C \neq C'$

We denote by $M \models \mathfrak{D}$ that M is a model of $\mathfrak{D}$. If every model of $\mathfrak{D}$ is also a model of a formula φ, we say that $\mathfrak{D}$ *entails* φ (or φ *follows* from $\mathfrak{D}$), and denote this by $\mathfrak{D} \models \varphi$.

The theory $\mathcal{M}(\mathfrak{D})$ of $\mathfrak{D}$ expresses incomplete knowledge about the real world. Thus, in general, it has several (non-isomorphic) models (and the actual world corresponds to one of those models). Then, a locally closed database contains a standard database instance but this structure will not be the only model of the database.

We now discuss the basic semantical properties of locally closed databases. The first proposition shows that the model semantics and the logical semantics defined above coincide.

Proposition 6 *Each model of $\mathcal{M}(\mathfrak{D})$ is isomorphic to a model of $\mathfrak{D}$.*

Proof: By the fact that $\mathcal{M}(\mathfrak{D})$ contains $\mathsf{UNA}(Dom^{\mathfrak{D}})$ and $\mathsf{DCA}(Dom^{\mathfrak{D}})$. □

We therefore have the following generalization, for locally closed databases, of the formula (3.1) in Section 3.2:

$$\mathfrak{D} \models \mathcal{Q}[\bar{t}] \text{ iff } \mathcal{M}(\mathfrak{D}) \models \mathcal{Q}[\bar{t}]. \tag{3.2}$$

Since the number of Herbrand models of $\mathcal{M}(\mathfrak{D})$ is finite (due to a finite domain and the lack of function symbols) and all of them can be computed, query answering is decidable. Of course, the naive method of generating all models and computing answers to queries in all models, is impractical. In general, the number of models may be very high and is bounded by $2^{|HB(\sigma_{\mathfrak{D}})|}$. Thus, this number may be exponential in the size of $Dom^{\mathfrak{D}}$ and in the number of relation symbols of $\mathcal{R}(\Sigma)$, and double exponential in the maximal arity of the latter symbols. In the following sections, we will study the data complexity of query answering, and smarter query answering methods.

The next proposition shows that the database's semantics assumes the soundness of the database, i.e., that all atoms in D are true.

Proposition 7 *$\mathfrak{D}$ entails each atom in D.*

Proof: Obvious from the fact that each atom in $\mathfrak{D}$ appears as a conjunct in the soundness formula in $\mathcal{M}(\mathfrak{D})$. □

For standard databases, we had the basic property that the structure D was the only model of $\mathcal{M}_{st}(D)$. In the context of locally closed databases, this property is lost, but we retain the following weaker property:

Proposition 8 *A locally closed database $\mathfrak{D} = (D, \mathcal{L})$ is consistent, and D is its least model.*

Proof: Clearly, D satisfies $\mathfrak{D}$ and because each (Herbrand) model M of $\mathfrak{D}$ satisfies all atoms of D, $D \subseteq M$. □

Example 3.9 *Recall again the database and the three LCWAs of Example 3.3. These LCWAs can be rewritten as:*

$$\theta_1 = \mathcal{LCWA}(Dept(x, y), y = CS)$$
$$\theta_2 = \mathcal{LCWA}(Tel(x, y), Dept(x, CS) \vee x = LD)$$

Together with the database instance D they constitute the locally complete database $\mathfrak{D} = (D, \mathcal{L})$, where $\mathcal{L} = \{\theta_1, \theta_2\}$. Since new constants are not introduced by the LCWA expressions, soundness (Δ_1), domain closure (Δ_3) and unique names (Δ_4) axioms remain as in Example 3.2. However, since the database is now only locally closed, the completeness axiom (Δ_2) needs to be replaced by the following axiom:

$$\begin{array}{c} \forall x, y\big(y = CS \supset (Dept(x, y) \supset Dept(x, y) \in D)\big) \\ \bigwedge \\ \forall x, y\big((Dept(x, CS) \vee x = LD) \supset (Tel(x, y) \supset Tel(x, y) \in D)\big) \end{array}$$

The logical semantics of $\mathfrak{D}$ is the theory $\mathcal{M}(\mathfrak{D}) = \{\Delta_1, \Delta_2, \Delta_3, \Delta_4\}$. This theory is incomplete. Consider the query $\mathcal{Q} : Tel(DF, 4585625)$. Since David Finner (DF) is not from the computer science department nor he is Lien Desmet (where Tel is complete) and Tel does not store 4585625, the database does not know whether $Tel(DF, 4585625)$ is true or false, and hence it is not the case that $\mathcal{M}(\mathfrak{D}) \models \mathcal{Q}$. In other words, there is a model M_1 of $\mathcal{M}(\mathfrak{D})$ such that $Tel(DF, 4585625)^{M_1} = \mathbf{t}$ and a model M_2 (for instance the least model D) where $Tel(DF, 4585625)^{M_2} = \mathbf{f}$. On the other hand,

$$\mathcal{M}(\mathfrak{D}) \models \neg Tel(BD, 98374676).$$

3.3.2 Relations to CWA and OWA

Next, we show that our concept of locally closed databases is a generalization of both relational databases with the open-world assumption (OWA) (Abiteboul and Duschka 1998; Grahne 2002) and of relational databases with Reiter's closed-world assumption (CWA) (Reiter 1982).

Proposition 9 *A locally closed database $\mathfrak{D} = (D, \emptyset)$ corresponds to the database D under the OWA.*

Proof: In both cases the database is totally incomplete, consisting only of the elements in the database instance plus UNA and DCA. □

We will call this an *open* database. Equivalently, an open database can be represented by the locally closed database $(D, \{\mathcal{LCWA}(P(\bar{x}), \mathbf{f}) \mid P \in \mathcal{R}(\Sigma)\})$ containing the LCWA with the empty (false) window of expertise for each predicate in $\mathcal{R}(\Sigma)$.

Proposition 10 *For a database instance D, let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database in which $\mathcal{L} = \{\mathcal{LCWA}(P(\bar{x}), \mathbf{t}) \mid P \in \mathcal{R}(D)\}$. Then $\mathcal{M}_{st}(D)$ and $\mathcal{M}(\mathfrak{D})$ are equivalent.*

Proof: By Definitions 3.2 and 3.7, as for each predicate P occurring in D, $\mathcal{M}_D(\mathcal{LCWA}(P(\overline{x}), \mathbf{t}))$ is equivalent to the database completeness assumption regarding P. □

It follows from propositions 9 and 10, that the LCWA allows one to express any database, from totally incomplete to totally complete.

3.4 Further Representation Considerations Regarding the LCWA

Before turning to the computational aspects of reasoning with locally closed databases, we include in this section some further considerations about the way the LCWA is represented and some extensions to more general contexts.

3.4.1 LCWA with Multiple Objects

Typically, expertise in some area is not limited to one predicate. For instance, databases like that of Example 1.5 are often complete with respect to several types of information about the CS staff (e.g., teaching duties, addresses, etc.) and not only with respect to their telephone numbers. For this, one could extend the basic notion of a LCWA expression to allow sets of objects, that is

$$\theta = \mathcal{LCWA}(\{P_1(\bar{x}_1), \ldots, P_n(\bar{x}_n)\},\ \Psi[\bar{x}]),$$

where the $P_i \in \mathcal{R}(\Sigma)$ are predicate symbols (the LCWA objects) and $\Psi[\bar{x}]$ is a first-order formula over Σ with free variables $\bar{y}$ such that $\bar{y} \subseteq \bar{x} = \bigcup_{i=1}^{n} \bar{x}_i$. When a LCWA expression takes this form, the meaning of θ under a database D is extended as follows:

$$\mathcal{M}_D(\theta) = \forall \bar{x} \Big(\Psi[\bar{x}] \supset \big(\bigwedge_{i=1}^{n} \big(P_i(\bar{x}_i) \supset \big(P_i(\bar{x}_i) \in P_i^D \big) \big) \big) \Big).$$

Notice that such a combined LCWA may contain several object atoms $P(\bar{x}_1)$, $\ldots, P(\bar{x}_n)$ with the same predicate P but different tuples of variables $\bar{x}_i$. In Section 3.4.2 below we consider such a case.

In the rest of this thesis we assume only one predicate object for each LCWA expression. As the following proposition shows, this assumption does not harm the generality:

In the following, given a formula Ψ, we will denote by $\exists|_{\bar{x}}\Psi$ the existential quantification of all free variables in Ψ, except of those in $\bar{x}$.

Proposition 11 *Let* $\theta = \mathcal{LCWA}(\{P_1(\bar{x}_1), \ldots, P_n(\bar{x}_n)\}, \Psi)$ *and*

$$\theta_i = \mathcal{LCWA}(P_i(\bar{x}_i), \exists|_{\bar{x}_i}\Psi),$$

$i = 1, \ldots n$. *Then* $\{\theta\}$ *and* $\{\theta_1, \ldots, \theta_n\}$ *are logically equivalent in every database* D, *i.e.,* $\mathcal{M}_D(\theta)$ *and* $\bigwedge_{i=1}^{n} \mathcal{M}_D(\theta_i)$ *are logically equivalent for every* D.

Proof: The equivalence is obtained by applying some simple rewriting rules on the relevant formulas:

$$\begin{aligned}
\mathcal{M}_D(\theta) &\equiv \forall\bar{x}\big(\Psi[\bar{x}] \supset (\bigwedge_{i=1}^{n}(P_i(\bar{x}_i) \supset P_i(\bar{x}_i) \in P_i^D))\big) \\
&\equiv \bigwedge_{i=1}^{n} \forall\bar{x}\big(\Psi[\bar{x}] \supset (P_i(\bar{x}_i) \supset P_i(\bar{x}_i) \in P_i^D)\big) \\
&\equiv \bigwedge_{i=1}^{n} \forall\bar{x}_i\big(\exists|_{\bar{x}_i}\Psi[\bar{x}] \supset (P_i(\bar{x}_i) \supset P_i(\bar{x}_i) \in P_i^D)\big) \\
&\equiv \bigwedge_{i=1}^{n} \mathcal{M}_D(\theta_i). \qquad \square
\end{aligned}$$

$\square$

3.4.2 Expressing Complete Knowledge on an Object

A special case of the above sort of LCWA is when a database contains complete knowledge on a specific element of the domain. For example, the database of Examples 1.5 and 3.1 may be complete for Lien Desmet. For this kind of knowledge, it makes more sense to consider a sorted database schema Σ.

Definition 3.8 *Let* C *be a certain domain element. The complete object knowledge assumption about* C *is the expression*

$$\mathcal{LCWA}(\mathcal{O}, x = C),$$

where $\mathcal{O}$ *is the set of atoms* $P(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n)$, *for each predicate* P *and argument position* i *of the sort of* C.

Example 3.10 *Consider the database D of Example 3.1. The following expression declares that D has complete object knowledge about Lien Desmet (abbreviated LD):*

$$\mathcal{LCWA}(\{Dept(x,y), Tel(x,z)\}, x = LD).$$

Similarly, in Example 1.6, the assumption

$$\mathcal{LCWA}(\{CarO(x,y), Loc(x,z)\}, x = MC)$$

says that the database has complete object knowledge about Mary Clark.

3.4.3 LCWA with Several Databases

A context where local forms of closed-world assumptions seem even more relevant is in the context of distributed databases or, in general, all multiple-source environments. Typically, one expects that individual sources have some domain of expertise. In such a context, it could be useful to specify a LCWA that expresses collective expertise held by *several* sources together. That is, expressions like

$$\theta = \mathcal{LCWA}(\{D_1, \ldots, D_n\}, P(\bar{x}), \Psi[\bar{x}])$$

should represent complete knowledge, shared by databases $\{D_1, \ldots, D_n\}$, in the context Ψ, about the predicate P. The meaning of the expression above is similar to the case of one database (confer Definition 3.4).

Definition 3.9 *Let $D_1, \ldots, D_n$ be n database instances, $P \in \mathcal{R}(\Sigma)$, and $\theta = \mathcal{LCWA}(\{D_1, \ldots, D_n\}, P(\bar{x}), \Psi[\bar{x}])$. Define:*

$$\mathcal{M}_{\{D_1,\ldots,D_n\}}(\theta) = \mathcal{M}_{\{D_1,\ldots,D_n\}}\big(\mathcal{LCWA}(\{D_1, \ldots, D_n\}, P(\bar{x}), \Psi[\bar{x}])\big) = \forall\bar{x}\big(\Psi[\bar{x}] \supset (P(\bar{x}) \supset (P(\bar{x}) \in \textstyle\bigcup_{i=1}^{n} D_i))\big).$$

Note that all results and extensions in the previous sections also hold with respect to this generalization.

3.4.4 Boolean Combinations of LCWAs

So far, we have treated local closed-world assumptions as atomic statements which cannot be combined with other statements. Nevertheless, LCWAs can be combined in Boolean expressions for expressing more complex assumptions about the information endorsed by two or more databases. For instance, the following formula would express that "either D_1 or D_2 has complete knowledge about P":

$$\mathcal{M}_{D_1}(\mathcal{LCWA}(\{D_1\}, P(\bar{x}), \mathbf{t})) \vee \mathcal{M}_{D_2}(\mathcal{LCWA}(\{D_2\}, P(\bar{x}), \mathbf{t})).$$

Also, it may be useful, in a certain context, to express that the LCWA holds on Q if a LCWA holds on P. One could represent this as follows:

$$\mathcal{LCWA}(Q(\bar{x}), \mathcal{LCWA}(P(\bar{x}), \mathbf{t}))$$

Likewise, it is possible to express that the assumptions about D_1 and D_2 are complementary, and so forth.

3.4.5 LCWA and the Soundness Assumption

A common, implicit assumption about our notion of locally closed databases is that they are sound, that is: database instances do not contain erroneous information. As pointed out by Motro (Motro 1989) and Levy (Levy 1996), it is easy to dualize the concept of a locally complete database to the concept of *locally sound database*. Such an assumption can be elegantly formalized using a formula that resembles the LCWA. To see this, recall that the meaning of $\mathcal{LCWA}(P(\overline{x}), \Psi[\overline{x}])$ is given by

$$\mathcal{M}_D(\theta) = \forall\overline{x}\Big(\Psi[\overline{x}] \supset \big(P(\overline{x}) \supset (P(\overline{x}) \in P^D)\big)\Big).$$

Now, a local soundness assumption can be obtained by a simple switching of the subformulas above:

$$\forall\overline{x}\Big(\Psi[\overline{x}] \supset \big((P(\overline{x}) \in P^D) \supset P(\overline{x})\big)\Big).$$

Unconditional soundness of the database is obtained by replacing Ψ by the propositional constant $\mathbf{t}$ and is equivalent to the soundness formula $\bigwedge_{A\in D} A$ in Definition 3.7. We call such an expression a local soundness assumption, and we denote it by $\mathcal{LSA}$. Observe that in locally sound databases the soundness axioms have to be dropped.

Example 3.11 *We can represent the fact that the database is correct and complete with respect to car owners in the area of the Bronx:*

- $\theta_1 = \mathcal{LCWA}(CarOwner(x, y), Location(x, Bx))$
- $\theta_2 = \mathcal{LSA}(CarOwner(x, y), Location(x, Bx))$

The meaning of θ_1 is:

$$\forall\overline{x}\Big(Location(x, Bx) \supset \big(CarOwner(\overline{x}) \supset (CarOwner(\overline{x}) \in CarOwner^D)\big)\Big).$$

and the meaning of θ_2:

$$\forall\overline{x}\Big(Location(x, Bx) \supset \big((CarOwner(\overline{x}) \in CarOwner^D) \supset CarOwner(\overline{x})\big)\Big).$$

We can combine both formulas to obtain:

$$\forall \overline{x}\Big(Location(x, Bx) \supset \big((CarOwner(\overline{x}) \in CarOwner^D) \equiv CarOwner(\overline{x})\big)\Big).$$

The formula states correctness and completeness of car owners living in the Bronx. Observe that if the window of expertise $Location(x, Bx)$ is change by the unconditional $\mathbf{t}$*, then the axioms state both unconditional completeness and correctness.*

3.4.6 A Circumscriptive Approach to the LCWA

We now consider an alternative approach to the representation of the closed-word assumption, this time by second-order formulas, and show its equivalence to the approach given in the previous section.

Consider again the expression $\mathcal{LCWA}(CarO(x, y, z), x = MC)$. The local closed-world assumption could also be defined in this case in terms of sets, as follows:

$$\{y \mid \mathsf{CarO}(\mathsf{MC}, y)\} = \{y \mid \mathsf{CarO}(\mathsf{MC}, y) \in D\}.$$

Since the set on the left-hand side of this equation is always a superset of the set on the right-hand side, the condition could be rephrased as follows:

$$\{y \mid \mathsf{CarO}(\mathsf{MC}, y)\} \subseteq \{y \mid \mathsf{CarO}(\mathsf{MC}, y) \in D\}.$$

This condition is specified in terms of a set inclusion property, and it is common to express such conditions by means of circumscriptive formulas. These formulas express the aspiration that the set of tuples of a certain predicate, satisfying a certain condition, should be as minimal as possible. It is not surprising, therefore, that a variant of the notion of local closed-world assumption presented here has already been expressed in term of circumscriptive axioms (see (Doherty et al. 2000) and Section 3.6).

Definition 3.10 *Let $\theta = \mathcal{LCWA}(P(\bar{x}), \Psi[\bar{x}])$ be a local closed-world assumption for a database instance D. The* pseudo-circumscriptive form *of θ is the following (second-order) formula, denoted $C_D(\theta)$:*

$$\forall \overline{\Theta}\left(\bigwedge_{A \in D} A[P/\Theta] \supset \left(\forall \bar{x}\left(\Psi(\bar{x}) \supset \forall\big(\Theta \leqslant P\big)\right) \supset \forall \bar{x}\left(\Psi(\bar{x}) \supset (P \leqslant \Theta)\right)\right)\right),$$

where

- Θ *is a predicate variable with the same arity of P,*
- $P \leqslant Q$ *is an abbreviation for $P(\bar{x}) \supset Q(\bar{x})$.*

$C_D(\theta)$ is called pseudo-circumscriptive since it differs from a pure circumscription schema by introducing the first-order formula Ψ into the representation. Just as in Definition 3.4, Ψ represents the context in which P should be minimal.

Definition 3.11 *Let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database. Denote $\mathfrak{C}(\mathfrak{D})$ the following set of sentences:*

Soundness*:*	$\bigwedge_{A \in D} A$
Local Completeness*:*	$\bigwedge_{\theta \in \mathcal{L}} \mathcal{C}_D(\theta)$.
Domain Closure Axiom *(*$\mathsf{DCA}(Dom^{\mathfrak{D}})$*):*	$\forall x(\bigvee_{C \in Dom^{\mathfrak{D}}} x = C_i)$
Unique Name Axiom *(*$\mathsf{UNA}(Dom^{\mathfrak{D}})$*):*	$\bigwedge_{C \neq C' \in Dom^{\mathfrak{D}}} C \neq C'$

Theorem 6 *For every database $\mathfrak{D}$, $\mathcal{M}(\mathfrak{D})$ is equivalent to $\mathfrak{C}(\mathfrak{D})$.*

Proof: We have to show that when $\bigwedge_{A \in D} A$ holds,

$$\forall \bar{x}\Big(\Psi(\bar{x}) \supset \big(P(\bar{x}) \supset \big(P(\bar{x}) \in D\big)\big)\Big) \tag{3.3}$$

is equivalent to

$$\forall \Theta \Bigg(\underbrace{\bigwedge_{A \in D} A[P/\Theta]}_{(a)} \supset \Bigg(\underbrace{\forall \bar{x} \Big(\Psi(\bar{x}) \supset \forall \big(\Theta \leqslant P\big) \Big)}_{(b)} \supset \underbrace{\forall \bar{x} \Big(\Psi(\bar{x}) \supset (P \leqslant \Theta) \Big)}_{(c)} \Bigg) \Bigg) \tag{3.4}$$

Indeed,

($\Rightarrow$) First, observe that 3.4 is trivially true when (a) is false, hence it suffices to prove (c) is true whenever (a) is true. Moreover, (c) is trivially true when $\Psi(\bar{x})$ or $P(\bar{x})$ is false hence it suffices to prove that (c) is true whenever (a), $\Psi(\bar{x})$ and $P(\bar{x})$ are all true.

Let M be a model of $\bigwedge_{A \in D} A$ and (3.3), and consider some value Θ^M in M for the predicate variable Θ. We show that if $\bigwedge_{A \in D} A[P/\Theta]$ is satisfied, so is the sub-formula (c) of (3.4), and hence the whole formula (3.4) is true as well. Let us prove, then, that sub-formula (c) holds. Assume that for some $\bar{x}$, $\Psi(\bar{x})$ and $P(\bar{x})$ are true in M. As M is a model of (3.3), this implies that $P(\bar{x}) \in D$, i.e. for some tuple of terms $\bar{t}$ in the table of P in D, the equality $\bar{x} = \bar{t}^M$ holds in M. Since Θ^M satisfies $\bigwedge_{A \in D} A[P/\Theta]$, it follows that $\bar{x} \in \Theta^M$.

($\Leftarrow$) Let M be a model of $\bigwedge_{A \in D} A$ and (3.4). From $\bigwedge_{A \in D} A$ it follows that $\Theta \leqslant P$. It is obvious that $\Theta \leqslant P$ implies (b). Consequently (c) holds. Assume that there exist values $\bar{x}$ such that $\Psi(\bar{x})$ and $P(\bar{x})$ hold in M. To prove (3) we need to show that $P(\bar{x}) \in D$; or equivalently that there exists $\bar{t} \in D$ such that $\bar{x} = \bar{t}^M$. Because of (c) holds, it follows that $\bar{x} \in \Theta^M$. By our choice of Θ^M, this mean that for $\bar{c} \in D$, $\bar{x} = \bar{c}^M$. $\square$

By the above theorem, the counterparts of Propositions 11, 8, and 10 in terms of $\mathfrak{C}(\mathfrak{D})$ are also obtained.

The structure of the pseudo-circumscriptive formula of definition 3.10, which does not depend on the content of the database, suggests the potentiality of extending the concept of local closed world assumption to more general databases. For instance, one might consider a disjunctive database such as $D = \{P(a) \vee P(b)\}$, and the assumption $\mathcal{LCWA}(P(x), \mathbf{t})$. Intuitively, this could be interpreted as "although the database is not complete with respect to P, at least it knows that no other element of the domain can belong to P, except from a or b (where the 'or' here is interpreted exclusively)". The formula in Definition 3.10 expresses a set inclusion minimization, and in this case it would state an unconditional minimization of any extension of P. That is, an interpretation that satisfies both the disjunctive expression in D and the circumscriptive form of $\mathcal{LCWA}$ will necessarily state that either $P(a)$ or $P(b)$ is true, *but not both*.

3.5 Queries in Locally Closed Databases

So far, we have described how to *represent* partial completeness in databases. We now turn to *reasoning* with this kind of databases, that is: query answering in locally closed databases. In this section, we formally define this problem and the related problem of determining complete information on queries. The computational (data) complexity of these problems is investigated with respect to $Dom^{\mathfrak{D}}$, the finite active domain of the database. The intractability results that are obtained, motivate the work in the next chapter, where efficient approximate methods under-estimating certain answers and overestimating possible answers are developed, as well as conditions under which these approximate methods produce complete answers.

In standard databases and under the closed-world assumption, the user poses a query and, if the query is safe, the system retrieves the tuples that make the query true. If the query is grounded, the answer is always either 'yes' or 'no'. Due to their incomplete nature, in locally closed databases the situation is slightly different. A tuple $\bar{t}$ can make a query $\mathcal{Q}[\bar{t}]$ 'true', 'false' or 'unknown'. The situation is illustrated in Figure 3.2. In this schema the rectangles represent the universe of a set of tuples $\bar{t}$. The different colors in each area represent the tuples that make the query $\mathcal{Q}$ false, unknown (possibly true/possibly false) or true. The more negative information is added to the database (by means of local closed-world assumption), the smaller the area in gray. When this area disappears, i.e. there is no incomplete information for $\mathcal{Q}$ and every tuple $\bar{t}$ is either true or false, we say that the database has *complete world information on a query*. This is represented in figure (b), where there is no uncertainty about tuples with respect to query Q. This situation is studied in the following section.

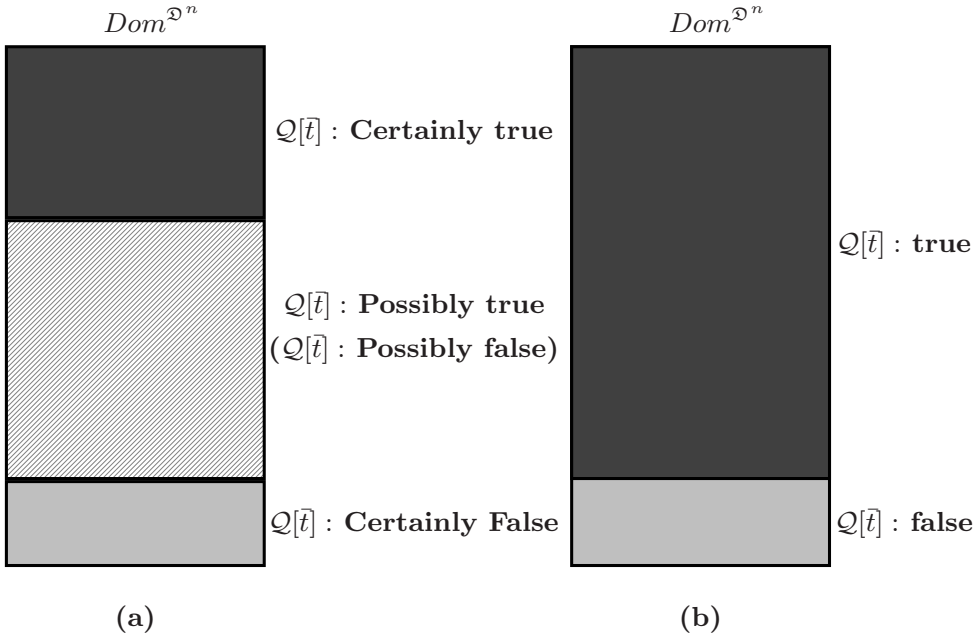


Figure 3.2: (a) represents generic answers to query $\mathcal{Q}(\bar{x})$ in a locally closed database. (b) represents the answers in a standard database under the closed-world assumption.

3.5.1 Query Answering and Complete Information

Definition 3.12 (Certain and possible answers) *Let $\mathfrak{D}$ be a locally closed database over Σ, $\mathcal{Q}[\bar{x}]$ a first-order query over $\sigma_{\mathfrak{D}}$ (whose free variables are in $\bar{x}$), and $\bar{t}$ a tuple of constants in $Dom^{\mathfrak{D}}$.*

- *$\bar{t}$ is a* certain answer *in $\mathfrak{D}$ for $\mathcal{Q}[\bar{x}]$, if $\mathfrak{D} \models \mathcal{Q}[\bar{t}]$.*
- *$\bar{t}$ is a* possible answer *in $\mathfrak{D}$ for $\mathcal{Q}[\bar{x}]$, if $\mathfrak{D} \cup \{\mathcal{Q}[\bar{t}]\}$ is satisfiable (equivalently, if $\mathfrak{D} \not\models \neg\mathcal{Q}[\bar{t}]$).*

In the sequel, given a locally closed database $\mathfrak{D} = (D, \mathcal{L})$, we denote by $Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$ the set of certain answers of $\mathcal{Q}[\bar{x}]$ in $\mathfrak{D}$ and by $Poss_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$ the set of possible answers of $\mathcal{Q}[\bar{x}]$ in $\mathfrak{D}$.

In contrast to complete databases, in locally closed databases we have to specify whether we are interested in tuples that, for a given query Q, are certainly true, certainly false or unknown. It is not difficult to see that the following relations always hold:

- $Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) = Dom^{\mathfrak{D}} \setminus Poss_{\mathfrak{D}}(\neg\mathcal{Q}[\bar{x}])$.
- $Cert_{\mathfrak{D}}(\neg\mathcal{Q}[\bar{x}]) = Dom^{\mathfrak{D}} \setminus Poss_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$.
- $Poss_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) = Dom^{\mathfrak{D}} \setminus Cert_{\mathfrak{D}}(\neg\mathcal{Q}[\bar{x}])$.
- $Poss_{\mathfrak{D}}(\neg\mathcal{Q}[\bar{x}]) = Dom^{\mathfrak{D}} \setminus Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$.

We illustrate query answering and these properties with a simple example:

Example 3.12 *Consider the LCWA expression*

$$\mathcal{LCWA}(CarOwner(x, y, z), x = MC \vee x = JS),$$

stating that the database knows all the cars of Mary Clark and John Smith. The database instance is as follows:

$$\begin{aligned} D = \{&CarOwner(MC, BMW550, Bx5462),\\ &CarOwner(MC, Volvo330, Mh3423),\\ &CarOwner(JS, Volvo230, Bx5242),\\ &CarOwner(PS, Mercedes320, Qn5452)\} \end{aligned}$$

We illustrate the concepts of certain and possible answers by means of some simple queries:

1. *$Cert(CarOwner(MC, BMW550, Bx5462))$: yes.*

2. $Cert(\neg CarOwner(MC, Peugeot207, y))$: $y = Dom^{\mathfrak{D}}$. *Since the database is complete with respect to cars of Mary Clark, every tuple which is not in the table and which contains MC as first attribute, is certainly false. It follows that for every car ID y the tuple* $CarOwner(MC, Peugeot207, y)$ *is certainly false.*

3. $Poss(CarOwner(PS, BMW230, Qn5452))$: *true. Since there is no closure on Peter Steward, every combination of Car and ID is a possible answer for CarOwner. These answers violate the intuition that every car must have one and only one car ID* [2].

4. $Poss(CarOwner(MC, x, y))$:

$$\{(BMW550, Bx5462), (Volvo330, Mh3423)\}.$$

The database is complete for Mary Clark. It follows that certain and possible answers for the query $Poss(CarOwner(MC, x, y))$ *coincide, and that they can be extracted solely from the database regardless of the active domain. This property is generalized in Proposition 13 below.*

Proposition 12 *Let* $\mathcal{Q}[\bar{x}]$ *be a query containing only base predicates of* $\mathfrak{D}$. *Then*

$$Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) = Poss_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) = \{\bar{a} \mid D \models \mathcal{Q}[\bar{a}]\}.$$

Another interesting problem for a query $\mathcal{Q}[\bar{x}]$ in a locally closed database $\mathfrak{D}$ is whether $\mathfrak{D}$ has complete knowledge on $\mathcal{Q}[\bar{x}]$. The idea of complete information on queries has also been called *Closed-World Information* CWI on a query, and it was used by Levy (Levy 1996) in the context of incomplete databases. In (Etzioni et al. 1997), this notion was introduced to the logical agents setting. It can be defined as follows:

Definition 3.13 (Closed-world information, CWI) *A locally closed database* $\mathfrak{D}$ *over* Σ *has* closed-world information *on a query* $\mathcal{Q}[\bar{x}]$, *if for each tuple* $\bar{t}$ *of constants in* $Dom^{\mathfrak{D}}$, *either* $\mathfrak{D} \models \mathcal{Q}[\bar{t}]$ *or* $\mathfrak{D} \models \neg\mathcal{Q}[\bar{t}]$.

Obviously, when $\mathfrak{D}$ has complete information about $\mathcal{Q}[\bar{x}]$ then certain and possible answers coincide, i.e., $Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) = Poss_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$. This property is depicted in Figure 3.2. Such queries are of practical importance, since there is no uncertainty about their answers.

Proposition 13 $\mathfrak{D}$ *has CWI on a query* $\mathcal{Q}[\bar{x}]$ *iff*

$$Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) = Poss_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) = \{\bar{a} \mid D \models \mathcal{Q}[\bar{a}]\}.$$

[2] In the next chapter we analyze such situations and study how to incorporate so-called *integrity constraints* into locally closed databases to give account of this situation.

Proof: Since D is a model of $\mathfrak{D}$, the following equation holds:

$$\{\bar{a} \mid D \models \mathcal{Q}[\bar{a}]\} \subseteq Poss_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) = Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) \subseteq \{\bar{a} \mid D \models \mathcal{Q}[\bar{a}]\},$$

and so the proposition is obtained. □

Example 3.13 *See item 4. in Example 3.12.*

It follows that queries with CWI can be answered directly in the database D. This was Levy's motivation to study CWI. Observe that LCWA and CWI are related concepts that capture different phenomena. The LCWA expresses completeness of a part of a *database table* in a relational database, while the CWI identifies completeness of a *query* posed to the database.

However, frequently, LCWA induce CWI on queries. For example, a locally closed database $\mathfrak{D} = (D, \mathcal{L})$, such that $\mathcal{LCWA}(P(x), x = a) \in \mathcal{L}$, conveys CWI on $P(a)$, no matter what D is. Likewise, in Example 3.12, the expression $\mathcal{LCWA}(CarOwner(x, y, z), x = MC \vee x = JS)$ induced CWI on the queries $CarOwner(MC, x, y)$ and $CarOwner(JS, x, y)$.

This observation can be generalized, as the following proposition shows:

Proposition 14 *If $\theta = \mathcal{LCWA}(P(\bar{x}), \Psi(\bar{x}))$ is an LCWA of $\mathfrak{D}$ and, for some formula $\Phi[\bar{x}]$, $\mathfrak{D}$ conveys CWI on $\Phi[\bar{x}] \wedge \Psi[\bar{x}]$, then $\mathfrak{D}$ conveys CWI on the queries $\Phi[\bar{x}] \wedge \Psi[\bar{x}] \wedge (\neg)P(\bar{x})$, where $(\neg)P(\bar{x})$ denotes the positive or negative literal containing $P(\bar{x})$.*

Proof: Since we have CWI on the first part of the formula, it holds that for each tuple $\bar{t}$ of $Dom^{\mathfrak{D}}$, the sentence $\Phi[\bar{t}] \wedge \Psi[\bar{t}]$ is either false in all models of $\mathfrak{D}$ or true in all of them. In the first case, $\Phi[\bar{t}] \wedge \Psi[\bar{t}] \wedge (\neg)P(\bar{t})$ is also false in all models of $\mathfrak{D}$ and $\mathfrak{D} \models \neg(\Phi[\bar{t}] \wedge \Psi[\bar{t}] \wedge (\neg)P(\bar{t}))$, so CWI holds in this case. In the second case, $\Psi[\bar{t}]$, the window of expertise is true and we have complete information about $P(\bar{t})$ and either $\mathfrak{D} \models P(\bar{t})$ or $\mathfrak{D} \models \neg P(\bar{t})$. Hence either $\mathfrak{D} \models \Phi[\bar{t}] \wedge \Psi[\bar{t}] \wedge (\neg)P(\bar{t})$ or $\mathfrak{D} \models \neg(\Phi[\bar{t}] \wedge \Psi[\bar{t}] \wedge (\neg)P(\bar{t}))$ so CWI also holds in this case. □

For the case that $\Phi[\bar{x}] = \mathbf{t}$, the proposition implies that CWI holds for $\Psi[\bar{x}] \wedge (\neg)P(\bar{x})$ whenever CWI holds for the window of expertise $\Psi[\bar{x}]$ of predicate P. This is the case when the window of expertise contains only base predicates.

The condition that $\mathfrak{D}$ should have CWI on $\Phi[\bar{x}] \wedge \Psi[\bar{x}]$ is necessary. Assume that $\mathfrak{D}$ contains $\mathcal{LCWA}(P(x), Q(x))$ and no LCWA for Q. Clearly, there is no CWI on the query $Q(x) \wedge P(x)$. For example, if $Q(c) \notin D$, then $c \in Poss_{\mathfrak{D}}(Q(x) \wedge P(x)) \setminus Cert_{\mathfrak{D}}(Q(x) \wedge P(x))$. If we now add $\mathcal{LCWA}(Q(x), \mathbf{t})$, then Q becomes a base predicate and we obtain CWI on $Q(x) \wedge P(x)$.

The following proposition shows that the class of queries with CWI is closed under certain operations.

Proposition 15 *Let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database and $\Psi[\bar{x}], \Phi[\bar{y}]$ be queries on which $\mathfrak{D}$ conveys CWI. Then $\mathfrak{D}$ conveys CWI on $\neg\Psi[\bar{x}], \Psi[\bar{x}] \wedge \Phi[\bar{y}]$, $\Psi[\bar{x}] \vee \Phi[\bar{y}]$, and on $(\exists x\Psi)[\bar{x}']$, $(\forall x\Psi)[\bar{x}']$ when $x \in \bar{x}$ and $\bar{x}' = \bar{x} \setminus \{x\}$.*

Proof: We prove the case of $(\exists x\Psi)[\bar{x}']$ (assuming that CWI exists for $\Psi[\bar{x}]$); the other cases are similar. Let $\bar{t}'$ be a tuple of terms of the size of $\bar{x}'$ and assume that $\mathfrak{D} \not\models (\exists x\Psi)[\bar{t}']$. Then, for some model M of $\mathfrak{D}$, we have $M \not\models (\exists x\Psi)[\bar{t}']$, or equivalently, $M \models (\forall x\neg\Psi)[\bar{t}']$. Thus, for all $c \in Dom^{\mathfrak{D}}$, $M \models \neg\Psi[(\bar{t}', c)]$. Since there is CWI for the query $\Psi[\bar{x}]$, there is CWI for $\neg\Psi[\bar{x}]$ as well. Therefore, for all models N of $\mathfrak{D}$, for all $c \in Dom^{\mathfrak{D}}$, we have $N \models \neg\Psi[(\bar{t}', c)]$, from which it follows that $\mathfrak{D} \models (\forall x\neg\Psi)[\bar{t}']$, or equivalently, $\mathfrak{D} \models \neg(\exists x\Psi)[\bar{t}']$ and thus CWI holds for the latter query. □

Example 3.14 *Consider a database $\mathfrak{D}$ consisting of the following set of local closed-world assumptions:*

$$\mathcal{L} = \left\{ \begin{array}{ll} \mathcal{LCWA}(P_1(x), \mathbf{t}) & \mathcal{LCWA}(P_2(x), \mathbf{t}) \\ \mathcal{LCWA}(Q(x), P_1(x) \wedge P_2(x)) & \mathcal{LCWA}(Q(x), S(x)) \\ \mathcal{LCWA}(S(x), Q(x)) & \mathcal{LCWA}(R(x), Q(x)) \end{array} \right\}$$

From the above propositions, some of the formulas to which $\mathfrak{D}$ determines CWI can be derived by the following stages:

1. $(\neg)P_1(x)$, $(\neg)P_2(x)$.
2. $P_1(x)\wedge P_2(x)$, $(\neg)P_1(x)\wedge P_2(x)$. $P_1(x)\wedge(\neg)P_2(x)$, $(\neg)P_1(x)\wedge(\neg)P_2(x)$.
3. $(\neg)Q(x) \wedge P_1(x) \wedge P_2(x)$.
4. $(\neg)S(x) \wedge Q(x) \wedge P_1(x) \wedge P_2(x)$, *and so forth.*

3.5.2 Complexity Results

In this section, we investigate the data complexity of querying locally closed databases. First, we identify a useful tractable class:

Proposition 16 *Let $\mathfrak{D}$ be a locally closed database. For every positive query $\mathcal{Q}[\bar{x}]$, i.e., in which all database predicates occur positively, $Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$, i.e. the set : $\{\bar{d} \in D^n \mid \mathfrak{D} \models \mathcal{Q}[\bar{d}]\}$, is the set $\{\bar{d} \in D^n \mid D \models \mathcal{Q}[\bar{d}]\}$. Moreover, this set can be computed in polynomial time.*

Proof: Consider a tuple $\bar{d}$ such that $D \models \mathcal{Q}[\bar{d}]$. Because D is the least model and $\mathcal{Q}$ is positive, it holds that $M \models \mathcal{Q}[\bar{d}]$ for every model M of $\mathfrak{D}$, i.e. $\mathfrak{D} \models \mathcal{Q}[\bar{d}]$ and $\bar{d}$ is a certain answer. Polynomial time follows from well-known data complexity results of relation calculus query answering (Vardi 1982). □

For positive queries, locally closed databases behave as standard relational databases under the closed-world assumption. In contrast to the last result, in Proposition 17 below we show that query answering in locally closed databases is in general a computationally hard problem.

Following the usual measure of complexity in databases, the results below are specified in terms of data complexity, that is, in terms of the size $|Dom^{\mathfrak{D}}|$

of the domain of the database instance (assuming that all the rest is fixed). Accordingly, we consider the following decision problems:

$$\mathrm{Poss}_{\mathcal{L}}(\mathcal{Q}[\bar{x}]) = \{(D, \bar{t}) \mid \bar{t} \in \mathit{Poss}_{(D,\mathcal{L})}(\mathcal{Q}[\bar{x}])\},$$
$$\mathrm{Cert}_{\mathcal{L}}(\mathcal{Q}[\bar{x}]) = \{(D, \bar{t}) \mid \bar{t} \in \mathit{Cert}_{(D,\mathcal{L})}(\mathcal{Q}[\bar{x}])\}.$$

The rationality behind the definition of these decision problems is that it seems natural to assume that the local closed-world assumptions in $\mathcal{L}$, just like integrity constrains, are fairly constant during the lifetime of a database $\mathfrak{D}$ compared to the data in D, and that queries do not grow beyond certain limits. Therefore, we consider the complexity of fixed parameter problems of $\mathrm{Poss}_{\mathcal{L}}(\mathcal{Q}[\bar{x}])$ and $\mathrm{Cert}_{\mathcal{L}}(\mathcal{Q}[\bar{x}])$ where both $\mathcal{L}$ and $\mathcal{Q}$ are fixed.

Proposition 17 *The decision problem* $\mathrm{Poss}_{\mathcal{L}}(\mathcal{Q}[\bar{x}])$ *is in* NP [3] *for all* $\mathcal{L}$ *and* $\mathcal{Q}[\bar{x}]$ *and is* NP*-hard for some of them.* $\mathrm{Cert}_{\mathcal{L}}(\mathcal{Q}[\bar{x}])$ *is in* coNP *for each* $\mathcal{L}$ *and* $\mathcal{Q}[\bar{x}]$ *and is* coNP*-hard for some of them.*

Proof: There is a one-to-one correspondence between models of $\mathfrak{D}$ and supersets D' of D satisfying $\mathcal{L}$. An algorithm to check whether $\bar{t}$ is a possible or certain answer of $\mathcal{Q}[\bar{x}]$ in $\mathfrak{D} = (D, \mathcal{L})$ is to choose non-deterministically such a superset D' of D, and check whether D' satisfies $\mathcal{Q}[\bar{t}]$ for each $\theta \in \mathcal{L}$. As these checks are polynomial in the size of the domain of D, it follows that $\mathrm{Poss}_{\mathcal{L}}(\mathcal{Q}[\bar{x}])$ is in NP and $\mathrm{Cert}_{\mathcal{L}}(\mathcal{Q}[\bar{x}])$ is in coNP.

Hardness is shown by a reduction from the graph kernel problem (Garey and Johnson 1979) as follows: let $\mathcal{R}(\Sigma) = \{\mathit{Edge}, \mathit{Kernel}, P\}$ and consider the following local closed-world assumptions:

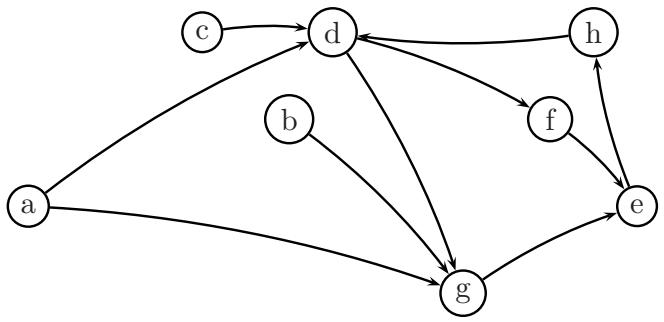
$$\mathcal{L} = \left\{ \ \mathcal{LCWA}(\mathit{Edge}(x, y), \mathbf{t}), \quad \mathcal{LCWA}(P(c), \neg\Phi) \ \right\}$$

where Φ is the formula:

$$\forall x \forall y \big(\mathit{Kernel}(x) \wedge \quad \mathit{Kernel}(y) \supset \neg \mathit{Edge}(x, y) \big) \ \bigwedge$$
$$\forall x \big(\neg \mathit{Kernel}(x) \supset \exists y (\mathit{Kernel}(y) \wedge \mathit{Edge}(y, x)) \big)$$

Clearly, Φ expresses that Kernel is a kernel of the graph described by Edge. For a given graph G, let D be the database with the vertices of G as domain, the edges of G represented by Edge and $\mathit{Kernel}^D = P^D = \emptyset$. It is easy to show that $\mathfrak{D} = (D, \mathcal{L})$ has a model in which P is a relation containing c, i.e. c is a possible answer for $P(x)$, iff G has a kernel. Since deciding whether a graph has a kernel is an NP-complete problem, deciding whether c is a possible answer to the query $P(x)$ is NP-hard, and deciding whether c is a certain answer to $\neg P(x)$ is coNP-hard. An instance of the reduction is shown in Figure 3.3. □

[3] Readers not familiar with the theory of computational complexity can consult the introductory book by Garey et al. (Garey and Johnson 1979)



$$Dom^{\mathfrak{D}} = \{a, b, c, d, e, f, g, h\}$$

$$Edge = \{(G, E), (F, E), (D, F), (D, G), (E, H), \\ (H, D), (B, G)(C, D), (A, D), (A, G)\}$$

$$Kernel = \emptyset$$

$$\mathcal{L} = \{\theta_1, \theta_2, \theta_3\}$$

$$\theta_1 = \mathcal{LCWA}(Edge(x, y), \mathbf{t})$$

$$\theta_2 = \mathcal{LCWA}\big(P(c), \neg\big(\forall x \forall y\big(Kernel(x) \wedge Kernel(y) \supset \neg Edge(x, y)\big) \ \bigwedge \\ \forall x\big(\neg Kernel(x) \supset \exists y(Kernel(y) \wedge Edge(y, x))\big)\big)$$

$$\mathcal{Q} : P(c)$$

Figure 3.3: An instance of the reduction from the Kernel Problem. The domain of the database corresponds to all nodes of the input graph. The database for the relation *Edge* is constructed from the edges in the graph, and it is assumed that the representation is complete (θ_1). θ_2 encodes that the database is complete on $P(c)$ if the input graph has a kernel. Nodes a, b, c, f and h form a kernel. Another kernel is formed by d, e and b.

Next, we examine the complexity of determining CWI. For a given set $\mathcal{L}$ of LCWA and query $\mathcal{Q}[\bar{x}]$, consider the following decision problem:

$$\mathrm{CWI}_{\mathcal{L}}(\mathcal{Q}[\bar{x}]) = \{D \mid (D, \mathcal{L}) \text{ has CWI on } \mathcal{Q}[\bar{x}]\}.$$

As the following proposition shows, the decision problem whether a locally closed database has complete knowledge on a given query, is also not tractable:

Proposition 18 *The decision problem* $\mathrm{CWI}_{\mathcal{L}}(\mathcal{Q}[\bar{x}])$ *is in* coNP *for each* $\mathcal{L}$ *and* $\mathcal{Q}[\bar{x}]$*, and is* coNP*-hard for some of them.*

Proof: Observe that D is a model of $\mathfrak{D}$. Therefore, $\mathfrak{D}$ has CWI on a query $\mathcal{Q}[\bar{x}]$ iff for each model D' of $\mathfrak{D}$, it holds that $\{\bar{t} \mid D' \models \mathcal{Q}[\bar{t}]\} = \{\bar{t} \mid D \models \mathcal{Q}[\bar{t}]\}$. To compute this, we non-deterministically choose a superset $D' \supseteq D$, verify whether each $\theta \in \mathcal{L}$ is satisfied and verify for all tuples $\bar{t}$ whether $\mathcal{Q}[\bar{t}]^D = \mathcal{Q}[\bar{t}]^{D'}$; if a D' is found for which this does not hold then $\mathfrak{D}$ has no CWI on $\mathcal{Q}[\bar{x}]$.

What remains to be shown is the existence of coNP-hard instances for fixed parameters $\mathcal{L}$ and $\mathcal{Q}[\bar{x}]$. This follows from the fact that, in the case of the kernel databases constructed in the proof of Proposition 17, $\mathfrak{D}$ has complete information on $P(c)$ iff $\neg P(x)$ has c as a certain answer. The latter decision problem is coNP-hard. □

Proposition 18 provides the complexity of deciding whether there is CWI on a query $\mathcal{Q}[\bar{x}]$ in a specific database $\mathfrak{D} = (D, \mathcal{L})$. In (Levy 1996), Levy presents a polynomial decision procedure for determining whether, for a given set $\mathcal{L}$ of local closed-world assumptions and query $\mathcal{Q}[\bar{x}]$, there is CWI on $\mathcal{Q}[\bar{x}]$ in *all* locally closed databases containing $\mathcal{L}$. This decision procedure takes advantage of the fact that Levy's windows of expertise are special cases of (positive) conjunctive queries (called by Levy 'variable-interval queries') and $\mathcal{Q}[\bar{x}]$ is the union of (positive) conjunctive queries. Not surprisingly, such a decision procedure does not exist for the general framework of the current thesis.

Proposition 19 *The question whether all locally closed databases* $(\cdot, \mathcal{L})$ *convey CWI on a query* $\mathcal{Q}[\bar{x}]$ *is undecidable.*

Proof: The proof is a variant of the coNP-hardness proof; the main idea is that the problem at hand can be reduced to the validity checking problem of first-order formulas in the class of finite structures, and this problem is, by Trakhtenbrot's theorem (Trakhtenbrot 1963), undecidable. Consider a relational vocabulary Σ containing one unary predicate P and a constant c. Take $\mathcal{Q}[\bar{x}] = P(c)$ and $\mathcal{L} = \{\theta\}$ with $\theta = \mathcal{LCWA}(P(c), \varphi)$, where φ is a sentence not containing P. We observe that a database $(D, \mathcal{L})$ has no CWI on $P(c)$ iff $\neg\varphi$ has a finite model. Indeed, if $\neg\varphi$ has a finite model M, then take D to be M extended with the empty table for P. Clearly, both D and the extension of D

in which c is added to P's table, are models of $(D, \mathcal{L})$, and hence this database has no CWI on $P(c)$. Vice versa, a database $(D, \mathcal{L})$ that has no CWI on $P(c)$, does not contain $P(c)$ but has a finite model M in which $P(c)$ is true. This model satisfies $\mathcal{M}_D(\theta) \equiv \varphi \supset (P(c) \supset P(c) \in D)$, and hence, $M \models \neg\varphi$. It follows from the observation that there is CWI on $P(c)$ in all databases $(D, \mathcal{L})$ iff φ is satisfied in all finite structures. By Trakhtenbrot's theorem, this is undecidable.

□

3.5.3 Summary of the Results

In Table 3.1 we describe the results obtained in this section together with some related results by Levy (Levy 1996).

Summary Complexity Results		
Computational Task	*Restriction language*	*Complexity*
$Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$	Positive Queries	P
$\text{Poss}_{\mathcal{L}}(\mathcal{Q}[\bar{x}])$	none	NP
$\text{Cert}_{\mathcal{L}}(\mathcal{Q}[\bar{x}])$	none	CoNP
$\text{CWI}_{\mathcal{L}}(\mathcal{Q}[\bar{x}])$	Database fixed	CoNP
$\text{CWI}_{\mathcal{L}}(\mathcal{Q}[\bar{x}])$	$\mathcal{L}$: Conjunctive queries	P (Levy 1996)
$\text{CWI}_{\mathcal{L}}(\mathcal{Q}[\bar{x}])$	none	Undecidable

Table 3.1: Complexity of computational tasks for locally closed databases.

Except for the cases where the language of the queries or the local closed-world assumptions is restricted, the results in this section give little reason for optimism regarding practical applicability of locally closed databases in the general case. However, it is the case in many practical applications that we do not need to have *all* certain answers to a query; often, it suffices to have a sufficiently large subset of those answers. E.g., if a company searches an (incomplete) database for a provider of some urgently required services, it will be happy if it finds *some* candidate providers; this list does not need to be complete. Likewise, in many applications, it would not harm if the answers to a *possible* query contained a few extra "impossible" elements. E.g., if a company wants to advertise one of its services and queries the database for a group of potential clients, it would not mind to receive some additional companies that could not really be possibly interested. So, one reasonable strategy to solve the complexity problem would be to develop tractable approximate methods for query answering. This is the motivation for the approach we present in the next chapter.

Another approach to reduce computational complexity is to restrict the expressiveness of the language of either queries, LCWA expressions or both, so

that efficient query processing is possible. This is the approach taken by Levy (Levy 1996). In the next chapter we explore this alternative as well, though in a slightly indirect way: we will show that for certain classes of queries and local closed-world assumptions, the approximate methods are *optimal* in the sense that they compute exactly the certain and possible answers to queries. Thus, these combinations of queries and local closed-world assumptions provide tractable sub-formalisms.

3.6 Related Work

The research about representing and reasoning with incomplete information in relational databases is wide and diverse, and relations between the different approaches are not always investigated or known. We do not intend to give here an exhaustive description of all related work. Instead, we focus on previous research that we consider to be close in spirit to our own goals and explain in detail the relations with our work.

3.6.1 Local Forms of CWA in Databases

To the best of our knowledge, at least two different local forms of CWA have been proposed in the literature. These two different approaches are formalized using different styles of syntax and semantics and with different goals in mind, making it quite difficult to compare them. Therefore, we will explain these approaches in the terminology of our thesis and we will focus on the relation with our approach. We will start this comparison with the approach we consider closer to ours, namely the one of Levy (Levy 1996).

3.6.1.1 Levy's Local Completeness Approach

In Levy's databases terminology, a locally closed database is called a *partial database*. Just like in our presentation, it consists of a database instance and a set of variants of the local closed-world assumptions, called *local completeness expressions*. To formalize the semantics of such an expression, Levy distinguishes between two sets of relations, *virtual* and *available*. The virtual relations P represent the predicates in the real world, while the available relations P' represent the database tables. Local closed-world assumptions are represented in a database notation by expressions of the form

$$LC(R', R, C),$$

where R' and R are respectively available and virtual relations, and C – the *constraint* in Levy's terminology – is a conjunction of positive virtual atoms not including R with free variables $\bar{x}$ and $\bar{y}$, where $\bar{x}$ is a tuple of variables

standing for R's attributes. Taking Levy's original example, the assumption that a database is complete for all movies after 1965, is represented by

$$LC(Movie', Movie, year \geqslant 1965).$$

Here, $Movie$ is a database predicate with attributes $(Title, Director, Year)$ and $year$ is the variable standing for the attribute $Year$. In our notations, this assumption is expressed by

$$\mathcal{LCWA}(Movie(x, y, z), z \geqslant 1965).$$

Interpreting Levy's semantics for partial databases in the context of our approach, a model of a partial database $\mathfrak{D} = (D, \mathcal{L})$ is defined as any structure M with the domain of D, such that for each virtual predicate R, $R^D = (R')^M \subseteq R^M$, and for each $LC(R'(\bar{x}), R(\bar{x}), C[\bar{x}, \bar{y}]) \in \mathcal{L}$.

$$M \models \forall \bar{x}\, (\exists \bar{y}\, C[\bar{x}, \bar{y}] \supset (R(\bar{x}) \supset R'(\bar{x}))).$$

Thus, such a model represents both a possible state of the world (in the virtual predicates) and an image of the database (in the available predicates). The logical semantics of $\mathfrak{D}$ can be formalized by the following theory $\mathcal{M}'(\mathfrak{D})$:

Definition of the available predicates:

$$\bigwedge_{P \in \mathcal{R}(\Sigma)} \forall \bar{x}(P'(\bar{x}) \equiv P(\bar{x}) \in D)$$

Soundness:

$$\bigwedge_{P \in \mathcal{R}(D)} \forall \bar{x}(P'(\bar{x}) \supset P(\bar{x}))$$

LCWA:

$$\bigwedge_{LC(P(\bar{x}), C[\bar{x}, \bar{y}]) \in \mathcal{L}} \forall \bar{x}(\exists \bar{y} C[\bar{x}, \bar{y}] \supset (P(\bar{x}) \supseteq P'(\bar{x})))$$

Domain Closure Axiom ($\mathsf{DCA}(Dom^D)$):

$$\forall x(\bigvee_{C \in Dom^D} x = C)$$

Unique Name Axiom ($\mathsf{UNA}(Dom^D)$):

$$\bigwedge_{C \neq C' \in Dom^D} C \neq C'$$

The formulas above represent, respectively, the content of the database D by available predicates P', the soundness of the database, the meaning of the local closed-world assumptions, and the database domain. Note that, here, by introducing the available predicates as representing the database tables, all axioms except for the definition of these available predicates are database independent.

We now show that the semantics of partial databases and of locally closed databases coincide.

Definition 3.14 (Equivalence of theories) *Let $\Sigma \subseteq \Sigma_1 \cap \Sigma_2$. We say that theories T_1 over Σ_1, and T_2 over Σ_2 are* equivalent in Σ*, if for each Σ_1-model M of T_1, there is a Σ_2 model N of T_2 such that $M|_\Sigma = N|_\Sigma$, and vice versa.*

Proposition 20 *For each locally closed database $\mathfrak{D}$ over Σ, it is true that $\mathcal{M}(\mathfrak{D})$ and $\mathcal{M}'(\mathfrak{D})$ are equivalent in Σ.*

Proof: By substituting $P'(\bar{x})$ by $P(\bar{x}) \in D$ in the soundness formula and by simplifying the resulting formula, we can obtain $\bigwedge_{A \in D} A$. By substituting $P'(\bar{x})$ by $P(\bar{x}) \in D$ in the sentences formalizing the LCWA, we obtain $\bigwedge_{\theta \in \mathcal{L}} \mathcal{M}_D(\theta)$. The resulting formula is nothing else than $\mathcal{M}(\mathfrak{D})$ augmented with an explicit definition of the predicates P'. The latter predicates have no further occurrences except for in their definitions. It follows that this formula is equivalent in Σ to $\mathcal{M}(\mathfrak{D})$, and so the proposition holds. □

With this syntactic and semantic machinery in order, Levy studies the problem of deciding whether a partial database has sufficient information to be able to solve a query $\mathcal{Q}[\bar{x}]$ in a complete way. That is, whether the answers for this query computed from the database, are exactly those that hold true in the real world. In this thesis we have referred to this problem as deciding *closed-world information* (CWI) on a query (see Section 3.5, Definition 3.13). Levy tackles this problem by reducing it to the problem of determining independence of queries from updates (see (Levy and Sagiv 1993) for a detailed description of independence of queries from updates). Exploiting results on the latter problem, he is able to show decidability of the first problem for certain subclasses of his formalism.

Apart from the difference in forms that we use logical notations rather than Levy's database notations, our local closed-world assumptions are more expressive than Levy's. In particular, in our case the windows of expertise need not be conjunctive nor positive. Also the motivations for studying locally closed databases and the results obtained belong to different fields. While Levy mostly studies the problem of CWI on queries, we will study other types of reasoning (see also Section 3.5), in particular query answering.

3.6.1.2 Motro's Validity and Completeness Constraints

Perhaps the first study of local forms of CWA was done by Motro (Motro 1989), who investigated partial forms of validity (soundness) and completeness

in relational databases. Both *validity constraints* and *completeness constraints* are represented as queries (in databases terminology, views) over the database vocabulary.

Intuitively, a completeness constraint is a query $\Psi[\bar{x}]$ such that its answers in the real world are also the answers computed from the database D; validity constraints specify the inverse. Motro then introduces an algorithm whose input is a set $\mathcal{V}$ of validity constraints, a set $\mathcal{C}$ of completeness constraints, and an arbitrary query $\mathcal{Q}[\bar{x}]$. The algorithm computes a lower approximation $\mathcal{Q}_l[\bar{x}]$ and an upper approximation $\mathcal{Q}_u[\bar{x}]$ of the query answers. Thus, any answer for $\mathcal{Q}[\bar{x}]$ computed from D that satisfies $\mathcal{Q}_l[\bar{x}]$ is a correct answer (that is, it is an answer to the query in the real world), and any correct answer for $\mathcal{Q}[\bar{x}]$ in the real world that satisfies $\mathcal{Q}_u[\bar{x}]$ will be computed from D. The algorithm requires that the query and all the constraints are existentially quantified conjunctions of positive literals.

Formally, a completeness constraints is an expression of the form $\mathsf{CC}(\Psi[\bar{x}])$[4]. Let us now investigate the formal semantics of this formalism. For the sake of simplicity, we focus on a database where $\mathcal{V}$ contains for every $P \in \mathcal{R}(\Sigma)$, so the database is sound. A completeness constraint $\Psi[\bar{x}] \in \mathcal{C}$ means that its answers in *the real world* can be computed from the database. We do not know what is the real world, but the information conveyed about the real world by this constraint is formalized by the following FO formula:

$$\forall \bar{x} \left(\Psi[\bar{x}] \supset \bigvee_{\bar{a} \in \{\bar{a} \mid D \models \Psi[\bar{a}]\}} \bar{x} = \bar{a} \right). \tag{3.5}$$

In other words, this axiom expresses that if $\Psi[\bar{a}]$ is true in the real world then $\bar{a}$ is one of the answers of the query $\Psi[\bar{x}]$ in the database instance D. Thus, the logical semantics of a database $(D, \mathcal{C})$ (assuming soundness of database tables) is the theory consisting of DCA and UNA, the database atoms and the above formulas. Note that the problem of whether such a database conveys CWI on a query is reduced to the problem of answering the query using the views in $\mathcal{C}$ (Levy et al. 1995).

As the following example shows, completeness constraints are semantically different than our LCWA in the sense that there exist LCWA that cannot be expressed as completeness constraints and vice versa.

Example 3.15 *Consider $\mathcal{LCWA}(P, Q)$. An equivalent (set of) completeness constraint(s) Ψ should have the property that for every database instance D, $\mathfrak{D} = (D, \{\mathcal{LCWA}(P, Q)\})$ and $\mathfrak{D}' = (D, \{\Psi\})$ are equivalent. Take $D = \{\}$, in which case $\mathcal{M}(\mathfrak{D}) \equiv \neg Q \vee \neg P$. There is only one completeness constraint Ψ (modulo equivalence) that makes $\mathfrak{D}'$ equivalent to $\mathfrak{D}$, and this is $\Psi \equiv Q \wedge P$.*

[4] In Motro's formalism formula Ψ is restricted to conjunctive relational calculus expressions (Abiteboul et al. 1995). For the sake of studying the correspondence between the LCWA and completeness constraints we have assumed Ψ to be an arbitrary first-order formula.

Let's now vary the database D such that $D = \{P\}$. Then $\mathcal{M}(\mathfrak{D}) \equiv P$, while $\mathcal{M}(\mathfrak{D}')$ contains the axiom $\neg P \vee \neg Q$ which entails, given P, $\neg Q$. Hence, this LCWA and completeness constraint are not equivalent. For the converse, it is easily shown by a similar technique that the completeness constraint $P \wedge Q$ cannot be expressed as one or more LCWA.

A particular case for which it is straightforward to prove that the LCWA and completeness constraints are semantically equivalent is the following:

$$\mathcal{LCWA}(P(\bar{x}), \varphi[\bar{x}]) \text{ and } \mathsf{CC}(P(\bar{x}) \wedge \varphi[\bar{x}]),$$

where $\varphi[\bar{x}]$ contains only base predicates.

Example 3.16 *Consider, for instance, the following LCWA and CC:*

$$\mathcal{LCWA}(Dept(x, y), y = CS) \text{ and } \mathsf{CC}(Dept(x, y) \wedge y = CS).$$

The latter completeness constraint is equivalent to $\mathsf{CC}(Dept(x, CS))$, and so according formula 3.5:

$$\begin{aligned} \mathcal{M}_D(\mathsf{CC}(P(\bar{x}) \wedge \varphi[\bar{x}])) &\equiv \forall \bar{x} \left(P(\bar{x}) \wedge \varphi[\bar{x}] \supset \bigvee\nolimits_{\bar{a} \in \{\bar{a} \mid D \models P(\bar{a}) \wedge \varphi[\bar{a}]\}} \bar{x} = \bar{a} \right) \\ &\equiv \bigwedge\nolimits_{\bar{a} \in \{\bar{a} \mid D \models \varphi[\bar{a}]\}} P(\bar{a}) \supset (P(\bar{a}) \in D) \\ &\equiv \mathcal{M}_D(\mathcal{LCWA}(P(\bar{x}), \varphi[\bar{x}])). \end{aligned}$$

Here, the second equivalence follows from the CWI on $\varphi[\bar{x}]$. Thus, both databases are equivalent.

We can compare the merits of both types of local forms of the CWA in terms of expressive power. It appears from the above discussion that LCWA cannot be expressed as completeness constraints if they have a window of expertise without CWI. This is demonstrated by the following two LCWAs:

$$\begin{aligned} &\mathcal{LCWA}(CarO(x, y, z), Loc(x, Bx)), \\ &\mathcal{LCWA}(Loc(x, y), (x = Bx \wedge \exists u, v : CarO(u, v, z))) \end{aligned}$$

If these are the only LCWA expressions, there is CWI for neither of these windows of expertise. LCWA like these cannot be expressed by completeness constraints. Vice versa, an interesting example of a completeness constraint that cannot be expressed as a LCWA would be in the context of a database with a relation $Tel(x, y, z)$ representing that x works in department y and has telephone number z, and the completeness constraint $CC(\exists z Tel(x, CS, z))$. It expresses that the database knows at least one telephone number for each person of the computer science department (CS) that has a telephone. Such a completeness constraint cannot be expressed as a LCWA.

The discussion above suggests that it might be interesting to combine the two sorts of local CWA's. In such a generalized formalism, LCWA may have the following form:

$$\mathcal{LCWA}(\varphi[\bar{x}], \Psi[\bar{x}]),$$

where φ and Ψ are first-order formulas. Its logical semantics is given by

$$\forall \bar{x} \left(\Psi[\bar{x}] \supset \left(\varphi[\bar{x}] \supset \bigvee_{\bar{a} \in \{\bar{a} | D \models \varphi[\bar{a}]\}} \bar{x} = \bar{a} \right) \right).$$

This generalizes both LCWA and completeness constraints. An example that could be dealt with in this generalized formalism and in neither of the original ones is a variant of the telephone example. Assume that we have split up the predicate $Tel(x, y, z)$ in two predicates $Dep(x, y)$ and $Tel(x, z)$. In this case, we can express that at least one telephone number is known for a person of CS by the following generalized LCWA:

$$\mathcal{LCWA}\big(\exists y Tel(x, y), Dep(x, CS)\big).$$

3.6.1.3 Closed-World Reasoning for Planning

The sort of local CWA as proposed by Motro, was later used in other contexts as well. In (Etzioni et al. 1997), Etzioni et al. use the same notion of local CWA as introduced by Motro, but in the context of planning problems in agents. The information contained in the planner agent consists of a consistent and correct set L of (positive or negative) literals and a collection of completeness constraints. A polynomial approximate (sound but incomplete) algorithm is presented to compute certain answers for queries in the form of conjunctions of atoms. It is in this paper that the term *Closed-World Information* (CWI) is introduced. A difference with Motro's approach is the use of a set L of literals rather than a database instance D. However, L can be translated in the framework by Motro by having a database instance D consisting of the positive literals and by extending $\mathcal{C}$ with the completeness constraints $\{P(\bar{a}) \mid \neg P(\bar{a}) \in L\}$.

3.6.1.4 Doherty's Circumscriptive Approach

Doherty et al. (Doherty et al. 2000) also study a generalized form of completeness constraints in the context of planning. Their work extends the work of Etzioni et al. by considering arbitrary first-order queries in $\mathcal{C}$. They introduce a circumscriptive semantics for knowledge-bases consisting of a set L of literals and a set $\mathcal{C}$ of completeness constraints. The corresponding semantics of such a knowledge-base consists of DCA, UNA, the set L, and a circumscription axiom minimizing the answers to the queries in $\mathcal{C}$. It is shown that for constraints in the form of existentially quantified conjunctions of atoms, their circumscriptive

semantics coincides with that of Etzioni et al.'s. Other results are that for certain types of completeness constraints, in particular for semi-Horn clauses, the circumscriptive axiom can be translated in a fixpoint query, allowing to solve queries in polynomial time. In (Doherty et al. 2006), this approach is further extended to *rough databases* with integrity constraints.

The above approach of defining the logical semantics of a completeness constraint $\mathsf{CC}(\Psi[\bar{x}])$ by the formula in (3.5)

$$\forall \bar{x} \ \big(\Psi[\bar{x}] \supset \bigvee_{\bar{a} \in \{\bar{a} | D \models \Psi[\bar{a}]\}} \bar{x} = \bar{a}\big)$$

is extendible to the case where Ψ is an arbitrary first-order formula. It is easy to see that when Ψ is a positive formula, the first-order formalism and the circumscriptive-based approach coincide. Indeed, the minimal answer to a positive query is the one obtained in the smallest model, and this is given by D (or by the structure given by the positive literals in L). However, for non-positive queries, the two approaches strongly deviate. Consider for example the completeness constraint $\mathsf{CC}(\neg P(x))$. The FO logical semantics for this constraint is:

$$\forall \bar{x} \ \Big(\neg P(\bar{x}) \supset \bigvee_{P(\bar{a}) \notin D} \bar{x} = \bar{a}\Big).$$

The tuples $\bar{d}$ that falsify

$$\bigvee_{P(\bar{a}) \notin D} \bar{d} = \bar{a},$$

are exactly those for which $P(\bar{d}) \in D$. It follows that, given the DCA, the above formula is equivalent to :

$$\forall \bar{x} \ \Big(P(\bar{x}) \supset (P(\bar{x}) \in D)\Big).$$

Surprisingly, this constraint restates that D is sound in P and is entailed by the soundness assumption. Now, according to the circumscriptive approach in this case, $\neg P(x)$ is minimized, which entails that P is true for all domain elements. This shows that Doherty et al.'s circumscriptive semantics for general completeness constraints does not comply with the original intuition of Motro and Etzioni et al., which is that all answers of a completeness constraint in the real world can be computed from the database.

3.6.1.5 Levesque Only-Know Logic

Our approach is also tightly linked with Levesque's work on reasoning on incomplete knowledge-bases (Levesque 1982; Levesque 1990). In these papers, Levesque presents a framework of first-order knowledge-bases KB (i.e., KB is a first-order theory) and modal logic queries. In this framework, $\mathbf{K}\mathcal{Q}[\bar{x}]$ denotes

the query for the certain answers of $\mathcal{Q}[\bar{x}]$ and $\neg\mathbf{K}\neg\mathcal{Q}[\bar{x}]$ is the query for the possible answers of $\mathcal{Q}[\bar{x}]$. Such a modal formula is evaluated using modal S5 semantics (Chellas 1980) with respect to the possible world structure $W(KB)$ consisting of all models of KB.

Levesque also studies the use of modal logic to assert completeness or incompleteness on a first-order knowledge base. For instance, the statement

$$\forall x \forall y(Dept(x, CS) \supset (Tel(x, y) \supset \mathbf{K}Tel(x, y)))$$

expresses that we know the telephone numbers of people from the CS department. He continues by warning that we should not just extend KB with such formulas, as the result of this would be a modal logic knowledge-base with several unpleasant features (see Section 6 of (Levesque 1982) for a thorough discussion on this issue). The solution he proposes is to *reduce* such a modal formula to a first-order formula in the context of KB. In particular, given a modal formula $\alpha[\bar{x}]$, there is a first-order formula $\beta[\bar{x}]$ such that

$$KB \models \forall \bar{x}(\alpha[\bar{x}] \equiv \beta[\bar{x}]),$$

where $\models$ means truth in $W(KB)$. Levesque also presents an algorithm to compute β from KB and α. Interestingly, if we assume that KB consists of $\mathsf{DCA} \wedge \mathsf{UNA} \wedge \bigwedge_{P(\bar{a}) \in D} P(\bar{a})$, then applying this reduction algorithm on a formula of the form:

$$\forall x \forall y\big(Dept(x, CS) \supset (Tel(x, y) \supset \mathbf{K}Tel(x, y))\big)$$

we obtain (after simplification using DCA):

$$\forall x \forall y\big(Dept(x, CS) \supset (Tel(x, y) \supset (Tel(x, y) \in D))\big)$$

which of course is the logical semantics of

$$\mathcal{LCWA}(Tel(x, y), Dept(x, CS))$$

This observation extends to arbitrary LCWA. Thus, our approach coincides with (a fragment of) Levesque's formalism.

3.6.1.6 Mediator-Based Systems

In the context of the integration of distributed databases[5] there are also several approaches to apply local forms of closure and local forms of soundness axioms. In (Grahne and Mendelzon 1999a), Grahne and Mendelzon study integrated databases using the local-as-view approach. The integrated database consists of a set of local databases and a set of conjunctive queries defining source predicates in terms of the global schema. Local predicates can be declared as

[5] In the next section we formally describe mediatorbased systems.

either open (sound), closed (complete) or clopen (sound and complete). Grahne and Mendelzon present a coNP-algorithm for determining consistency of such databases and for computing certain and possible answers of global queries. In (Duschka 1997), it is studied how query answering in LAV databases can be optimized using more general completeness constraints on the sources.

3.6.2 Deductive Databases and Logic Programming

The area of deductive databases is tightly connected with logic programming. In this context, several approaches have been proposed to relax the CWA. Below, we discuss the most relevant ones.

3.6.2.1 Answer set programming.

In (Gelfond and Lifschitz 1991), Gelfond and Lifschitz 'extend' the formalism of *general logic programming* by dropping the closed-world assumption and allowing *classical negation* to appear in heads and bodies of rules. One of the main motivations behind this extension was the necessity in many applications of relaxing the closed-world assumption to represent partially complete knowledge[6]. Whereas in general logic programming the CWA means falsity of all atoms that are not entailed by the premises, in answer set programming falsity of atoms needs to be represented explicitly by rules with negated literals in their head. This extension allows to represent local forms of closed-world assumption in a similar way as the one considered in our work. Consider, for instance, the following program rule:

$$\neg p(\bar{x}) \leftarrow \mathsf{not}\ p(\bar{x}), \varphi.$$

Under stable-models semantics (Gelfond and Lifschitz 1988), the truth of $\neg p(\bar{x})$ is established unless there is an evidence that $p(\bar{x})$ is true (i.e., $p(\bar{x})$ is not in the database) and φ is provable. It follows that, in our terminology, φ is the window of expertise of predicate p. The semantics of an answer set program is given by stable models, which are (consistent) sets of literals.

3.6.2.2 Other generalized forms of closed-world assumptions.

In (Minker 1982), Minker introduced disjunctive databases as an extension of deductive databases for representing incomplete knowledge. In disjunctive databases, disjunctions may appear in the heads of the rules. As noted already by Reiter in (Reiter 1978), such expressions may be inconsistent with the CWA. For instance, since the disjunctive database $\{q \vee p\}$ neither entails p nor q, the CWA would entail both $\neg p$ and $\neg q$ which of course is inconsistent with the database. Minker (Minker 1982) addressed this problem by introducing

[6] Personal communication with M. Gelfond

the *Generalized Closed-World Assumption* (GCWA), which specifies that a negated atom is considered to be true if the atom does not appear in any minimal model of the database. Note that when a database does not contain disjunctive clauses, then GCWA and CWA coincide. This shows, in particular, that disjunctive databases and locally closed databases are different ways of representing incomplete knowledge, and that the GCWA is a very different principle than the LCWA.

Another extension of the CWA in the context of logic programming is the *Any World Assumption*, introduced by Loyer and Straccia in (Loyer and Straccia 2005). They argue that due to the incompleteness of information, any semantics of logic programming has to rely on a default assumption on the missing information. In case of the stable semantics, this default is *false*, while in case of OWA, the default is *unknown*. Loyer and Straccia view these as two extreme cases and develop a semantics in which the default assumption can be any pre-specified value in an arbitrary truth space. While this approach aims at somewhat similar goals as ours, their context, techniques, and viewpoints, are completely different.

3.7 Related Topics

3.7.1 Safety

A major weakness in both semantic accounts of databases is concerned with the assumption of a fixed (and finite) domain. In reality, a database consists only of tables and never explicitly describes its domain. Even when tables are complete with respect to the relations they store, it will rarely be the case that the "only" objects of the domain of discourse are those mentioned in the database. E.g., a departmental database could contain an entry of all personnel of the department and their addresses, but does not "know" all cities. Similarly, the database may know the telephone numbers of all its members, but not all the telephone numbers of its collaborators (or even some other numbers). Also, databases typically contain natural or real numbered attributes, but there are infinitely more of those objects than the ones mentioned in the database. Thus, from a semantic point of view, it would be more accurate to model a (complete) database as a *collection* of structures M that interpret all predicates by the tables but have different (and often infinite) domains. The logical account of such a database is given more accurately by dropping the Domain Closure Axiom from $\mathcal{M}_{st}(D)$. The problem with such an improved semantical account is of course that certain queries may have infinite or undetermined answers (e.g., $x = x$). To fence against this problem, it is common to restrict the class of queries to *domain independent* and *finite* queries: queries that have the same finite set of answers in all structures that share the same tables. The restriction to this class of queries blends perfectly with another practical concern,

namely that database (first-order-)queries are solved using relational algebra and therefore, need to be *safe* – a condition which guarantees domain independence and finiteness (Abiteboul et al. 1995). For such queries, a finite domain Dom^D can be determined from the database tables, the integrity constraints, the query and, in this thesis, the collection of local closed-world assumptions, such that the answer to a query $\mathcal{Q}[\bar{x}]$, i.e., the set $\{\bar{a} \in (Dom^D)^n \mid D \models \mathcal{Q}[\bar{a}]\}$, is finite. We will return to this issue in Section 4.5.

3.7.2 An Application of the LCWA to Mediator-Based Systems

A mediator-based system (sometimes called data integration system) consists of a global database schema unifying a number of heterogeneous databases that store information about a common domain of discourse (see (Grahne 2002; Grahne and Mendelzon 1999a; Lenzerini 2002; Levy et al. 1996; Van Nuffelen et al. 2004)). The semantical relations between the global schema and the data sources are usually defined by means of so-called schema mappings. These are formulas restricted to an ad-hoc sub-language of first-order logic. To retrieve answers, the user poses queries in terms of the global schema. A planner, using the knowledge encoded in the schema mappings, reformulates the global query in terms of the data sources, from which the answers are extracted.

In this section we demonstrate how the concept of the local closed-world assumption can be applied to mediator-based systems. This is motivated by the fact that, often, each data source collaborating in a mediator system is complete with respect to (part of) its own domain (but incomplete with respect to the larger domain of the mediator). The idea then is that closed-world assumption expressed at the level of the data sources could allow to derive 'negative' information at the level of the mediator.

For instance, it is common in bioinformatics domains the integration of data repositories that store information about experiment with certain microorganism (yeast). Each experiment is carried out under a different condition. Local databases are then experts on the specific setting in which the experiment took place, but they may be ignorant about the results of other experiments. The mediator then integrates all the local sources, and local closure is 'propagated' to the mediator by means of the schema mappings. The biologist can then have a broader view on the studied microorganism while retaining complete knowledge about the specific experiments.

Definition 3.15 *A* mediator-based system *based on a relational vocabulary* $\Sigma = \Sigma_G \cup \Sigma_{\mathfrak{D}_1} \cup \cdots \cup \Sigma_{\mathfrak{D}_n}$ *is a tuple* $\mathfrak{M} = \langle G, S, M \rangle$*, where*

- Global schema G*: is an empty first-order theory based on a* global *relational vocabulary* Σ_G*.*

- Local sources S: *is a set of locally closed databases* $S = \{\mathfrak{D}_1, \ldots \mathfrak{D}_n\}$ *based on vocabularies* $\Sigma_{\mathfrak{D}_1}, \ldots, \Sigma_{\mathfrak{D}_n}$, *respectively.*

- Schema mappings M: *is a first-order theory based on vocabulary* Σ *that describes the ontological relationships between the global schema* Σ_G *and the local sources.*

We now describe in some detail each of these components:

- **Global schema:** It corresponds to the global ontology of the system. User queries are posed in terms of this schema but no instance is assumed to be stored at this level.

- **Local sources:** These are independent locally complete databases that supply data to the mediator. They participate as independent entities and may leave the mediator system at will.

- **Schema mapping:** It is the logical theory that formalizes the logical relationships between the global and the local ontologies. Depending on the dynamics of the mediator system, formulas of the schema mapping may be syntactically restricted. We briefly mention the most common restrictions:

  – *Local-as-View (LAV)*: This is the preferred approach for dynamic systems in which the local sources enter and leave the system constantly. It consists of a number of formulas of the form

  $$\forall \bar{x} P_i(\bar{x}) \equiv \Psi_{p_i}(\bar{x}),$$

  where P_i is a predicate of a local source and Ψ_{p_i} is a first-order formula in terms of the global vocabulary Σ_G. This expression can be viewed as definitional knowledge, and its semantics has also been given in terms of ID-Logic definitions (Van Nuffelen et al. 2004). In order to maintain tractability of query answering, Ψ_{p_i} is often restricted to conjunctions of positive literals (Lenzerini 2002).

  – *Global-as-View (GAV)*: It is the dual of the LAV approach. Often, GAV is applied in contexts where the local sources collaborate with the system during long periods of time. Each GAV expression is a formula of the form

  $$\forall \bar{x} P_G(\bar{x}) \equiv \Psi_{P_G}(\bar{x}),$$

  where P_G is a predicate of the global schema and Ψ_{P_G} is a first-order formula expressed in terms of the local vocabularies. As in the case of the LAV approach, the formulas Ψ_{P_G} are commonly restricted to conjunction of positive literals.

– *Global-Local-as-view (GLAV)(Pottinger and Levy 2000)*: This extension of LAV allows formulas of the form

$$\forall \bar{x} \psi_L(\bar{x}) \equiv \phi_G(\bar{x}),$$

where ψ and ϕ_G are conjunctions of positive literals to LAV .

The semantics of a data integration system $\mathfrak{M}$ is defined in terms of the theory $\Gamma_{\mathfrak{M}} = G \cup S \cup M$ [7]. In general, $\Gamma_{\mathfrak{M}}$ is an incomplete theory. This is to be expected, given the incomplete information in the data sources and the nature of the global schema. Query answers are defined as in the case of locally complete databases (recall Definition 3.12), but with respect to $\Gamma_{\mathfrak{M}}$ instead of $\mathfrak{D}$. Queries, however, can only be expressed in terms of the global schema.

The following simple example shows how a mediator-based system can interact with locally complete databases as local sources.

Example 3.17 *Consider the Σ-based mediator-based system $\mathfrak{M} = \langle G, S, M \rangle$, where $S = \{\mathfrak{D}_1, \mathfrak{D}_2\}$:*

$$\mathfrak{D}_1 = (\{R_1(a,b)\}, \mathcal{LCWA}_{\mathfrak{D}_1}(R_1(x,y), x = c)),$$
$$\mathfrak{D}_2 = (\{R_2(e,f), R_2(b,g)\}, \mathcal{LCWA}_{\mathfrak{D}_2}(R_2(x,y), y = e)),$$

and

$$M = \{\forall x, y \; G(x,y) \equiv \exists z R_1(x,z) \wedge R_2(z,y)\}$$

Data source $\mathfrak{D}_1$ is an expert on R_1 with respect to its first attribute regarding c, while $\mathfrak{D}_2$ is an expert on the second attribute of R_2 with respect to e.

The theory of the mediator consists of the formulas $\Gamma_{\mathfrak{M}} = \mathcal{M}(\mathfrak{D}_1) \cup \mathcal{M}(\mathfrak{D}_2) \cup M$. The following queries and their answers illustrate the behavior of the mediator:

- $Cert_{\Gamma_{\mathfrak{M}}}(G(x,y)) = \{(a,g)\}$. *Indeed, in every model M of $\Gamma_{\mathfrak{M}}$, $G(a,g)$ must be true.*

- $Poss_{\Gamma_{\mathfrak{M}}}(G(m,n)) =$ yes, *since the conjunction may be true according to what is stored in $\mathfrak{D}_1$ and $\mathfrak{D}_2$.*

- $Poss_{\Gamma_{\mathfrak{M}}}(G(c,e)) =$ no. *Because $R_1(c,x)$ is false (for any arbitrary x) in every model M of $\Gamma_{\mathfrak{M}}$, and hence the conjunction in M can never be true.*

[7] Here, S represents the union of all the semantics of the locally closed databases in the system.

3.8 Bibliographical Note

The literature on incomplete databases and more in general on incomplete knowledge bases is abundant and diverse. Different contexts call for different solutions, and probably no single approach is the best in every situation. In section 3.6 we have presented related works that are close in spirit to ours. However, there are other types of incomplete knowledge in databases that deserve at least a brief mention.

A well-known source of databases incompleteness are the so-called null values. A null value is a constant that replaces an unknown value in the database. Alternative semantics have been proposed for null values and for databases that contain them, and there seems to be no consensus about what a null value really is. However, it is common practise to consider two null values as representing two different constants of the domain of discourse. As a consequence, null values are usually exempted from the unique name axiom. Paradigmatic work on null values is the work by Reiter (Reiter 1986), where an influential semantics to databases with null values is introduced. Libkin (Libkin 1998) focuses on query answering in the presence of null values, and introduces a mechanism based on universal algebra with databases that contain null values. Bertossi and Bravo (Bravo and Bertossi 2006) provide semantics to databases that contain null values and that are inconsistent with respect to a given set of integrity constraints.

Chapter 4

Approximative Reasoning for the LCWA

4.1 Introduction

In Section 3.5 we studied the computational complexity of different reasoning tasks in locally closed databases. In the general case, query answering for either certain or possible answers turns out to be intractable. This fact have provided the motivation for developing *efficient approximate methods for query answering*. This approach is also motivated by the fact that in many applications there is no need to have all the answers to a query and often it is enough to have sufficiently large subsets of them. For instance, a company searching in an (incomplete) database for a provider of some urgently required service will be satisfied by finding some candidate providers, so an exhaustive search for all the providers is not always needed.

In this chapter, we provide query answering techniques based on approximations of all the two-valued models of a locally closed database, using three-valued structures. To assess the accuracy of this techniques, we provide a number of conditions that guarantee optimal answers to queries. More specifically, in this chapter:

- We present the logical foundations of three-valued logic for query approximations. An important result of our analysis is a fixpoint operator that induces a mechanism to construct an approximation of a locally complete database $\mathfrak{D}$. The computed approximation can then be queried to retrieve approximations of certain or possible answers.

- To tackle two important drawbacks of explicitly computing a three-valued structure, namely storing a large structure and updating it every time the database changes, we provide an algorithm based on fixpoint logic

that allows to retrieve answers while leaving the three-valued structure encoded in a new transformed query.

- Certain types of integrity constraints are integrated in the query answering process. In particular, primary key constraints can be incorporated into the framework.

- A thorough analysis of the optimality of approximative query answering is developed. It shows that our approach retrieves correct and complete answers for a considerably large and important class of queries and locally closed databases.

This chapter is organized as follows:

- In Section 4.2, we present the basic three-valued logic machinery that supports our approximation algorithms. In particular, we show how to simulate and query three-valued structures using standard two-valued ones. In Section 4.2.2, we define the concept of approximation of a first-order theory and adapt the notions of certain and possible answers to the three-valued setting.

- In Section 4.3, we apply the generic three-valued approximation method to the locally complete databases setting. The first approach consists of computing a three-valued approximation of all the models of a locally complete database. This method is then refined in Section 4.3.1, where we present a fixpoint approach that allows to leave the three-valued interpretation implicit. In Section 4.3.4, we study a specific but important class of databases called hierarchically closed. In such databases the query answering process in the three-valued approximation can be simulated using purely first-order queries posed against the initial database instance. The framework is extended with functional dependencies in Section 4.3.5.

- In Section 4.4, we investigate conditions on queries and locally closed databases that guarantee that the answers obtained by the approximation approach are optimal. We conclude the chapter with a summary and a discussion of the obtained results.

4.2 Approximative Reasoning

The basic idea of approximative reasoning in the context of the local closed-world assumption is to compute a three-valued structure that provides a "good" approximation of all models of $\mathfrak{D}$ and then to evaluate queries with respect to this approximation. In the next sections, we will present a (polynomial) algorithm to compute such an approximation structure and we will show how

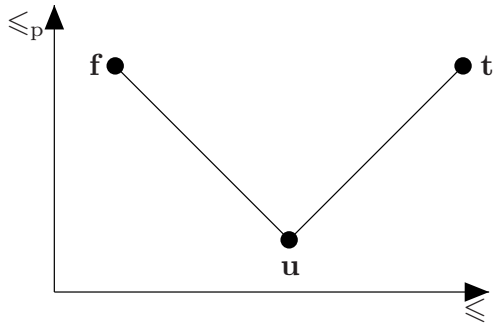


Figure 4.1: Truth and precision orders in three-valued logic.

we can use this to obtain approximative answers for queries. Then we will focus on two obvious weaknesses of this approach.

The first weakness is the need to recompute the three-valued approximation every times the database changes. Indeed, if the above approach is applied naively, then after each modification a new approximating structure should be computed from scratch. Clearly, this is not practically feasible. As a solution we will develop a technique based on fixpoint queries that allows us to leave the three-valued approximating structure implicit, so that we can compute answers with respect to it, using only the content of the database instance and symbolic reasoning on the LCWAs.

The second weakness is related to the precision of the approximate reasoning. We somehow need to prove that, at least in useful cases, the algorithm computes sufficiently precise approximations. In Section 4.4 we will do this analysis and show that for some broad classes of locally closed databases, our methods provide optimal answers, that is, answers as defined in Section 3.5. This will also imply that for these classes of locally closed databases, 'standard' query answering (as defined in Section 3.5) is solvable in a tractable manner.

4.2.1 Three-Valued Structures

We first review the main concepts of three-valued logics needed in the sequel. The truth values of three-valued semantics $\mathcal{THREE} = \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, respectively standing for true, false and unknown, are usually arranged in two orders: the truth order, $\leqslant$, which is a linear order given by $\mathbf{f} \leqslant \mathbf{u} \leqslant \mathbf{t}$, and the precision order $\leqslant_p$, which is a partial order on $\mathcal{THREE}$ in which $\mathbf{u}$ is the least element, and $\mathbf{t}$ and $\mathbf{f}$ are incomparable maximal elements. The structure of $\mathcal{THREE}$ is drawn in Diagram 4.1.

The conjunction $\wedge$ in $\mathcal{THREE}$ is defined by the $\leqslant$-glb of this structure, the disjunction $\vee$ is defined by the $\leqslant$-lub, and the negation operator $\neg$ is associated with the $\leqslant$-involution, that is: $\neg\mathbf{t} = \mathbf{f}$, $\neg\mathbf{f} = \mathbf{t}$, and $\neg\mathbf{u} = \mathbf{u}$.

A three-valued relation (of arity n) on some domain Dom is a function from

Dom^n to $\mathcal{THREE}$. A three-valued structure $\mathcal{K}$ of vocabulary Σ consists of a domain $Dom(\mathcal{K})$, for each constant symbol $c \in \mathcal{C}(\Sigma)$ an appropriate domain element $c^{\mathcal{K}} \in Dom(\mathcal{K})$, and for each predicate symbol $P \in \mathcal{R}(\Sigma)$ a three-valued relation $P^{\mathcal{K}}$ in $Dom(\mathcal{K})$. The predicates $\mathbf{t}$, $\mathbf{f}$ and $=$ are still interpreted in the standard way. A structure $\mathcal{K}$ is called *two-valued* if $\mathbf{u}$ is not in the range of its relations $P^{\mathcal{K}}$.

Example 4.1 *Consider a relational vocabulary Σ, where $\mathcal{R}(\Sigma) = \{P, Q\}$ and $\mathcal{C}(\Sigma) = \{a, b\}$. The following corresponds to a three-valued structure ($\mathcal{K}$) of Σ:*

- $Dom(\mathcal{K}) = \{a, b, c\}$.
- $a^{\mathcal{K}} = a,\ b^{\mathcal{K}} = b,\ c^{\mathcal{K}} = c$.

$$P^{\mathcal{K}}(a) = \mathbf{t},\ P^{\mathcal{K}}(b) = \mathbf{u},\ P^{\mathcal{K}}(c) = \mathbf{t}$$
$$Q^{\mathcal{K}}(a) = \mathbf{f},\ Q^{\mathcal{K}}(b) = \mathbf{t},,\ Q^{\mathcal{K}}(c) = \mathbf{t}$$

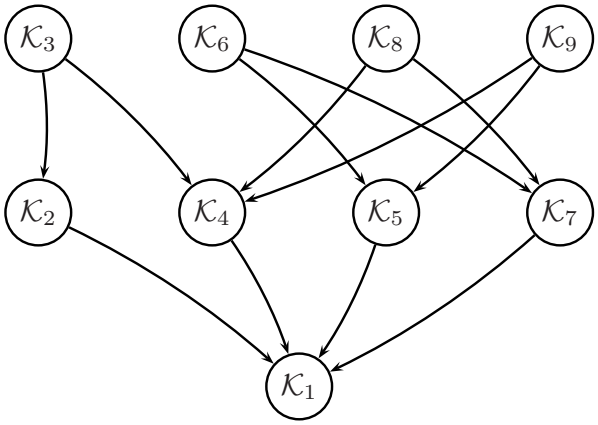
If we change the interpretation of $P^{\mathcal{K}}(b)$ to either $\mathbf{t}$ or $\mathbf{f}$, then $\mathcal{K}$ becomes a standard two-valued structure.

The concept of a Herbrand structure generalizes to the three-valued case. An equivalent representation of a three-valued Herbrand structure is as a function from the Herbrand base $HB(\Sigma)$, the set of ground atoms of Σ, to $\mathcal{THREE}$. The collection of three-valued Herbrand interpretations of Σ is denoted $\mathfrak{L}^c_\Sigma$ and its subset of two-valued structures is denoted $\mathfrak{L}_\Sigma$. We will drop Σ when clear from the context. On the set $\mathfrak{L}^c$, the truth order $\leqslant$ and precision order $\leqslant_{\mathrm{p}}$ are defined by a pointwise extension of the corresponding orders in $\mathcal{THREE}$. In $\mathfrak{L}^c$, $\leqslant$ is a lattice order and $\leqslant_{\mathrm{p}}$ is a chain-complete order[1]. It is well-known that a monotone operator on a chain-complete poset has a least fixpoint. This property will be exploited in the following section.

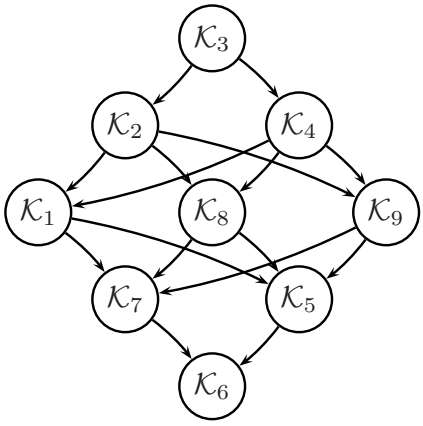
Example 4.2 *Consider the following three-valued structures $\mathcal{K}_1, \ldots, \mathcal{K}_9$ with vocabulary Σ such that $\mathcal{R}(\Sigma) = \{P, Q\}$ and $\mathcal{C}(\Sigma) = \{a\}$, where $Dom(\mathcal{K}_i) = \{a\}$, $a^{\mathcal{K}_i} = a$, $i = 1, \ldots, 9$, and*

$$\mathcal{K}_1 = \{P^{\mathcal{K}_1}(a) = \mathbf{u},\ Q^{\mathcal{K}_1}(a) = \mathbf{u}\} \qquad \mathcal{K}_2 = \{P^{\mathcal{K}_2}(a) = \mathbf{t},\ Q^{\mathcal{K}_2}(a) = \mathbf{u}\}$$
$$\mathcal{K}_3 = \{P^{\mathcal{K}_3}(a) = \mathbf{t},\ Q^{\mathcal{K}_3}(a) = \mathbf{t}\} \qquad \mathcal{K}_4 = \{P^{\mathcal{K}_4}(a) = \mathbf{u},\ Q^{\mathcal{K}_4}(a) = \mathbf{t}\}$$
$$\mathcal{K}_5 = \{P^{\mathcal{K}_5}(a) = \mathbf{f},\ Q^{\mathcal{K}_5}(a) = \mathbf{u}\} \qquad \mathcal{K}_6 = \{P^{\mathcal{K}_6}(a) = \mathbf{f},\ Q^{\mathcal{K}_6}(a) = \mathbf{f}\}$$
$$\mathcal{K}_7 = \{P^{\mathcal{K}_7}(a) = \mathbf{u},\ Q^{\mathcal{K}_7}(a) = \mathbf{f}\} \qquad \mathcal{K}_8 = \{P^{\mathcal{K}_8}(a) = \mathbf{t},\ Q^{\mathcal{K}_8}(a) = \mathbf{f}\}$$
$$\mathcal{K}_9 = \{P^{\mathcal{K}_9}(a) = \mathbf{f},\ Q^{\mathcal{K}_9}(a) = \mathbf{t}\}$$

Here, $\mathfrak{L}^c_\Sigma = \{\mathcal{K}_1, \ldots, \mathcal{K}_9\}$ and $\mathfrak{L}_\Sigma = \{\mathcal{K}_3, \mathcal{K}_6, \mathcal{K}_8, \mathcal{K}_9\}$. These are graphically represented in Figure 4.2.



(a) Chain-complete order $\leqslant_p$ on $\mathfrak{L}^c_\Sigma$.



(b) Lattice order $\leqslant$ on $\mathfrak{L}^c_\Sigma$.

Figure 4.2: $\leqslant_p$ and $\leqslant$ orders on $\mathfrak{L}^c_\Sigma$.

The truth assignment for compound sentences can be extended to three-valued structures in several ways. In this thesis, we will discuss two of them: the standard *Kleene truth assignment* and *supervaluation* (see below). The standard Kleene truth assignment (Kleene 1952) is defined through similar recursive rules as in the case of the two-valued truth assignment:

$$(\psi \wedge \phi)^{\mathcal{K}} = Min_{\leqslant}(\psi^{\mathcal{K}}, \phi^{\mathcal{K}});$$
$$(\neg\psi)^{\mathcal{K}} = (\psi^{\mathcal{K}})^{-1};$$
$$(\forall x\psi[x])^{\mathcal{K}} = Min_{\leqslant}\{(\psi[a])^{\mathcal{K}} \mid a \in Dom(\mathcal{K})\};$$

Example 4.3 *The following corresponds to Kleene evaluation on some of the three-valued structures of Example 4.2:*

- $(P(a) \wedge Q(a))^{\mathcal{K}_1} = Min_{\leqslant}(P(a)^{\mathcal{K}_1}, Q(a)^{\mathcal{K}_1}) = \mathbf{u}$.
- $(P(a) \wedge Q(a))^{\mathcal{K}_4} = Min_{\leqslant}(P(a)^{\mathcal{K}_4}, Q(a)^{\mathcal{K}_4}) = \mathbf{u}$.
- $(Q(a) \vee \neg Q(a))^{\mathcal{K}_5} = Max_{\leqslant}(P(a)^{\mathcal{K}_5}, \neg Q(a)^{\mathcal{K}_5}) = \mathbf{u}$

Observe that although $(Q(a) \vee \neg Q(a))$ is a valid formula and hence always 'true' under standard two-valued evaluation, under Kleene evaluation its truth value is 'unknown'.

A well-known and, in this thesis, useful result, is that Kleene structures can be simulated by two-valued structures. For a vocabulary Σ, define a vocabulary Σ' as the set of all constant symbols of Σ and, for each predicate $P \in \mathcal{R}(\Sigma)$, the predicate symbols P^c and $P^{c\neg}$ (intuitively, P^c stands for *certainly* P and $P^{c\neg}$ for *certainly not* P). Now,

Definition 4.1 *Let $\mathcal{K}$ be a three-valued Σ-structure. We say that a 2-valued Σ'-structure I* simulates *$\mathcal{K}$, if and only if $\mathcal{K}$ and I have the same domain and assign the same interpretations to constant symbols, and for each predicate $P \in \mathcal{R}(\Sigma)$, it holds that $(P^c)^I = \{\bar{d} \mid P(\bar{d})^{\mathcal{K}} = \mathbf{t}\}$ and $(P^{c\neg})^I = \{\bar{d} \mid P(\bar{d})^{\mathcal{K}} = \mathbf{f}\}$.*

Now, given a Σ-formula $\varphi[\bar{x}]$, let $\varphi[\bar{x}]^c$ be the Σ'-formula obtained by substituting $P^c(\bar{t})$ for positive occurrences of atoms $P(\bar{t})$ and substituting $\neg P^{c\neg}(\bar{t})$ for negative occurrences of atoms $P(\bar{t})$, for all $P \in \mathcal{R}(\Sigma)$ [2]. The following proposition is well-known.

[1] In a chain complete order, each chain, i.e. each totally ordered subset, has a least upper-bound.

[2] Equivalently but in other words, $\varphi[\bar{x}]^c$ is obtained by substituting $P^c(\bar{t})$ for positive atoms $P(\bar{t})$ and $P^{c\neg}(\bar{t})$ for negative atoms $\neg P(\bar{t})$.

Proposition 21 *If I simulates $\mathcal{K}$, then for each formula $\varphi[\bar{x}]$ and suitable tuple of domain elements $\bar{d}$, it holds that $\varphi[\bar{d}]^{\mathcal{K}} = \mathbf{t}$ if and only if $(\varphi[\bar{d}]^c)^I = \mathbf{t}$ and $\varphi[\bar{d}]^{\mathcal{K}} = \mathbf{f}$ if and only if $((\neg\varphi[\bar{d}])^c)^I = \mathbf{t}$.*

Example 4.4 *Consider a vocabulary Σ and the three-valued interpretation $\mathcal{K}_2$ from example 4.2. We can construct an interpretation I that simulates $\mathcal{K}_2$ following definition 4.1. Indeed:*

- $\mathcal{R}(\Sigma') = \{P^c, P^{c\neg}, Q^c, Q^{c\neg}\}$; $\mathcal{C}(\Sigma') = a$.
- $Dom(I) = Dom(\mathcal{K}_2) = \{a\}$.
- $a^I = a^{\mathcal{K}_2} = \{a\}$.
- $(P^c)^I = \{a\}$, $(P^{c\neg})^I = \emptyset$, $(Q^c)^I = \emptyset$, $(Q^{c\neg})^I = \emptyset$.

We can now use interpretation I to answer queries posed to $\mathcal{K}_2$. Consider $\varphi = (P(a) \vee \neg Q(a))$. Applying the query transformation on φ, we obtain $(\varphi^c)^I = (P^c(a) \vee Q^{c\neg}(a))^I = \mathbf{t}$. Indeed, as stated in Proposition 21, $\varphi^{\mathcal{K}_2} = \mathbf{t}$.

As a corollary, we obtain tractability of three-valued truth evaluation and query answering.

Corollary 1 *Let $\mathcal{K}$ be a finite three-valued Σ-structure. Then for each Σ-formula $\varphi[\bar{x}]$, the sets $\{\bar{d} \mid (\varphi[\bar{d}])^{\mathcal{K}} = \mathbf{t}\}$, $\{\bar{d} \mid (\varphi[\bar{d}])^{\mathcal{K}} = \mathbf{f}\}$ and $\{\bar{d} \mid (\varphi[\bar{d}])^{\mathcal{K}} = \mathbf{u}\}$ can be computed in polynomial time in the size of $\mathcal{K}$.*

A crucial monotonicity property of Kleene truth assignment is that $\mathcal{K} \leqslant_{\mathrm{p}} \mathcal{K}'$ implies $\varphi^{\mathcal{K}} \leqslant_{\mathrm{p}} \varphi^{\mathcal{K}'}$, for every sentence φ.

An alternative truth assignment satisfying the same property is *supervaluation* (van Fraassen 1966).

Definition 4.2 (Supervaluation) *The supervaluation $sv_{\mathcal{K}}(\varphi)$ of a sentence φ in a three-valued structure $\mathcal{K}$ is defined as*

$$sv_{\mathcal{K}}(\varphi) = glb_{\leqslant_{\mathrm{p}}}\{\varphi^M \mid M \in \mathfrak{L} : \mathcal{K} \leqslant_{\mathrm{p}} M\}.$$

Example 4.5 *Consider again the query $\mathcal{Q} = Q(a) \vee \neg Q(a)$ of Example 4.3. Under Kleene truth assignment, its value is $\mathbf{u}$. Under supervaluation, however, $sv_{\mathcal{K}_5}(Q(a) \vee \neg Q(a)) = \mathbf{t}$.*

It is possible to prove that for each $\mathcal{K}$ and φ, $\varphi^{\mathcal{K}} \leqslant_{\mathrm{p}} sv_{\mathcal{K}}(\varphi)$. In terms of complexity, the price for this augmented precision is that for a given formula φ, deciding whether $sv_{\mathcal{K}}(\varphi) = \mathbf{t}$ for finite $\mathcal{K}$, is coNP, and is coNP-hard for some φ.

4.2.2 Approximations by Three-Valued Structures

Let Σ be a vocabulary consisting of finitely many predicates and constants.

Definition 4.3 *Let Γ be a satisfiable theory based on Σ, containing $\mathsf{UNA}(\Sigma) \wedge \mathsf{DCA}(\Sigma)$[3]. We say that a three-valued Herbrand Σ-interpretation $\mathcal{K}$* approximates *Γ (from below) if and only if for every 2-valued Herbrand model M of Γ, $\mathcal{K} \leqslant_{\mathrm{p}} M$.*

In our context we will be interested in the most precise approximation:

Definition 4.4 (Optimal approximation, $\mathcal{O}_\Gamma$) *The* optimal approximation *$\mathcal{O}_\Gamma$ for Γ is the three-valued Herbrand structure defined by $glb_{\leqslant_{\mathrm{p}}}(\{M \mid M \models \Gamma\})$, where M ranges over all the 2-valued Herbrand models of Γ.*

The structure $\mathcal{O}_\Gamma$ is the most precise of all three-valued Herbrand Σ-structures approximating Γ, and it is well-defined since the set of its Herbrand models is non-empty and every nonempty set $S \subseteq \mathfrak{L}^c$ has a greatest $\leqslant_{\mathrm{p}}$-lower bound.

An alternative characterization of $\mathcal{O}_\Gamma$ is given in the following proposition:

Proposition 22 *For each ground atom $P(\bar{a}) \in HB(\Sigma)$ it holds that $P(\bar{a})^{\mathcal{O}_\Gamma} = \mathbf{t}$ if and only if $\Gamma \models P(\bar{a})$ and $P(\bar{a})^{\mathcal{O}_\Gamma} = \mathbf{f}$ if and only if $\Gamma \models \neg P(\bar{a})$.*

Thus, to obtain the optimal approximation $\mathcal{O}_\Gamma$ for a database $\mathfrak{D} = (D, \mathcal{L})$ we must compute for every predicate P in $\mathcal{R}(\Sigma)$ the certain answers of $P(\bar{x})$ and the certain answers of $\neg P(\bar{x})$. As as straight-forward corollary of Propositions 22 and 17, we obtain the following negative complexity result for computing $\mathcal{O}_\Gamma$.

Corollary 2 *Let $\mathfrak{D} = (D, \mathcal{L})$ be a locally complete database based on vocabulary Σ. Checking whether a three-valued interpretation $\mathcal{K}$ is the optimal approximation $\mathcal{O}_\Gamma$ of $\mathfrak{D}$, is $\mathsf{coNP}-Complete$.*

Since all models of a theory containing $\mathsf{UNA}(\Sigma) \wedge \mathsf{DCA}(\Sigma)$ are isomorphic to a Herbrand structure, the following proposition holds.

Proposition 23 *Let $\mathcal{K}$ be an approximation of Γ. For any sentence φ, if $\varphi^{\mathcal{K}} = \mathbf{t}$, then $\Gamma \models \varphi$ and if $\varphi^{\mathcal{K}} = \mathbf{f}$, then $\Gamma \models \neg\varphi$.*

The inverse, of course, does not hold, not even when $\mathcal{K} = \mathcal{O}_\Gamma$.

Example 4.6 *For a propositional example, let $\Sigma = \{P\}$ and $\Gamma = \emptyset$. The optimal approximation of Γ is $\{P : \mathbf{u}\}$. Consider the following formulas:*

- *$P \vee \neg P$. It holds that $\emptyset \models P \vee \neg P$, while $(P \vee \neg P)^{\mathcal{O}_\emptyset} = \mathbf{u}$.*

[3] Hence, all its models are (isomorphic to) Herbrand models.

- $(P \wedge \neg P)$. *Clearly, it does not suffice that* $\varphi^{\mathcal{K}} = \mathbf{u}$ *to guarantee that* $\Gamma \cup \{\varphi\}$ *or* $\Gamma \cup \{\neg\varphi\}$ *is satisfiable. Indeed,* $(P \wedge \neg P)^{\mathcal{O}_\emptyset} = \mathbf{u}$*, but this formula is not satisfiable.*

Crucial in the development of computing three-valued approximations is the ability to pose queries to them. For that, we need to extend the notions of certain and possible answers from the two to the three-valued case.

Definition 4.5 *Given a three-valued* Σ*-interpretation* $\mathcal{K}$ *and a query* $\mathcal{Q}[\bar{x}]$ *in* Σ*, denoted by* $\bar{a}$ *a tuple of domain elements.*

- $\bar{a}$ *is a* certain answer *in* $\mathcal{K}$ *for* $\mathcal{Q}[\bar{x}]$*, if* $\mathcal{Q}[\bar{a}]^{\mathcal{K}} = \mathbf{t}$.
- $\bar{a}$ *is a* possible answer *in* $\mathcal{K}$ *for* $\mathcal{Q}[\bar{x}]$*, if* $\mathcal{Q}[\bar{a}]^{\mathcal{K}} \geq \mathbf{u}$.

We denote by $Cert_{\mathcal{K}}(\mathcal{Q}[\bar{x}])$ *the set of certain answers of* $\mathcal{Q}[\bar{x}]$ *in* $\mathcal{K}$*, and by* $Poss_{\mathcal{K}}(\mathcal{Q}[\bar{x}])$ *the corresponding set of possible answers.*

Example 4.7 *Consider again the three-valued* Σ*-structures of Example 4.2. The following corresponds to possible and certain answers for the query* $\mathcal{Q}[x] = P(x) \vee Q(x)$ *with respect to some of the structures.*

$$\begin{array}{ll} Cert_{\mathcal{K}_1}(\mathcal{Q}[\bar{x}]) = \emptyset & Poss_{\mathcal{K}_1}(\mathcal{Q}[\bar{x}]) = \{a\} \\ Cert_{\mathcal{K}_2}(\mathcal{Q}[\bar{x}]) = \{a\} & Poss_{\mathcal{K}_2}(\mathcal{Q}[\bar{x}]) = \{a\} \\ Cert_{\mathcal{K}_3}(\mathcal{Q}[\bar{x}]) = \{a\} & Poss_{\mathcal{K}_3}(\mathcal{Q}[\bar{x}]) = \{a\} \end{array}$$

Given that truth values of sentences can be computed in polynomial time, we have the following tractability result.

Proposition 24 *For each finite three-valued* Σ*-structure* $\mathcal{K}$ *and* Σ*-query* $\mathcal{Q}[\bar{x}]$*, the sets* $Cert_{\mathcal{K}}(\mathcal{Q}[\bar{x}])$ *and* $Poss_{\mathcal{K}}(\mathcal{Q}[\bar{x}])$ *are polynomially computable in the size of* $\mathcal{K}$*.*

A locally closed database $\mathfrak{D} = (D, \mathcal{L})$ corresponds to a theory $\mathcal{M}(\mathfrak{D})$ which is satisfiable and includes $\mathsf{UNA}(\Sigma_{\mathfrak{D}}) \wedge \mathsf{DCA}(\Sigma_{\mathfrak{D}})$. Our goal will be to construct an approximating structure $\mathcal{K}$ of $\mathcal{M}(\mathfrak{D})$. In this case we shall sometimes say that $\mathcal{K}$ approximates $\mathfrak{D}$. Also, we shall sometimes write $\mathcal{O}_{\mathfrak{D}}$ instead of $\mathcal{O}_{\mathcal{M}(\mathfrak{D})}$ to denote the optimal approximation of $\mathcal{M}(\mathfrak{D})$.

The following claim immediately follows from the relevant definitions.

Proposition 25 *Let* $\mathfrak{D} = (D, \mathcal{L})$ *be a locally closed database and* $\mathcal{K}$ *an approximation of* $\mathfrak{D}$*. Then:*

$$Cert_{\mathcal{K}}(\mathcal{Q}[\bar{x}]) \subseteq Cert_{\mathcal{O}_{\mathfrak{D}}}(\mathcal{Q}[\bar{x}]) \subseteq \mathrm{Cert}_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$$

$$\subseteq$$

$$\mathrm{Poss}_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) \subseteq Poss_{\mathcal{O}_{\mathfrak{D}}}(\mathcal{Q}[\bar{x}]) \subseteq Poss_{\mathcal{K}}(\mathcal{Q}[\bar{x}]).$$

An approximation $\mathcal{K}$ of $\mathfrak{D}$ allows us to compute an under estimate of the certain answers and an over-estimate of the possible answers of a query, in polynomial time. From the results in this section it is clear that a tractable method to compute three-valued approximations produces a tractable sound approximative query answering method. Such a method is provided in the next section.

In the same spirit, an approximation $\mathcal{K}$ of $\mathfrak{D}$ gives us also a tractable, sound but incomplete method to verify CWI for queries. Indeed, by Proposition 13,

Proposition 26 *$\mathfrak{D}$ has CWI on $\mathcal{Q}[\bar{x}]$ if $Cert_{\mathcal{K}}(\mathcal{Q}[\overline{x}]) = Poss_{\mathcal{K}}(\mathcal{Q}[\overline{x}])$ and this can be tested in polynomial time.*

4.3 Constructing Approximations by Fixpoint Computations

Now that we know that querying three-valued structures can be done in polynomial time, the next step is to construct such an approximation for locally complete databases. The query will then be answered by a tractable algorithm using the approximation. As we have already seen, in general there are many approximations for a logic theory. What we are after is the optimal approximation, or at least a 'good' approximation. To do that, we rely on the knowledge stored in the database as tuples and the LCWAs.

Let $\mathfrak{D}$ be a locally closed database with a domain $Dom = \mathcal{C}(\Sigma_{\mathfrak{D}})$. In order to construct a three-valued approximation for $\mathfrak{D}$, we first introduce a fixpoint operator on $\mathfrak{L}^c$, the set of three-valued Herbrand structures with domain Dom.

Definition 4.6 (The operator $\mathcal{A}pp_{\mathfrak{D}}$) *The operator $\mathcal{A}pp_{\mathfrak{D}} : \mathfrak{L}^c \to \mathfrak{L}^c$ maps a three-valued structure $\mathcal{K}$ to $\mathcal{K}' = \mathcal{A}pp_{\mathfrak{D}}(\mathcal{K})$ such that, for every predicate P of $\mathcal{R}(\Sigma)$ and every tuple $\bar{a}$:*

$$P(\bar{a})^{\mathcal{K}'} = \begin{cases} \mathbf{t} & \text{if } P(\bar{a}) \in D, \\ \mathbf{f} & \text{if } P(\bar{a}) \notin D \text{ and there exists } \mathcal{LCWA}(P(\bar{x}), \Psi_P[\bar{x}]) \in \mathcal{L} \\ & \text{such that } \Psi_P[\bar{a}]^{\mathcal{K}} = \mathbf{t}, \\ \mathbf{u} & \text{otherwise.} \end{cases}$$

The idea for constructing an approximation of $\mathfrak{D}$ is to start from the structure with total ignorance (the valuation $\perp$ that assigns $\mathbf{u}$ to every ground atom), and to iterate $\mathcal{A}pp_{\mathfrak{D}}$, thereby gradually extending the definite knowledge using the database and its LCWAs. In the first iteration the operator derives all true facts; afterwards it only converts unknown facts into false facts.

Proposition 27 *$\mathcal{A}pp_{\mathfrak{D}}$ is a $\leqslant_{\mathrm{p}}$-monotone operator on the chain complete poset $\mathfrak{L}^c$, thus it has a $\leqslant_{\mathrm{p}}$-least fixpoint. Moreover, the least fixpoint can be computed in polynomial time in the size of the database.*

Proof: It is straightforward to verify that $\mathcal{App}_{\mathfrak{D}}$ is $\leqslant_{\mathrm{p}}$-monotone. By (an extension of) the well-known Knaster-Tarski theorem (see (Davey and Priestley 2002) for a good discussion of this theorem), it follows that the sequence of three-valued structures, starting with the least precise structure and constructed by iterating $\mathcal{App}_{\mathfrak{D}}$ is monotonically increasing in precision and reaches $\mathcal{App}_{\mathfrak{D}}$'s $\leqslant_{\mathrm{p}}$-least precise fixpoint. Polynomial complexity follows from the fact that per application of the operator, the number of queries to be solved is polynomial in the size of the database and each query can be solved in polynomial time, while the number of iterations is at most polynomial in the size of the database. □

Definition 4.7 (Three-valued interpretation $\mathcal{C}_{\mathfrak{D}}$) *The three-valued interpretation that is the $\leqslant_{\mathrm{p}}$-least fixpoint of $\mathcal{App}_{\mathfrak{D}}$ is denoted by $\mathcal{C}_{\mathfrak{D}}$.*

Example 4.8

1. *Consider $D=\emptyset$ and $\theta_1=\mathcal{LCWA}(P,R)$. In the first iteration, none of the first two rules of the operator apply: the first one is not applicable since the database is empty; the second is not applicable because P's window of expertise R does not evaluate to $\mathbf{t}$. As a consequence, $R^{\mathcal{C}_{\mathfrak{D}}}=\mathbf{u}$ and $P^{\mathcal{C}_{\mathfrak{D}}}=\mathbf{u}$.*

2. *Suppose that $D=\emptyset$, $\theta_1=\mathcal{LCWA}(Q,\mathbf{t})$, and $\theta_2=\mathcal{LCWA}(P,\neg Q)$. In the first iteration, the windows of expertise of Q is true and D is empty, hence the second rule applies and we derive $Q^{\mathcal{C}_{\mathfrak{D}}}=\mathbf{f}$. In the second iteration, the window of expertise of P holds, and the second rule yields $P^{\mathcal{C}_{\mathfrak{D}}}=\mathbf{f}$, and a fixpoint is reached.*

3. *Consider again the database instance of Example 1.5 and the following LCWA expression from Example 3.3:*

$$\mathcal{LCWA}(Dept(x,y),y=CS).$$

In the first iteration, we obtain that $Dept(a,b)^{\mathcal{C}_{\mathfrak{D}}}=\mathbf{t}$ for all the tuples (a,b) in $Dept^D$.

In the second iteration, we can derive that $Dept(a,CS)^{\mathcal{C}_{\mathfrak{D}}}=\mathbf{f}$ for all $a\notin\{BD,TD\}$. For the remaining tuples (a,b) not covered by the previous two cases, we have $Dept(a,b)^{\mathcal{C}_{\mathfrak{D}}}=\mathbf{u}$.

4. *Suppose we now change the previous LCWA for the following:*

$$\mathcal{LCWA}(Tel(x,y),Dept(x,CS)),$$

while the database instance remains as before. The interesting truth value is for the Tel relation. In the first iteration, the first rule applies and we obtain $Tel(x,y)^{\mathcal{C}_{\mathfrak{D}}}=\mathbf{t}$ for all (a,b) in $Tel(x,y)^D$ and that $Dept(a,b)^{\mathcal{C}_{\mathfrak{D}}}=\mathbf{t}$ for all the tuples (a,b) in $Dept^D$.

Following the second rule of the operator, we obtain in the second iteration, $Tel(x,y)^{\mathcal{C}_\mathfrak{D}} = \mathbf{f}$ *for all tuples* (a,b) *such that* $a = BD \wedge b \neq 5985625$ *or* $a = TD \wedge b \neq 584521$*. All the other tuples remain unknown.*

5. *We now consider the interaction between two LCWA expressions:*

$$\theta_1 = \mathcal{LCWA}(Dept(x,y), y = CS).$$
$$\theta_1 = \mathcal{LCWA}(Tel(x,y), \neg Dept(x,CS)).$$

We assume the database instance is as in the previous two examples. As before, in the first iteration we obtain $Tel(x,y)^{\mathcal{C}_\mathfrak{D}} = \mathbf{t}$ *for all* (a,b) *in* $Tel(x,y)^D$ *and that* $Dept(a,b)^{\mathcal{C}_\mathfrak{D}} = \mathbf{t}$ *for all the tuples* (a,b) *in* $Dept^D$*.*

In the second iteration, we can derive once again that $Dept(a,CS)^{\mathcal{C}_\mathfrak{D}} = \mathbf{f}$ *for all* $a \notin \{BD, TD\}$*. Then, the second rule applies for the windows of expertise* $\neg Dept(x,CS)$*, and so for all* $a \notin \{BD, TD\}$ *that are not tuples in* Tel^D*, it is the case that* $Tel^D = \mathbf{f}$*. For instance* $Tel(Lien\ Desmet, 0923323)^{\mathcal{C}_\mathfrak{D}} = \mathbf{f}$*.*

The following theorem shows that $\mathcal{C}_\mathfrak{D}$ is a sound approximation of $\mathfrak{D}$.

Theorem 7 (Soundness) $\mathcal{C}_\mathfrak{D}$ *approximates* $\mathfrak{D}$*.*

Proof: Let M be a (Herbrand) model of $\mathfrak{D}$. It is easy so see that $\mathcal{C}_\mathfrak{D} \leqslant_\mathrm{p} M$. Indeed, any true atom in $\mathcal{C}_\mathfrak{D}$ belongs to the database, and hence is true in M; for any false atom $P(\bar{a})$ in $\mathcal{C}_\mathfrak{D}$, $P(\bar{a}) \notin D$ and $\Psi_P[\bar{a}]$ is true in $\mathcal{C}_\mathfrak{D}$. Since M satisfies the local closed-world assumption of P, $P(\bar{a})$ is false in M. Hence $\mathcal{C}_\mathfrak{D} \leqslant_\mathrm{p} M$. Since this holds for every Herbrand model of $\mathfrak{D}$, we have that $\mathcal{C}_\mathfrak{D}$ approximates $\mathfrak{D}$. □

As a corollary to Proposition 25, we obtain a soundness theorem of the approximate method.

Corollary 3 *For any locally closed database* $\mathfrak{D} = (D, \mathcal{L})$*, it holds that*

$$Cert_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}[\overline{x}]) \subseteq \mathrm{Cert}_\mathfrak{D}(\mathcal{Q}[\overline{x}]) \subseteq \mathrm{Poss}_\mathfrak{D}(\mathcal{Q}[\overline{x}]) \subseteq Poss_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}[\overline{x}]).$$

The previous results give us a tractable method for computing possible and certain answers to queries by first computing $\mathcal{C}_\mathfrak{D}$ and then evaluating queries against it. The latter can be implemented using standard techniques from database systems, by rewriting queries and solving them in a database *simulating* $\mathcal{C}_\mathfrak{D}$, as explained in Proposition 21. We describe the process for obtaining certain answers in Algorithm 4.1. Since the transformed query has the same size as the original query, there is no loss of efficiency compared to standard (two-valued) database querying. By application of Proposition 26, we obtain also a tractable approximate method for checking CWI.

Tractability, of course, has a price. As the following example shows, by applying this method, in certain cases, we lose accuracy.

Algorithm 4.1 Computing Certain Answers from $\mathcal{C}_{\mathfrak{D}}$.

1: **Input:** A locally complete database $\mathfrak{D} = (D, \mathcal{L})$ and a database query $\mathcal{Q}[\bar{x}]$ over Σ.
2: Compute iteratively three-valued structure $\mathcal{C}_{\mathfrak{D}}$ as described by $\mathcal{A}pp_{\mathfrak{D}}$.
3: Compute simulation I of $\mathcal{C}_{\mathfrak{D}}$:
4: Define vocabulary Σ'
5: $\mathcal{C}(\Sigma) = \mathcal{C}'(\Sigma)$
6: **for all** $P \in \mathcal{R}(\Sigma)$ **do**
7: Add $P^c \in \mathcal{R}(\Sigma')$
8: Add $P^{c\neg} \in \mathcal{R}(\Sigma')$
9: **end for**
10: $dom(I) = dom(\mathcal{C}_{\mathfrak{D}})$
11: $(P^c)^I = \{\bar{d} \mid P(\bar{d})^{\mathcal{C}_{\mathfrak{D}}} = \mathbf{t}\}$
12: $(P^{c\neg})^I = \{\bar{d} \mid P(\bar{d})^{\mathcal{C}_{\mathfrak{D}}} = \mathbf{f}\}$
13: Substitute P^c for positive occurrences of atoms P in $\mathcal{Q}[\bar{x}]$.
14: Substitute $\neg P^{c\neg}$ for negative occurrences of atoms P.
15: Evaluate $\mathcal{Q}^c[\bar{x}]$ with respect to I.
16: **Output:** set of certain answers.

Example 4.9 *Let us compare $\mathcal{C}_{\mathfrak{D}}$ and $\mathcal{O}_{\mathfrak{D}}$, where:*

$$\mathfrak{D} = (\emptyset, \{\theta\}).$$
$$\theta = \mathcal{LCWA}(Q, P \vee \neg P).$$

This database has models in which P is true and others in which P is false, but, because of its LCWA, Q is false in all of them. Thus $P^{\mathcal{O}_{\mathfrak{D}}} = \mathbf{u}$ and $Q^{\mathcal{O}_{\mathfrak{D}}} = \mathbf{f}$. However, since $P \vee \neg P$ evaluates to $\mathbf{u}$ in each structure $\mathcal{K}$ for which $P^{\mathcal{K}} = \mathbf{u}$, we have that $Q^{\mathcal{C}_{\mathfrak{D}}} = \mathbf{u}$. The answer for the query $\neg Q$ in $\mathcal{C}_{\mathfrak{D}}$ is therefore $\mathbf{u}$, while it is $\mathbf{t}$ when posed with respect to $\mathfrak{D}$ or $\mathcal{O}_{\mathfrak{D}}$.

In Section 4.4, the loss of accuracy and the cases when it can be prevented will be studied in more detail. But first we address an important drawback of this approach, namely the need of a concrete representation of $\mathcal{C}_{\mathfrak{D}}$ and its re-computation each time the database changes.

4.3.1 Simulating Approximate Query Answering by Fixpoint Queries

The re-computation of $\mathcal{C}_{\mathfrak{D}}$ after each update of D in the locally closed database $\mathfrak{D} = (D, \mathcal{L})$ can be (partially) avoided by using fixpoint formulas (Libkin 1995;

Ebbinghaus and Flum 2005) or Datalog programs (Ullman 1988) that symbolically describe the construction of $\mathcal{C}_\mathfrak{D}$. Using these expressions, certain or possible answers to queries can be computed by transforming the query into a fixpoint query or in a query with respect to some Datalog program. The resulting query can be run directly against D, without explicitly constructing $\mathcal{C}_\mathfrak{D}$. The query answering process inherits the computational complexity of fixpoints logic and, as we will show, query answering remains tractable for both certain and possible answer.

We start this section with providing the basic concepts of fixpoints logic. As it turns out that the fragments of fixpoint logic we use here are equivalent to fragments of Datalog (Ullman 1988), logic programming under the well-founded semantics (Van Gelder et al. 1991) and ID-Logic (Denecker et al. 2001), the formalism we present is flexible enough to be implemented in a number of systems that support these logics. In fact, in the next chapter we present a prototype implementation on the model expansion/generator system IDP. The selection of fixpoint logics over the aforementioned formalisms is motivated by the extended use and the increasing importance of these logics in database circles.

Next in this section we will present an algorithm to produce fixpoint queries based on a locally complete database and a query for certain or possible answers. It will not come as a surprise that the construction of the fixpoint queries will resemble the fixpoint operator $\mathcal{A}pp_\mathfrak{D}$. It will turn out that for certain types of queries and databases, the resulting expression will be a standard first-order logic query. In such cases, the queries can be evaluated using standard SQL engines.

The advantages of this approach over the evaluation of queries over three-valued structures can be summarized as follows:

- No need to recompute the three-valued structure $\mathcal{C}_\mathfrak{D}$ each time the original database D is updated.
- Since it will not be needed to explicitly store negative facts, the storage space will be limited to that of the original database.
- Standard engines for fixpoint query answering can be used to answer queries.

4.3.2 The logic LFP^{simult}

We will construct a query for certain or possible answers by a fixpoint expression in the logic LFP^{simult}of simultaneous fixpoints. For reasons of convenience, we use a slightly different notation for such expressions.

Definition 4.8 *Let Σ be a first-order vocabulary. A least fixpoint expression is a formula in the form of $[lfp_{R_i,\Delta}](\bar{t})$, where $\bar{t}$ is a tuple of terms of length*

n_i, R_i ($1 \leq i \leq n$) is a predicate variable of arity n_i and Δ is a collection of rules of the form

$$\{\forall \bar{x}_j \; R_j(\bar{x}_j) \leftarrow \varphi_j[\bar{x}_j] \mid 1 \leq j \leq n\},$$

where each φ_j is a formula over $\Sigma \cup \{R_1, \ldots, R_n\}$, $R_1, \ldots, R_n$ are predicate variables with only positive occurrences in φ_j, and the arity of each R_j is n_j, the length of $\bar{x}_j$.

In the standard notations of LFP^{simult}, the fixpoint expression $[lfp_{R_i,\Delta}](\bar{t})$ is denoted by $[lfp_{R_1(\bar{x}_1),\ldots,R_n(\bar{x}_n)}(\varphi_1[\bar{x}_1], \ldots, \varphi_n[\bar{x}_n]), R_i](\bar{t})$, meaning that $\bar{t}$ belongs to relation R_i where $R_1, \ldots, R_n$ are defined by simultaneous monotone induction by the formulas $\varphi_1, \ldots, \varphi_n$ respectively. Both notations are syntactic variants. The rule notation is more convenient for linking predicates and the formulas defining them and, more importantly, for showing the close relationship to Datalog.

We now extend the two-valued truth assignment to fixpoint expressions:

Definition 4.9 *Let U be a Σ-structure and $[lfp_{R_i,\Delta}](\bar{t})$ a fixpoint expression without free variables (but possibly containing domain elements of U). We define $([lfp_{R_i,\Delta}](\bar{t}))^U = \mathbf{t}$ if and only if $\bar{t}^U \in \mathcal{R}_i$, where $\mathcal{R}_i$ is the i'th argument in the least fixpoint $(\mathcal{R}_1, \ldots, \mathcal{R}_n)$ of the operator Γ_Δ, associated to Δ and U.*

The operator Γ_Δ operates on tuples $(S_1, \ldots, S_n)$ of relations of arity $n_1, \ldots, n_n$, such that $\Gamma(S_1, \ldots, S_n) = (S'_1, \ldots, S'_n)$ for

$$S'_i = \{\bar{d} \mid U[R_1 : S_1, \ldots, R_n : S_n] \models \varphi_i[\bar{d}]\}.$$

Here, $U[R_1 : S_1, \ldots, R_n : S_n]$ denotes the structure extending U by interpreting each R_i by S_i.

Observe that the defined operator is indeed a monotone lattice operator and its least fixpoint is well-defined. It is also worth noting that Δ is an extended Datalog program as defined in (Van Gelder 1993), or a positive definition, as defined in FO[ID] (see (Denecker and Ternovska 2004)). Its semantics, i.e. its least fixpoint, coincides with the well-founded model of Δ. In Datalog, the fixpoint query $[lfp_{R_i,\Delta}](\bar{t})$ would be posed as the query $R_i(\bar{t})$ with respect to the Datalog program Δ.

In the context of U, the definition Δ co-inductively defines the tuple of relations $(\mathcal{R}'_1, \ldots, \mathcal{R}'_n)$, as the greatest fixpoint of Γ_Δ, which is denoted here by $[gfp_{R_i,\Delta}](\bar{t})$. If $[gfp_{R_i,\Delta}](\bar{t})$ contains no free variables, we have $([gfp_{R_i,\Delta}](\bar{t}))^U = \mathbf{t}$ if $\bar{t}^U \in \mathcal{R}'_i$.

Least fixpoints and greatest fixpoints are connected through a duality result. Given the set Δ, define Δ' as the collection of rules

$$\{\forall \bar{x}_j \; R'_j(\bar{x}_j) \leftarrow \varphi'_j[\bar{x}_j] \mid 1 \leq j \leq n\},$$

where $\varphi'_j[\bar{x}_i]$ denotes the formula obtained from $\neg\varphi_j[\bar{x}_j]$ by replacing each occurrence of an atom $R_i(\bar{t})$ by $\neg R'(\bar{t})$. The formulas φ'_j are positive in the R'_k's. It is well known that for $(\mathcal{R}_1, \ldots, \mathcal{R}_n)$ the least fixpoint of Γ_Δ, and $(\mathcal{R}'_1, \ldots, \mathcal{R}'_n)$ the greatest fixpoint of $\Gamma_{\Delta'}$, the relations $\mathcal{R}_i, \mathcal{R}'_i$ are complements. It follows that $[lfp_{R_i,\Delta}](\bar{t})$ and $\neg[gfp_{R'_i,\Delta'}](\bar{t})$ are logically equivalent.

4.3.3 Query Approximation by Expressions in LFP^{simult}.

Suppose now that the database vocabulary Σ contains predicates $P_1, \ldots, P_n$. For each of these predicate symbols, we introduce four new predicate variables P_i^c, P_i^p, $P_i^{c\neg}$ and $P_i^{p\neg}$, of the same arity as P_i, to represent respectively the certain and the possible tuples of P_i, and the tuples that certainly and possibly do not belong to P_i. Along this intuition, it is clear that P_i^p and $P_i^{c\neg}$ (respectively, P_i^c and $P_i^{p\neg}$) represent each others complements.

To each formula Ψ with predicate symbols amongst $P_1, \ldots, P_n$, we associate two formulas, representing the certain instances and the possible instances of the formula when interpreted as a query:

- Ψ^c is the formula obtained by substituting $P_i^c(\bar{t})$ for each positive occurrence of $P_i(\bar{t})$ in Ψ, and substituting $\neg P_i^{c\neg}(\bar{t})$ for each negative occurrence of $P_i(\bar{t})$ in Ψ. As mentioned in Proposition 21, this formula can be used to compute three-valued answers for Ψ.

- Ψ^p is, inversely, the formula obtained by substituting $P_i^p(\bar{t})$ for each positive occurrence of $P_i(\bar{t})$ in Ψ, and substituting $\neg P_i^{p\neg}(\bar{t})$ for each negative occurrence of $P_i(\bar{t})$ in Ψ.

Observe that Ψ^c contains only predicate variables of the form P_i^c and $P_i^{c\neg}$, and Ψ^p contains only $P_i^p(\bar{t})$ and $P_i^{p\neg}(\bar{t})$. Moreover, both Ψ^c and Ψ^p contain only positive occurrences of the new predicate variables.

Definition 4.10 (Rules for certain answers, $\Delta^c_{\mathcal{Q},\mathcal{L}}$) *Let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database over a vocabulary Σ and let $\mathcal{Q}[\bar{x}]$ be a query. In addition to the predicate variables P_i^c and $P_i^{c\neg}$, we introduce two new variables Q^c and $Q^{c\neg}$ whose arity is the number of free variables of $\mathcal{Q}[\bar{x}]$, and define the set of* certain rules $\Delta^c_{\mathcal{Q},\mathcal{L}}$*, as the following collection*[4]*:*

- *two rules for the query:*

$$Q^c(\bar{x}) \leftarrow \mathcal{Q}[\bar{x}]^c$$
$$Q^{c\neg}(\bar{x}) \leftarrow (\neg\mathcal{Q}[\bar{x}])^c$$

[4]Recall Ψ_{P_i} is the window of expertise of P_i.

- *two rules for each database predicate P_i:*

$$P_i^c(\bar{x}_i) \leftarrow P_i(\bar{x}_i)$$
$$P_i^{c\neg}(\bar{x}_i) \leftarrow \neg P_i(\bar{x}_i) \wedge (\Psi_{P_i}[\bar{x}_i])^c$$

Define the certain answer query for $\mathcal{Q}[\bar{x}]$ as $[lfp_{Q^c, \Delta^c_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ and the possible answer query for $\mathcal{Q}[\bar{x}]$ as $\neg[lfp_{Q^{c\neg}, \Delta^c_{\mathcal{Q},\mathcal{L}}}](\bar{x})$.

Intuitively, the rules in Definition 4.10 define Q^c as the collection of certain instances of $\mathcal{Q}[\bar{x}]$ and $Q^{c\neg}$ as the collection of certain instances of $\neg\mathcal{Q}[\bar{x}]$. The rule for P_i^c captures the idea that elements of P_i^c, i.e. certain elements of P_i, are those in P_i's table, and the rule for $P_i^{c\neg}$ expresses that tuples for which P_i is certainly false, are those that do not occur in the database and for which the window of expertise is certainly satisfied. Note that the two fixpoint queries in Definition 4.10 are legal LFP^{simult}queries. In particular, all the new symbols occur positively in the formulas of the rules. We illustrate the construction of $[lfp_{Q^c, \Delta^c_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ by a simple example.

Example 4.10 *Consider the database instance of Example 1.5 and the LCWA expression*

$$\mathcal{LCWA}(Tel(x,y), Dept(x, CS))$$

We are interested in the certain answer for the query $\mathcal{Q} = \neg Tel(BD, 75683764)$. This is, we want to know for certain that Bart Delvaux has no telephone number 75683764. *We start constructing $\Delta^c_{\mathcal{Q},\mathcal{L}}$ with the rules for the query:*

$$Q^c(BD, 75683764) \leftarrow Tel^{c\neg}(BD, 75683764)$$
$$Q^{c\neg}(BD, 75683764) \leftarrow Tel^c(BD, 75683764)$$

And the rules for each database predicate:

$$Dept^c(x,y) \leftarrow Dept(x,y)$$
$$Dept^{c\neg}(x,y) \leftarrow \neg Dept(x,y) \wedge \mathbf{f}$$
$$Tel^c(x,y) \leftarrow Tel(x,y)$$
$$Tel^{c\neg}(x,y) \leftarrow \neg Tel(x,y) \wedge Dept^c(x, CS)$$

The evaluation of this expression can be computed by unfolding its rule body. This yields the answer 'yes' on the database of Example 1.5.

Similarly, we can define a set of possible rules.

Definition 4.11 (Rules for possible answers, $\Delta^p_{\mathcal{Q},\mathcal{L}}$) *Under the same assumptions as in Definition 4.10, we introduce variables Q^p and $Q^{p\neg}$ and define the set of possible rules $\Delta^p_{\mathcal{Q},\mathcal{L}}$ as the following collection:*

- *two rules for the query:*

$$Q^p(\bar{x}) \leftarrow \mathcal{Q}[\bar{x}]^p$$
$$Q^{p\neg}(\bar{x}) \leftarrow (\neg\mathcal{Q}[\bar{x}])^p$$

- *two rules for each database predicate P_i:*

$$P_i^p(\bar{x}_i) \leftarrow P_i(\bar{x}_i) \vee (\neg\Psi_{P_i}[\bar{x}_i])^p$$
$$P_i^{p\neg}(\bar{x}_i) \leftarrow \neg P_i(\bar{x}_i)$$

It is easy to verify that $\Delta^p_{\mathcal{Q},\mathcal{L}}$ is equivalent to the dual of $\Delta^c_{\mathcal{Q},\mathcal{L}}$. It follows that $\neg[lfp_{Q^{c\neg},\Delta^c_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ and $[gfp_{Q^p,\Delta^p_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ are equivalent, and so are $[lfp_{Q^c,\Delta^c_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ and $\neg[gfp_{Q^{p\neg},\Delta^p_{\mathcal{Q},\mathcal{L}}}](\bar{x})$.

We now illustrate the main ideas of this section with an elaborated example:

Example 4.11 *Consider again the database instance of Example 1.6. For convenience we abbreviate the $CarOwners$ predicate as $CarO$ and its tuples as $(PS, M, Q1)$, $(JS, V, B1)$ and $(MC, B, B2)$, and the Location predicate as Loc and its tuples as (PS, Qs), (JS, Bx) and (MC, Bx). Assume that we have two local closed-world assumptions, namely:*

$\mathcal{LCWA}(CarO(n, m, id), Loc(n, Bx))$*, expressing that the database contains all car owners living in the Bronx, and*

$\mathcal{LCWA}(Loc(n, l), (l = Bx \wedge \exists m, idCarO(n, m, id)))$*, expressing that the database knows all people living in the Bronx that own a car.*

Our interest is in what is certain and possible for the $CarO$ relation. There are two equivalent ways to express this. The first one uses the set of certain rules (in the rules below, implicit universal quantification is assumed):

$$\Delta^c_{\mathcal{Q},\mathcal{L}} = \left\{ \begin{array}{l} Q^c(n, m, id) \leftarrow CarO^c(n, m, id). \\ Q^{c\neg}(n, m, id) \leftarrow CarO^{c\neg}(n, m, id). \\ CarO^c(n, m, id) \leftarrow CarO(n, m, id). \\ CarO^{c\neg}(n, m, id) \leftarrow \neg CarO(n, m, id) \wedge Loc^c(n, Bx). \\ Loc^c(n, l) \leftarrow Loc(n, l). \\ Loc^{c\neg}(n, l) \leftarrow \neg Loc(n, l) \wedge l = Bx \wedge \exists m, idCarO^c(n, m, id). \end{array} \right\}$$

Interestingly, this set of rules is non-recursive. It follows that its least and greatest fixpoint coincide. Also, the query $[lfp_{Q^c,\Delta^c_{\mathcal{Q},\mathcal{L}}}](n, m, id)$ returns all elements of Q^c, and this relation can be computed by simply unfolding its rule body. By two unfolding steps, $Q^c(n, m, id)$ is rewritten to the database query $CarO(n, m, id)$. Thus, the certain answer is, not surprisingly, the set of database tuples of $CarO$:

$$(PS, M, Q1), \qquad (JS, V, B1), \qquad (MC, B, B2).$$

Similarly, to obtain the certainly false tuples for the query, we unfold the query $[lfp_{Q^{c\neg},\Delta^{c}_{\mathcal{Q},\mathcal{L}}}](n,m,id)$ *yielding the database query* $\neg CarO(n,m,id) \wedge Loc(n,Bx)$*. Its answer is the following set:*

$$\big\{(JS,m,id) \mid (m,id) \neq (V,B1)\big\} \bigcup \big\{(MC,m,id) \mid (m,id) \neq (B,B2)\big\},$$

which, for the two inhabitants of the Bronx, contains all tuples not in the database. Finally, the set of possible but uncertain answers is the complement of the union of the certain answers and the certainly false answers. This set is specified by the database query

$$\neg CarO(n,m,id) \wedge (CarO(n,m,id) \vee \neg Loc(n,Bx)),$$

which can be simplified to $\neg CarO(n,m,id) \wedge \neg Loc(n,Bx)$*. Its answer is the following set:*

$$\big\{(PS,m,id) \mid (m,id) \neq (M,Q1)\big\}.$$

Indeed, as Peter Steward does not live in the Bronx, we do not know whether he owns other cars. Moreover, the database does not contain the integrity constraints that a license plate is used for only one car, hence he can own any of the cars in the active domain and the license plate can also be any of those in the active domain.

Alternatively, to compute the above sets we could also use the set of possible rules:

$$\Delta^{p}_{\mathcal{Q},\mathcal{L}} = \left\{ \begin{array}{l} Q^{p}(n,m,id) \leftarrow CarO^{p}(n,m,id). \\ Q^{p\neg}(n,m,id) \leftarrow CarO^{p\neg}(n,m,id). \\ CarO^{p}(n,m,id) \leftarrow CarO(n,m,id) \vee Loc^{p\neg}(n,Bx). \\ CarO^{p\neg}(n,m,id) \leftarrow \neg CarO(n,m,id). \\ Loc^{p}(n,l) \leftarrow Loc(n,l) \vee l \neq Bx \vee \forall m, id CarO^{p\neg}(n,m,id). \\ Loc^{p\neg}(n,l) \leftarrow \neg Loc(n,l). \end{array} \right\}$$

Again, this is a non-recursive set of rules.

The expression $[gfp_{Q^{p},\Delta^{p}_{\mathcal{Q},\mathcal{L}}}](n,m,id)$ *can be unfolded to the database query* $CarO(n,m,id) \vee \neg Loc(n,Bx)$*, which is indeed the negation of database query solving* $[lfp_{Q^{c\neg},\Delta^{c}_{\mathcal{Q},\mathcal{L}}}](n,m,id)$*. Similarly,* $[gfp_{Q^{p\neg},\Delta^{p}_{\mathcal{Q},\mathcal{L}}}](n,m,id)$ *rewrites to* $\neg CarO(n,m,id)$*, the negation of the database query representing the certain answers of* $CarO(n,m,id)$*.*

We have shown how to compute certain and possible answers to queries by means of fixpoint expressions. Why now proceed to prove formally that the answers to such queries are indeed answers to the original query with respect to the three-valued interpretation.

Theorem 8 *Given a locally closed database* $\mathfrak{D} = (D,\mathcal{L})$ *over* Σ*, an arbitrary query* $\mathcal{Q}[\bar{x}]$ *and the relations* $(\mathcal{R}^{c}_{\mathcal{Q}}, \mathcal{R}^{c\neg}_{\mathcal{Q}}, \mathcal{R}^{c}_{1}, \mathcal{R}^{\neg c}_{1}, \ldots, \mathcal{R}^{c}_{n}, \mathcal{R}^{\neg c}_{n})$ *defined by* $\Delta^{c}_{\mathcal{Q},\mathcal{L}}$ *in* D*. The following equalities hold:*

- $\mathcal{R}_i^c = \{\bar{d} \mid P_i(\bar{d})^{\mathcal{C}_\mathfrak{D}} = \mathbf{t}\}$, *for each* $1 \leq i \leq n$,
- $\mathcal{R}_i^{c\neg} = \{\bar{d} \mid P_i(\bar{d})^{\mathcal{C}_\mathfrak{D}} = \mathbf{f}\}$, *for each* $1 \leq i \leq n$,
- $\mathcal{R}_\mathcal{Q}^c = \{\bar{d} \mid \mathcal{Q}[\bar{d}]^{\mathcal{C}_\mathfrak{D}} = \mathbf{t}\}$, *and*
- $\mathcal{R}_\mathcal{Q}^{c\neg} = \{\bar{d} \mid \mathcal{Q}[\bar{d}]^{\mathcal{C}_\mathfrak{D}} = \mathbf{f}\}$.

Proof: The proof is by comparison of the fixpoint computations of the operator $\mathcal{A}pp_\mathfrak{D}$ that constructs $\mathcal{C}_\mathfrak{D}$ and the operator $\Gamma_{\Delta^c_{\mathcal{Q},\mathcal{L}}}$. The first one constructs a sequence of three-valued Σ-structures $\mathcal{K}_\alpha$, the second one a sequence of tuples of relations, or equivalently, a sequence of Σ'-structures I_α where Σ' contains the new predicates P_i^c and $P_i^{c\neg}$. Note that the predicates $Q^c, Q^{c\neg}$ have no recursive occurrences in the bodies of $\Delta^c_{\mathcal{Q},\mathcal{L}}$, and hence, do not influence the construction of the P_i^c's and the $P_i^{c\neg}$'s. Hence, for the moment, we ignore them.

We will show that each I_α *simulates* $\mathcal{K}_\alpha$ in the sense of Definition 4.1. It is obvious that I_0 simulates $\mathcal{K}_0$, so it suffices to show that applying $\mathcal{C}_\mathfrak{D}$ and $\Gamma_{\Delta^c_{\mathcal{Q},\mathcal{L}}}$ preserves this property. Assume that I simulates $\mathcal{K}$ and $I' = \Gamma_{\Delta^c_{\mathcal{Q},\mathcal{L}}}(I), \mathcal{K}' = \mathcal{C}_\mathfrak{D}(\mathcal{K})$. We need to show that I' simulates $\mathcal{K}'$.

- For predicate P_i, $P_i(\bar{a})^{\mathcal{K}'} = \mathbf{t}$ if and only if $P_i(\bar{a}) \in D$, iff $P_i^c(\bar{a})^{I'} = \mathbf{t}$.
- Since I simulates $\mathcal{K}$, $\Psi_{P_i}[\bar{a}]^\mathcal{K} = \mathbf{t}$ if and only if $(\Psi_{P_i}[\bar{a}]^c)^I = \mathbf{t}$. Therefore, $P_i(\bar{a})^{\mathcal{K}'} = \mathbf{f}$ if and only if $P_i(\bar{a}) \notin D$ and $\Psi_{P_i}[\bar{a}]^\mathcal{K} = \mathbf{t}$, if and only if $P_i(\bar{a}) \notin D$ and $(\Psi_{P_i}[\bar{a}]^c)^I = \mathbf{t}$, if and only if $P_i^{c\neg}(\bar{a})^{I'} = \mathbf{t}$.

It follows that I' indeed simulates $\mathcal{K}'$. The consequence of this is that when $\mathcal{C}_\mathfrak{D}$ reaches a fixpoint, say at α (i.e. $\mathcal{K}_\alpha = \mathcal{C}_\mathfrak{D}$), then $\Gamma_{\Delta^c_{\mathcal{Q},\mathcal{L}}}$ reaches a fixpoint on all predicates P_i^c and $P_i^{c\neg}$ at α, and the fixpoint of $\Gamma_{\Delta^c_{\mathcal{Q},\mathcal{L}}}$ simulates $\mathcal{C}_\mathfrak{D}$. After one more iteration, $\Gamma_{\Delta^c_{\mathcal{Q},\mathcal{L}}}$ reaches a fixpoint also on the predicates Q^c and $Q^{c\neg}$. It holds that $\bar{d} \in \mathcal{R}_\mathcal{Q}^c$ if and only if $(\mathcal{Q}[\bar{d}]^c)^{I_\alpha} = \mathbf{t}$ if and only if $(\mathcal{Q}[\bar{d}])^{\mathcal{K}_\alpha} = \mathbf{t}$ (by Proposition 21). Likewise, $\bar{d} \in \mathcal{R}_\mathcal{Q}^{c\neg}$ if and only if $(\mathcal{Q}[\bar{d}])^{\mathcal{K}_\alpha} = \mathbf{f}$. This finishes the proof.

□

The proof of the above theorem shows that a naive bottom-up computation of the fixpoint query $[lfp_{Q^c, \Delta^c_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ boils down to be the same as first computing $\mathcal{C}_\mathfrak{D}$ and then evaluating $\mathcal{Q}[\bar{x}]$. In fact, the worst case complexity of this method is not better than that of the naive method. Of course, in practice, it is often not necessary to compute all relations, but only those that are relevant for the query. Moreover, in many cases, goal directed methods such as unfolding (see next section), magic sets (Bancilhon et al. 1986) or tabling (Swift 1999), will construct only fractions of those relations. This was illustrated in Example 4.11. On the other hand, if the database is relatively stable, in the sense that many queries are to be solved with respect to the same database,

it might be a good idea to compute $\mathcal{C}_{\mathfrak{D}}$ after all, and pose all queries directly against it. The gain of computing $\mathcal{C}_{\mathfrak{D}}$ grows with:

1. the number of queries that can be answered with it (which is dictated by the ratio of queries posed to the database over the number of updates performed to it), and
2. the size of the windows of expertise and the complexity of the dependency relations, for these factors determine to a great extent the complexity of the fixpoint queries.

Note that, after an update of D, we must update (the simulation of) $\mathcal{C}_{\mathfrak{D}}$, but this can be done by applying incremental methods for *view updating* on the rule set $\Delta^c_{\mathcal{L}}$.

A special case arises when $\Delta^c_{\mathcal{Q},\mathcal{L}}$ is non-recursive. This is studied in the next section.

4.3.4 Hierarchically Closed Databases

Given a locally closed database $\mathfrak{D} = (D, \mathcal{L})$ over a vocabulary Σ.

Definition 4.12 (Dependency graph) *The* LCWA dependency graph *of $\mathcal{L}$ is the directed graph on $\mathcal{R}(\Sigma)$, containing a directed edge from predicate Q to P if and only if there exists $\mathcal{LCWA}(P(\overline{x}), \Psi[\overline{x}]) \in \mathcal{L}$ such that Q occurs negatively in Ψ. A* hierarchically closed *database $\mathfrak{D}$ is one in which the LCWA dependency graph is cycle-free.*

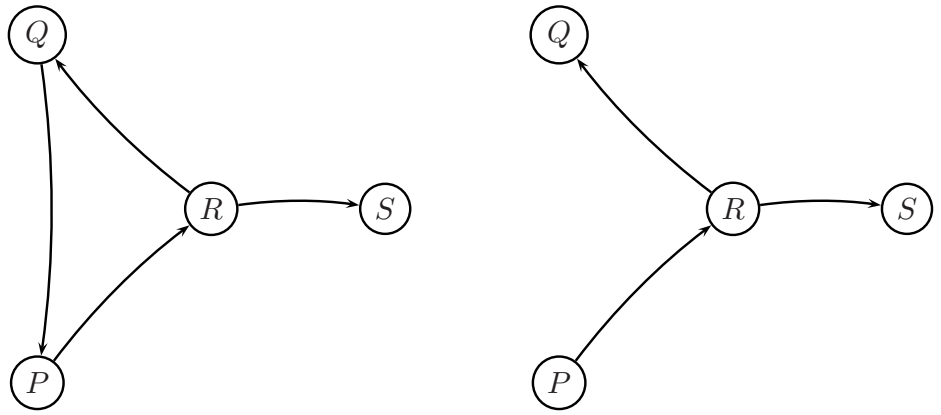
Example 4.12 *Consider the following set of LCWA based on vocabulary Σ, where $\mathcal{R}(\Sigma) = \{P, Q, R, S\}$:*

$$\theta_1 = \mathcal{LCWA}(S(x), \neg R(x)) \qquad \theta_3 = \mathcal{LCWA}(P(x), \neg Q(x))$$
$$\theta_2 = \mathcal{LCWA}(Q(x), P(x) \vee \neg R(x)) \qquad \theta_4 = \mathcal{LCWA}(R(x), \neg P(x))$$

Figure 4.3 (a) corresponds to the LCWA dependency graph of $\mathcal{L}$. In this case $\mathcal{L}$ contains a loop involving predicates P, Q, R, and hence any database $\mathfrak{D}$ with LCWAs $\mathcal{L}$ is not hierarchically closed. To break the loop we may, for instance, change θ_3 by $\mathcal{LCWA}(P(x), Q(x))$. The resulting dependency graph for the updated LCWA is depicted in figure (b).

It is easy to check that a locally closed database $\mathfrak{D}$ is hierarchically closed if and only if for arbitrary query $\mathcal{Q}[\bar{x}]$ both the set of certain rules $\Delta^c_{\mathcal{Q},\mathcal{L}}$ and the set of possible rules $\Delta^p_{\mathcal{Q},\mathcal{L}}$ are non-recursive.

An important observation is that positive occurrences of predicates in windows of expertise do not lead to cycles in the LCWA-dependency graph or to



(a) **A cyclic set of LCWAs** **(b)** **A cyclic-free set of LCWAs**

Figure 4.3: LCWA dependency graph of $\mathcal{L}$.

recursion in the query definitions $\Delta^c_{\mathcal{Q},\mathcal{L}}$ and $\Delta^p_{\mathcal{Q},\mathcal{L}}$. The relevance of this is that, in practical applications, recursive $\Delta^p_{\mathcal{Q},\mathcal{L}}$ will likely be quite rare. For example, if we reconsider the examples of the computer science telephone database or the database of the traffic tax administration, we see a pattern that such databases are specialized in a specific topic: members of the department, or car owners in the county. The predicates representing these topics yield the windows of expertise of other database predicates. These predicates occur positively in the windows of expertise, and moreover, windows of expertise will in general be simple formulas consisting of a conjunction of positive literals or base predicates. If this pattern is general, as we expect, then locally closed databases with cyclic LCWA-dependency graphs will be rare indeed.

The observation is important for two major reasons:

1. One is that, as shown in the next section about accuracy of approximations, for hierarchically closed databases, the constructed $\mathcal{C}_{\mathfrak{D}}$ will often coincide with the optimal approximation $\mathcal{O}_{\mathfrak{D}}$. This will prove beneficial for obtaining optimal answers to queries when certain additional conditions in queries are fulfilled.

2. In a hierarchically closed database $\Delta^c_{\mathcal{Q},\mathcal{L}}$ and $\Delta^p_{\mathcal{Q},\mathcal{L}}$ have a unique fixpoint and hence, least and greatest fixpoints coincide. In particular, both the certain answer query $[lfp_{Q^c,\Delta^c_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ and the (alternative) possible answer query $[gfp_{Q^p,\Delta^p_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ can be transformed in equivalent first-order queries by unfolding the definitions of all predicates. The main benefits of this is that queries for possible and/or certain answers can be performed by standard databases engines supporting SQL or relational algebra.

Algorithm 4.2 below rewrites a query $\mathcal{Q}[\bar{x}]$ into queries for certain or possible answers that can be posed directly against D.

Algorithm 4.2 The query rewriting function R^c (resp., R^p).

1: **Input:** a set $\mathcal{L}$ of LCWAs of a hierarchically closed database and a query $\mathcal{Q}[\bar{x}]$ over Σ.
2: Define $R^c(\mathcal{Q}[\bar{x}])$ and $R^p(\mathcal{Q}[\bar{x}])$ by simultaneous induction on the structure of $\mathcal{Q}[\bar{x}]$ as follows:
3: $R^c(P(\bar{t})) := R^p(P(\bar{t})) := P(\bar{t})$, if P is $\mathbf{t}, \mathbf{f}$ or $=$.
4: $R^c(P(\bar{t})) := P(\bar{t})$.
5: $R^p(P(\bar{t})) := (P(\bar{t}) \vee \neg R^c(\Psi_P(\bar{t}))$.
6: $R^c(\neg\varphi) := \neg R^p(\varphi)$.
7: $R^p(\neg\varphi) := \neg R^c(\varphi)$.
8: R^c and R^p distribute over $\wedge, \vee, \exists, \forall$.
9: **Output:** a query $R^c(\mathcal{Q}[\bar{x}])$ (resp., $R^p(\mathcal{Q}[\bar{x}])$) over Σ.

In the following, we denote by $Cert_A(\mathcal{Q}(\bar{x}))$ (resp. $Poss_A(\mathcal{Q}(\bar{x}))$) the answers obtained by evaluating $R^c(\mathcal{Q}(\bar{x}))$ (resp. $R^p(\mathcal{Q}(\bar{x}))$) in a database instance D.

Example 4.13 *Let $\mathcal{R}(\Sigma) = \{P, Q\}$ and $\mathfrak{D} = (D, \mathcal{L})$, where*

$$D = \{P(a), Q(c)\},\ \mathcal{L} = \{\mathcal{LCWA}(P(x), Q(x)),\ \mathcal{LCWA}(Q(x), x{=}c)\}.$$

$\mathfrak{D}$ is clearly hierarchically closed. Now, consider the query $\mathcal{Q}[x] = P(x)$.

- *For certain query answers, we evaluate $R^c(P(x)) = P(x)$ with respect to D.*
- *For possible answers we start with $R^p(P(x)) = (P(x) \vee \neg R^c(Q(x))$. The unfolding of $R^c(Q(x))$ leads to $R^p(P(x)) = P(x) \vee \neg Q(x)$. Evaluating this expression with respect to D we have that $Poss_A(P(x)) = dom(D) \setminus \{c\}$.*

These sets are exactly the certain and possible answers of the query $P(x)$ in $\mathfrak{D}$.

Example 4.14 *The procedures R^c and R^p correspond to the unfolding process as mentioned in Example 4.11. Not surprisingly, if we apply them to the query $CarO(n, m, id)$ of Example 4.11, we get*

$$R^c(CarO(n, m, id)) = CarO(n, m, id),$$
$$R^p(CarO(n, m, id)) = CarO(n, m, id) \vee \neg Loc(n, Bx),$$

which are exactly the queries obtained in Example 4.11.

Theorem 9 (Soundness of R^c and R^p) *The functions R^c and R^p are well-defined and for each query $\mathcal{Q}[\bar{x}]$, $\{\bar{a} \mid D \models R^c(Q[\bar{a}])\} = Cert_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}[\bar{x}])$ and $\{\bar{a} \mid D \models R^p(Q[\bar{a}])\} = Poss_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}[\bar{x}])$.*

Proof: It is not difficult to see that the operation $R^c(\mathcal{Q}[\bar{x}])$ basically simulates the process of, first, the construction of $\Delta^c_{\mathcal{Q},\mathcal{L}}$ from $\mathcal{L}$ and $\mathcal{Q}[\bar{x}]$, and second, the subsequent unfolding of the atom $Q^c(\bar{x})$. In the construction of $\Delta^c_{\mathcal{Q},\mathcal{L}}$, all positive occurrences of database predicates P_i in $\mathcal{Q}[\bar{x}]$ are first replaced by P_i^c; in the unfolding of $Q^c(\bar{x})$, these occurrences are replaced by P_i. Correspondingly, R^c preserves positive occurrences of database predicates. In the construction of $\Delta^c_{\mathcal{Q},\mathcal{L}}$, all negative occurrences of database predicates $P_i(\bar{t})$ in $\mathcal{Q}[\bar{x}]$ are replaced by $\neg P_i^{c\neg}(\bar{t})$, and in the unfolding phase, these negative literals are replaced by $P_i(\bar{t}) \vee \neg\Psi^c_{P_i}[\bar{t}]$. As for the rewriting procedure, R^c calls R^p for negative occurrences of database predicates, and this procedure replaces such negative occurrences by the corresponding formula $P_i(\bar{t}) \vee \neg R^c(\Psi_{P_i}[\bar{t}])$. By iterating the above argument, we see that $R^c(\Psi_{P_i}[\bar{t}])$ is the result of unfolding $\neg P_i^{c\neg}(\bar{t})$.

Similarly, we have that $R^p(\mathcal{Q}[\bar{x}])$ simulates the process of, first, the construction of $\Delta^p_{\mathcal{Q},\mathcal{L}}$ and, second, the subsequent unfolding of the atom $Q^p(\bar{x})$. Given the correctness and termination of the unfolding process in non-recursive definitions, the soundness of R^c and R^p follows from Theorem 8. □

Proposition 28 *Algorithm 4.2 produces a query whose size is:*[5]

- *Constant in the size of the database D.*
- *Linear in the size of the original query Q, where the linear factor C is:*
  - *Polynomial in the maximal size of the windows of expertise.*
  - *Exponential in the depth of the dependency relation between relations (i.e. the rank of the highest predicate).*

Proof: The size of a rewritten query is obviously constant in $|D|$, since R^c and R^p do not depend on D. Termination of the rewriting process implemented by R^c and R^p follows from the fact that $\mathfrak{D}$ is hierarchically closed. Let us define the rank of a database predicate P_i as the longest path in the LCWA-dependency graph from one of $\mathbf{t}, \mathbf{f}$ to P_i. To reduce an atom of rank n to a formula with atoms of at most rank $n-1$, blows up the atom with a factor less than W, the size of the largest window of expertise. To continue the reduction till the end blows up the atom with a factor W^n where n is the highest rank of a database predicate. So, a database atom in $\mathcal{Q}[\bar{x}]$ is blown up to a formula of maximally W^n. This is the factor C. □

As a corollary, we obtain that the algorithm retains tractability in the size of the database.

Theorem 10 (Complexity) *Given a hierarchically closed database $\mathfrak{D}$ and a query $\mathcal{Q}$, the computation time of $Cert_A(\mathcal{Q}[\bar{x}])$ and $Poss_A(\mathcal{Q}[\bar{x}])$ by Algorithm 4.2 is polynomial in $|D|$.*

[5]The size of a formula is expressed in number of atoms.

4.3.5 Towards Approximate Query Answering with Integrity Constraints

So far, local closed-world assumptions were the only source of negative information in $\mathfrak{D}$ (i.e. of provably false database atoms). In practice, another important source of negative information are integrity constraints, functional dependency constraints in particular. Assume, for instance, a key constraint on the n-th argument of a predicate P. In that case, the presence of the atom $P(a_1, \ldots, a_{n-1}, a)$ in the database entails the falsity of any atom $P(a_1, \ldots, a_{n-1}, b)$ for which $b \neq a$.

In this section, we show how the approximate reasoning method can be extended for handling functional dependency constraints. For the rest of this section, let a locally closed database $\mathfrak{D}$ consist of a database instance D, a set of local closed-world assumptions $\mathcal{L}$, and a set F of functional dependencies.

Definition 4.13 *The constraint that position l of predicate P functionally depends on argument positions $i_1, \ldots, i_k$ will be denoted $P : (i_1, \ldots, i_k) \rightarrow l$.*

Definition 4.14 *The logical semantics of such a constraint is the first-order formula*

$$\forall x_1, \ldots, x_n, y_1, \ldots, y_n P(x_1, \ldots, x_n) \wedge P(y_1, \ldots, y_n) \wedge \\ x_{i_1} = y_{i_1} \wedge \cdots \wedge x_{i_k} = y_{i_k} \supset x_l = y_l$$

For such a database $\mathfrak{D}$ and query $\mathcal{Q}[\bar{x}]$, we define $\Delta^c_{\mathcal{Q},\mathfrak{D}}$ as in Definition 4.10, except that for each $P : (i_1, \ldots, i_k) \rightarrow l \in F$, the following rule is added:

$$\forall (P^{c\neg}(x_1, \ldots, x_n) \leftarrow P^c(y_1, \ldots, y_n) \wedge x_{i_1} = y_{i_1} \wedge \cdots \wedge x_{i_1} = y_{i_1} \wedge x_l \neq y_l).$$

This rule infers that $P(x_1, \ldots, x_n)$ is false if D contains an atom $P(y_1, \ldots, y_n)$ which is identical with the previous one on argument positions $i_1, \ldots, i_k$ and different on position l.

Example 4.15 *In Example 4.11, each license plate identifies the car, and hence the car brand. This is the constraint $CarO : 3 \rightarrow 2$. Its logical semantics is given by the formula:*

$$\forall n, m, id, n_1, m_1 : CarO(n, m, id) \wedge CarO(n_1, m_1, id) \supset m = m_1$$

The rule set in the fixpoint expression can now be extended to:

$$\Delta^c_{\mathcal{Q},\mathcal{L}} = \left\{ \begin{array}{l} Q^c(n, m, id) \leftarrow CarO^c(n, m, id). \\ Q^{c\neg}(n, m, id) \leftarrow CarO^{c\neg}(n, m, id). \\ CarO^c(n, m, id) \leftarrow CarO(n, m, id). \\ CarO^{c\neg}(n, m, id) \leftarrow \neg CarO(n, m, id) \wedge Loc^c(n, Bx). \\ Loc^c(n, l) \leftarrow Loc(n, l). \\ Loc^{c\neg}(n, l) \leftarrow \neg Loc(n, l) \wedge l = Bx \wedge \exists m, id : CarO^c(n, m, id). \\ \mathbf{CarO^{c\neg}(n, m, id) \leftarrow CarO^c(n_1, m_1, id) \wedge m \neq m_1.} \end{array} \right\}$$

The last rule is the additional one. The two rules for $CarO^{c\neg}$ *in this set can be replaced by the unique rule:*

$$CarO^{c\neg}(n,m,id) \leftarrow \neg CarO(n,m,id) \wedge Loc^c(n,Bx) \vee \exists n_1,m_1 CarO^c(n_1,m_1,id) \wedge m \neq m_1.$$

Unfolding the query

$$\neg[lfp_{Q^p,\Delta^p_{\mathcal{Q},\mathcal{L}}}](n,m,id)$$

for the possible answers of $CarO(n,m,id)$ *yields:*

$$(CarO(n,m,id) \vee \neg Loc(n,Bx)) \wedge \forall n_1,m_1 \neg CarO(n_1,m_1,id) \vee m = m_1.$$

Or, equivalently:

$$(CarO(n,m,id) \vee \neg Loc(n,Bx)) \wedge \forall n_1,m_1 CarO(n_1,m_1,id) \supset m = m_1.$$

In particular, the possible answers in the extended database are the following:

$$(PS,M,Q1),\ (PS,V,B1),\ (PS,B,B2).$$

If, in addition, a car could belong to at most one person (i.e. $CarO : 3 \rightarrow 1$*), the additional rule is:*

$$CarO^{c\neg}(n,m,id) \leftarrow CarO^c(n_1,m_1,id) \wedge n \neq n_1.$$

Possible answers for $CarO(PS,n,m)$ *are now reduced to* $\{(PS,M,Q1)\}$*, i.e. there is CWI on this query.*

The following theorem shows that query answering within extended fixpoint queries framework are indeed sound approximations.

Theorem 11 *Given a database* $\mathfrak{D}$ *and query* $\mathcal{Q}[\bar{x}]$*. Let* $\mathcal{C}$ *denote the answers of* $[lfp_{Q^c,\Delta^c_{\mathcal{Q},\mathfrak{D}}}](\bar{x})$ *in* D*, and* $\mathcal{P}$ *the answers of* $\neg[lfp_{Q^{c\neg},\Delta^c_{\mathcal{Q},\mathfrak{D}}}](\bar{x})$ *in* D*. It holds that*

$$\mathcal{C} \subseteq Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) \subseteq Poss_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) \subseteq \mathcal{P}.$$

Proof: It is easy to extend the fixpoint operator $\mathcal{A}pp_{\mathfrak{D}}$ to incorporate functional dependencies. The extended operator is defined as follows:

$$P(\bar{a})^{\mathcal{K}'} = \begin{cases} \mathbf{t} & \text{if } P(\bar{a}) \in D, \\ \mathbf{f} & \text{if } P(\bar{a}) \notin D \text{ and there exists } \mathcal{LCWA}(P(\bar{x}),\Psi_P[\bar{x}]) \in \mathcal{L} \\ & \text{such that } \Psi_P[\bar{a}]^{\mathcal{K}} = \mathbf{t}, \\ \mathbf{f} & \text{if } P(\bar{a}) \notin D \text{ and } P:(i_1,\ldots,i_k) \rightarrow l \in \mathcal{L} \\ & \text{and for some } P(\bar{d})\text{: } P(\bar{d})^{\mathcal{K}} = \mathbf{t} \\ & \text{and } d_{i_1} = a_{i_1},\ldots,d_{i_k} = a_{i_k} \text{ but } d_l \neq a_l, \\ \mathbf{u} & \text{otherwise.} \end{cases}$$

It is easy to extend the proof of Theorem 7 to show that this operator produces an approximation of all models of $\mathfrak{D}$ (which now contains also functional dependency constraints). To show that the extended fixpoint query is correct, one can show that the operator of the extended rule set simulates $\mathcal{A}pp_{\mathfrak{D}}$. This is a straightforward extension of the proof of Theorem 8. □

4.3.6 Summary of the Results on Approximations

Here is a summary of the results obtained in this section.

- In Section 4.2.2 we defined an approximation of a first-order theory and provided a method based on a fixpoint operator to compute an approximation for a locally complete database. Next, we demonstrated how to pose queries to such an approximation and we proved that the retrieved answers were sound approximations of possible or certain answers. Also, we proved that the algorithm for computing the three-valued structure $\mathcal{C}_{\mathfrak{D}}$ and the querying mechanism run both in polynomial time and hence the whole query processing is tractable. This result is an improvement over the complexity of computing certain and possible answers from $\mathfrak{D}$, which were in coNP and NP respectively.

- We then addressed one of the drawbacks of the above method, namely the need to compute explicitly the database approximation. Towards this goal, we developed an algorithm to compute a fixpoint query that simulates the three-valued interpretation $\mathcal{C}_{\mathfrak{D}}$. We proved that this technique produces sound answers.

- For the special class of hierarchically closed databases we presented an algorithm that, given an initial query $\mathcal{Q}$, computes a first-order query $\mathcal{Q}'$ that, posed to the original database, retrieves the certain or possible answers for $\mathcal{Q}$.

- Finally we extended the framework to handle certain types of integrity constraints, namely key constraints.

The results obtained so far provide methods to find approximate answers from different types of locally complete databases. In the next section we investigate the accuracy of such methods, i.e. we investigate how reasonable are the answers to be expected from this framework.

4.4 Accuracy of Approximate Query Answering

4.4.1 Introduction

In this section we study the accuracy of the query answering methods proposed in this chapter. The question we address in this section is for which types of queries and databases the query answering mechanisms of the previous sections return optimal answers, where optimal will be understood as answers that are equivalent to those obtained from the meaning of the database.

Recall that all the query answering methods introduced in the previous sections are tractable, i.e. they run in polynomial time in the size of the database.

This section is organized as follows:

- In Section 4.4.2 we present examples in which the methods introduced in the last section do not return exact answers. Not surprisingly, these examples suggest that the quality of query answers depends on the precision of the computed approximation.
- Before addressing the problem in locally complete databases, we start with a general study about precision of three-valued interpretation. To such an end, we introduce the concept of square query.
- We present a toolbox that will allow us to derive conditions for which query answering is optimal. In addition, we present specific conditions for queries and windows of expertise that ensure that the answers so obtained are optimal.

4.4.2 Motivational Examples

A weakness of Corollary 3 is that it gives no guarantee whatsoever on the accuracy of the returned answers of the approximate query answering method. As the following example illustrates, in certain specific cases, all accuracy in approximate query answering might be lost.

Example 4.16 *Let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database over vocabulary Σ. Assume that Σ contains the predicate symbol P such that P does not appear in D, and $\mathcal{L}$ contains for each database predicate the LCWA*

$$\mathcal{LCWA}(Q(\bar{x}), P \vee \neg P)$$

Since this window of expertise is a tautology, this database has complete knowledge on all predicates. Or, $\mathfrak{D}$ is equivalent to the standard database D under closed-world assumption. I.e. the LCWAs are all logically equivalent to

$$\mathcal{LCWA}(Q(\bar{x}), \mathbf{t})$$

On the other hand, it is easy to see that $(P \vee \neg P)^{\mathcal{C}_\mathfrak{D}} = \mathbf{u}$. It follows that for each atom $Q(\bar{a})$, we have $Q(\bar{a})^{\mathcal{C}_\mathfrak{D}} = \mathbf{u}$ whenever $Q(\bar{a}) \notin D$. This boils down to the fact that $\mathcal{C}_\mathfrak{D}$ represents the database under OWA. I.e. all the additional information stored in the LCWA expressions is lost in the computed three-valued structure $\mathcal{C}_\mathfrak{D}$. As a consequence, many queries cannot be accurately answered using $\mathcal{C}_\mathfrak{D}$. For example, any query $\neg Q(\bar{a})$, for $Q(\bar{a}) \notin D$ has truth value $\mathbf{u}$ in $\mathcal{C}_\mathfrak{D}$ but is implied by $\mathfrak{D}$.

The above example shows the desirability of results that somehow demonstrate the accuracy of the approximate query answering method. In this section, we investigate conditions on $\mathfrak{D}$ that guarantee that $Cert_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}[\bar{x}]) =$

$\text{Cert}_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$. We will not be concerned here with the accuracy of possible answers. Indeed, from the proposition below, which follows directly from the dualization on the definitions of possible and certain answers, that such conditions follow immediately from those for certain answers.

Proposition 29 *For a given database $\mathfrak{D}$ and query $\mathcal{Q}[\bar{x}]$, then $Poss_{\mathcal{C}_{\mathfrak{D}}}(\mathcal{Q}[\bar{x}])$= $\text{Poss}_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$ if and only if $Cert_{\mathcal{C}_{\mathfrak{D}}}(\neg\mathcal{Q}[\bar{x}]) = \text{Cert}_{\mathfrak{D}}(\neg\mathcal{Q}[\bar{x}])$.*

We start our investigation with some examples that illustrate different ways in which optimality of certain answers to a query may be lost.

- **Weakness of the standard Kleene truth evaluation**. Indeed, the evaluation of the query under the standard Kleene truth assignment may be too weak. For example, for $\mathcal{R}(\Sigma) = \{P\}$ and $\mathfrak{D} = (\emptyset, \emptyset)$, we have that $\mathcal{C}_{\mathfrak{D}} = \mathcal{O}_{\mathfrak{D}} = \bot$, the least precise approximation. The query $P \vee \neg P$ has truth value **u** in $\bot$ under the Kleene truth assignment, but **t** under supervaluation (Definition 4.2). Thus, if we are prepared to pay the computational cost of using supervaluation during the construction of $\mathcal{C}_{\mathfrak{D}}$ (Denecker et al. 2004) (which takes us back to intractability) instead of the standard Kleene truth assignment, the loss of precision can be avoided in this case.

- **Three-value structures cannot capture all the information in $\mathfrak{D}$.** In other cases, the loss of accuracy is due to the fact that three-valued structures such as $\mathcal{C}_{\mathfrak{D}}$ and $\mathcal{O}_{\mathfrak{D}}$ are only coarse representations of the information in $\mathfrak{D}$.

  Example 4.17 *Consider the propositional locally closed database $\mathfrak{D} = (D, \mathcal{L})$, where $\mathcal{R}(\Sigma) = \{Q, R\}$, $D = \emptyset$, and $\mathcal{L} = \{\mathcal{LCWA}(Q, R)\}$. This database is equivalent to $\neg R \vee \neg Q$; therefore, $\mathcal{O}_{\mathfrak{D}} = \bot = \mathcal{C}_{\mathfrak{D}}$, i.e. Q and R are* **u**. *Consequently, the query $\neg R \vee \neg Q$ evaluates to unknown in $\mathcal{C}_{\mathfrak{D}}$ but is certainly satisfied in $\mathfrak{D}$.*

  In this case, the loss of precision of the approximate query answering method is not due to the weakness of Kleene truth assignment, but due to the weakness of three-valued structures as a way of representing incomplete knowledge. Three-valued approximations do not record certain logical relationships between predicates. In particular, Q and R are unknown indeed, yet $\neg R \vee \neg Q$ is known to be true in $\mathfrak{D}$, and this information cannot be captured by a partial structure. Using supervaluation cannot save the precision on this query.

- **$\mathcal{C}_{\mathfrak{D}}$ is not precise enough**. As illustrated in Example 4.16, a third cause for a loss of accuracy of computed approximate answers is when $\mathcal{C}_{\mathfrak{D}}$ is not sufficiently precise.

Clearly, the quality of query answers depends greatly on the precision of $\mathcal{C}_{\mathfrak{D}}$. Below, we illustrate some ways in which precision of $\mathcal{C}_{\mathfrak{D}}$ can be lost.

- **Window of expertise can not accurately be queried**. The optimality of $\mathcal{C}_{\mathfrak{D}}$ may be lost due to the fact that a window of expertise cannot accurately be queried. This was illustrated in Example 4.16, with the window of expertise $P \vee \neg P$. Another illustration occurs when we extend Example 4.17 by adding P to $\mathcal{R}(\Sigma)$ and adding $\mathcal{LCWA}(P, \neg R \vee \neg Q)$ to $\mathcal{L}$. In this case, the loss of accuracy of $\mathcal{C}_{\mathfrak{D}}$ on the query $\neg R \vee \neg Q$, which is the window of expertise of P, propagates to a loss of accuracy on P. Indeed, we have $P^{\mathcal{O}_{\mathfrak{D}}} = \mathbf{f}$, while $P^{\mathcal{C}_{\mathfrak{D}}} = \mathbf{u}$.
- **Precision of $\mathcal{C}_{\mathfrak{D}}$ may be lost when cycles exist in the local closed-world assumptions**. In the optimal approximation of

$$\mathfrak{D} = (\{\}, \{\mathcal{LCWA}(P, P)\}),$$

P is false since $\mathfrak{D}$ is equivalent to $\neg P$, while P is unknown in the constructed approximation of $\mathfrak{D}$, since P is unknown in the initial step of the construction and remains unknown when applying $\mathcal{A}pp_{\mathfrak{D}}$.

The above discussion shows that the loss of accuracy on queries and the loss of precision of the computed approximation are related problems. In particular, a frequent cause for the latter is a loss of accuracy on the windows of expertise. In the next sections, we study when accuracy can be guaranteed. Towards this end, we first introduce some useful auxiliary concepts and properties about modularity of locally complete databases.

4.4.3 Modularity Properties of Locally Closed Databases

As the last example shows, cycles in the LCWA dependency graph are an important property in the accuracy analysis. We now refine the concept of LCWA dependency graph of $\mathfrak{D}$. Below, the *transitive closure of the LCWA-dependency graph* will be denoted $\prec^-$.

Definition 4.15 (Full LCWA dependency graph) *The* full LCWA dependency graph *is the graph on $\mathcal{R}(\Sigma)$ consisting of edges (Q, P) such that Q occurs positively or negatively in a window of expertise of P. Its transitive closure is denoted $\prec$.*

The reflexive closures of $\prec$ and $\prec^-$ are denoted $\preceq$ and $\preceq^-$, respectively.

A strict hierarchically closed database *is one with a cycle-free full LCWA-dependency graph.*

Example 4.18 *We illustrate the concept of full LCWA dependency graph with some familiar examples:*

- *Because of the dependency of P on itself in the windows of expertise, any locally closed database with an expression of the form $\mathcal{LCWA}(P, P)$ is not strict hierarchically closed.*

- *The locally closed database of Example 4.11, with*

$$\mathcal{L}_1 = \left\{ \begin{array}{l} \mathcal{LCWA}(CarO(n, m, id), Loc(n, Bx)), \\ \mathcal{LCWA}(Loc(n, l), (l = Bx \wedge \exists n, m : CarO(n, m, id)) \end{array} \right\}$$

  is hierarchically closed but not strict hierarchically closed. Here we have a mutual dependency of predicates: $CarO$ depends on Loc and vice-versa. Since this dependency does not involve negative windows of expertise $\mathcal{L}$ is hierarchically closed.

Sometimes, given a query $\mathcal{Q}[\bar{x}]$, it will be convenient to extend the LCWA-dependency graph with $\mathcal{Q}[\bar{x}]$ by adding the query as a node and adding edges to it from each predicate P that occurs negatively in $\mathcal{Q}[\bar{x}]$. Also the full LCWA-dependency graph can be extended in this way, but here edges are added from each predicate with an arbitrary occurrence in $\mathcal{Q}[\bar{x}]$. Recall that $\prec^-$ is cycle-free if and only if for any query $\mathcal{Q}[\bar{x}]$, the set $\Delta_{\mathcal{Q},\mathcal{L}}$ (and also $\Delta'_{\mathcal{Q},\mathcal{L}}$) is non-recursive.

The following properties and definitions show that the accuracy of approximate answers to queries can be studied in a potentially smaller database.

Definition 4.16 *Let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database. A set $\sigma \subseteq \mathcal{R}(\Sigma)$ is $\prec$-downward closed if $P \in \sigma$ and $R \prec P$ then $R \in \sigma$.*

For such a set, we will consider below the restriction of $\mathfrak{D}$ to the predicates in σ. The pair $(D|_\sigma, \mathcal{L}|_\sigma)$ consisting of the restrictions of the structure D to the predicates in σ and the restriction of $\mathcal{L}$ to the LCWAs of the predicates in σ is denoted $\mathfrak{D}|_\sigma$. Clearly, $\mathfrak{D}|_\sigma$ is a locally closed database with the same domain as $\mathfrak{D}$. Note that predicates of σ may occur in the windows of expertise of predicates $Q \in \mathcal{R}(\Sigma) \setminus \sigma$ but not vice versa. Of interest are the $\prec$-downward closed sets $\{Q \mid Q \prec \tau\}$ and $\{Q \mid Q \preceq \tau\}$, for τ a database predicate or a query. We denote the restriction of $\mathfrak{D}$ to these sets by respectively $\mathfrak{D}|_{\prec\tau}$ and $\mathfrak{D}|_{\preceq\tau}$.

Lemma 1 (Extendibility) *If σ is a $\prec$-downward closed set of predicates, then each σ-model M of $\mathfrak{D}|_\sigma$ can be expanded to a model N of $\mathfrak{D}$ by setting $Q^N = Q^D$, for every $Q \in \mathcal{R}(\Sigma) \setminus \sigma$.*

Proof: Let M and N be as above. Clearly, N satisfies all database atoms, all LCWAs of $\mathfrak{D}|_\sigma$ and all LCWAs for predicates $Q \in \mathcal{R}(\Sigma) \setminus \sigma$ (since $N \models \forall\bar{x}(Q(\bar{x}) \supset Q(\bar{x}) \in D)$). Hence, $N \models \mathfrak{D}$. □

The above lemma allows for deriving some interesting modularity properties.

Proposition 30 *For any query $\mathcal{Q}[\bar{x}]$, we have:*

- $Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) = Cert_{\mathfrak{D}|_{\prec \mathcal{Q}}}(\mathcal{Q}[\bar{x}])$,
- $Poss_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) = Poss_{\mathfrak{D}|_{\prec \mathcal{Q}}}(\mathcal{Q}[\bar{x}])$.

Proof: It follows from the Extendibility Lemma 1 that for any $\prec$-downward closed set σ and any σ-formula φ, $\mathfrak{D} \models \varphi$ if and only if $\mathfrak{D}|_\sigma \models \varphi$, and $\mathfrak{D} \cup \{\varphi\}$ is satisfiable if and only if $\mathfrak{D}|_\sigma \cup \{\varphi\}$ is satisfiable. Both equations follow immediately. □

As a corollary to this proposition, we have the following proposition on CWI.

Corollary 4 *$\mathfrak{D}$ has CWI on $\mathcal{Q}[\bar{x}]$ if and only if $\mathfrak{D}|_{\prec \mathcal{Q}[\bar{x}]}$ has CWI on $\mathcal{Q}[\bar{x}]$.*

Proposition 31 *For any query $\mathcal{Q}[\bar{x}]$, we have:*

- $\mathcal{O}_{\mathfrak{D}}|_{\prec \mathcal{Q}} = \mathcal{O}_{(\mathfrak{D}|_{\prec \mathcal{Q}})}$,
- $\mathcal{C}_{\mathfrak{D}}|_{\prec \mathcal{Q}} = \mathcal{C}_{(\mathfrak{D}|_{\prec \mathcal{Q}})}$.

Proof: The first equation is an easy consequence of the Extendibility Lemma 1. For the second, observe that in computing the predicates of σ, the operator $\mathcal{A}pp_{\mathfrak{D}}$ only uses $\mathfrak{D}|_\sigma$. □

A direct consequence of this proposition is that:

- $Cert_{\mathcal{O}_{\mathfrak{D}}}(\mathcal{Q}[\bar{x}]) = Cert_{\mathcal{O}_{(\mathfrak{D}|_{\prec \mathcal{Q}})}}(\mathcal{Q}[\bar{x}])$, and
- $Cert_{\mathcal{C}_{\mathfrak{D}}}(\mathcal{Q}[\bar{x}]) = Cert_{\mathcal{C}_{(\mathfrak{D}|_{\prec \mathcal{Q}})}}(\mathcal{Q}[\bar{x}])$.

The above propositions show that the accuracy of approximate answers for a query $\mathcal{Q}[\bar{x}]$ can be studied in the potentially much smaller database $\mathfrak{D}|_{\prec \mathcal{Q}}$.

Our accuracy analysis starts off with a study of modularity properties of locally closed databases.

4.4.4 Squared Queries

This section introduces the base concept of accuracy analysis: the notion of a *squared query*. A query is squared when all its certain answers are true in the optimal approximation. Formally:

Definition 4.17 (Squared queries) *Let Γ be a satisfiable theory over Σ containing $\mathsf{DCA}(\Sigma) \wedge \mathsf{UNA}(\Sigma)$, and let $HU(\Sigma)$ be the Herbrand universe for Σ. We say that a query $\mathcal{Q}[\bar{x}]$ is* squared *in Γ if for every $\bar{d} \in HU(\Sigma)^n$, if $\Gamma \models \mathcal{Q}[\bar{d}]$ then $\mathcal{Q}[\bar{d}]^{\mathcal{O}_\Gamma} = \mathbf{t}$.*

Example 4.19 *We illustrate the concept of squared queries directly revisiting some locally complete databases already introduced.*

- *Consider again the database* $\mathfrak{D} = (\emptyset, \emptyset)$*, where* $\mathcal{R}(\Sigma) = \{P\}$ *and* $\mathfrak{D} = (\emptyset, \emptyset)$*, we have that* $\mathcal{C}_{\mathfrak{D}} = \mathcal{O}_{\mathfrak{D}} = \bot$*, the least precise approximation. The query* $\mathcal{Q} = P \vee \neg P$ *is not squared. Indeed, since* Q *is a tautology it holds that* $\mathfrak{D} \models Q$*, but* $Q^{\mathcal{O}_{\mathfrak{D}}} = \mathbf{u}$*.*

- *Another query that is not squared is* $\neg R \vee \neg Q$ *in the database of Example 4.17:* D *is empty, and* $\mathcal{L} = \{\mathcal{LCWA}(Q, R)\}$*. Since the database is equivalent to* $\neg R \vee \neg Q$*, it holds that* $\mathfrak{D} \models \neg R \vee \neg Q$*. However, since* $\mathcal{O}_{\mathfrak{D}} = \bot = \mathcal{C}_{\mathfrak{D}}$*, the query evaluates to unknown in* $\mathcal{O}_{\mathfrak{D}}$*.*

- *The query* P *in database is* $D = (\{P, Q\}, \emptyset)$ *is squared. Indeed,* P *is an atom of the database and hence* $\mathfrak{D} \models P$ *and* $P^{\mathcal{O}_{\mathfrak{D}}} = \mathbf{t}$*.*

It is easy to see that a query $\mathcal{Q}[\bar{x}]$ is squared if and only if for arbitrary $\bar{d} \in HU(\Sigma)^n$, $\mathcal{Q}[\bar{d}]^{\mathcal{O}_\Gamma} = \mathbf{u}$ implies that $\Gamma \cup \{\neg\mathcal{Q}[\bar{d}]\}$ is satisfiable. This does not mean that for each $\bar{d} \in HU(\Sigma)^n$, $\Gamma \cup \{\mathcal{Q}[\bar{d}]\}$ is satisfiable. For instance, $P \wedge \neg P$ is squared in the empty propositional theory, but $P \wedge \neg P$ is not satisfiable. The next proposition follows directly from the definitions of squared query and possible and certain answers:

Proposition 32 *The query* $\mathcal{Q}[\bar{x}]$ *is squared in* Γ *if and only if* $Cert_{\mathcal{O}_\Gamma}(\mathcal{Q}[\bar{x}]) = Cert_\Gamma(\mathcal{Q}[\bar{x}])$*. The query* $\neg\mathcal{Q}[\bar{x}]$ *is squared in* Γ *if and only if* $Poss_{\mathcal{O}_\Gamma}(\mathcal{Q}[\bar{x}]) = Poss_\Gamma(\mathcal{Q}[\bar{x}])$*.*

The above proposition shows that if we can construct the optimal approximation $\mathcal{O}_\Gamma$ of Γ, then we can compute all certain answers of all squared queries in Γ using $\mathcal{O}_\Gamma$. The proposition below shows that when Γ has a least model I, we can compute all certain answers of such queries using weaker approximations than $\mathcal{O}_\Gamma$. Observe that for any atom $P(\bar{a})$, $P(\bar{a})^I = \mathbf{t}$ if and only if $P(\bar{a})^{\mathcal{O}_\Gamma} = \mathbf{t}$.

Proposition 33 *Let* $\mathcal{Q}[\bar{x}]$ *be squared in* Γ*. Assume that* Γ *has a least model* I *and let* $\mathcal{K}$ *be an arbitrary approximation of* Γ *such that for each atom* $P(\bar{a})$*,* $P(\bar{a})^I = \mathbf{t}$ *if and only if* $P(\bar{a})^{\mathcal{K}} = \mathbf{t}$*. If* $P^{\mathcal{K}} = P^{\mathcal{O}_\Gamma}$ *for every predicate* P *that occurs negatively in* $\mathcal{Q}[\bar{x}]$*, then* $Cert_{\mathcal{K}}(\mathcal{Q}[\bar{x}]) = Cert_\Gamma(\mathcal{Q}[\bar{x}])$*.*

Proof: Since $\mathcal{K} \leqslant_{\mathrm{p}} \mathcal{O}_\Gamma$, we have that:

$$Cert_{\mathcal{K}}(\mathcal{Q}[\bar{x}]) \subseteq Cert_{\mathcal{O}_\Gamma}(\mathcal{Q}[\bar{x}]) = Cert_\Gamma(\mathcal{Q}[\bar{x}]).$$

To prove the other direction, let us observe that all predicates such that $P^{\mathcal{K}} \neq P^{\mathcal{O}_\Gamma}$ occur only positively in $\mathcal{Q}[\bar{x}]$ and moreover $P^{\mathcal{O}_{\mathfrak{D}}} \leq P^{\mathcal{K}}$ (where $\leq$ is the truth order, not the precision order). Indeed, for any atom $P(\bar{a})$, if $P(\bar{a})^{\mathcal{O}_{\mathfrak{D}}} = \mathbf{u}$, then since $\mathcal{K} \leqslant_{\mathrm{p}} \mathcal{O}_{\mathfrak{D}}$, we have $P(\bar{a})^{\mathcal{K}} = \mathbf{u}$, and if $P(\bar{a})^{\mathcal{O}_{\mathfrak{D}}} = \mathbf{t}$, then $P(\bar{a})^I = \mathbf{t} = P(\bar{a})^{\mathcal{K}}$. It follows from a standard monotonicity property of FO that for arbitrary $\bar{d}$, $\mathcal{Q}[\bar{d}]^{\mathcal{O}_\Gamma} \leq \mathcal{Q}[\bar{d}]^{\mathcal{K}}$. Hence, $Cert_{\mathcal{K}}(\mathcal{Q}[\bar{x}]) \supseteq Cert_{\mathcal{O}_\Gamma}(\mathcal{Q}[\bar{x}])$.

□

The proposition expresses that we can compute optimal answers for $\mathcal{Q}[\bar{x}]$ using a weaker than optimal approximation $\mathcal{K}$ which is unknown on atoms $P(\bar{a})$ that are false in $\mathcal{C}_\Gamma$, as long as the predicates P occur only positively in the query.

This proposition furnishes our first completeness results regarding locally complete databases. Let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database over Σ and $\mathcal{Q}[\bar{x}]$ a query. Below, we denote by $P \in^- \mathcal{Q}[\bar{x}]$ that P has a negative occurrence in $\mathcal{Q}[\bar{x}]$ and by $P \in^+ \mathcal{Q}[\bar{x}]$ that P occurs positively in $\mathcal{Q}[\bar{x}]$.

Theorem 12 (Completeness) *It holds that $Cert_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}[\bar{x}]) = Cert_\mathfrak{D}(\mathcal{Q}[\bar{x}])$ if $\mathcal{Q}[\bar{x}]$ is squared in $\mathfrak{D}$ and $P^{\mathcal{C}_\mathfrak{D}} = P^{\mathcal{O}_\mathfrak{D}}$, for every database predicate $P \in^- \mathcal{Q}[\bar{x}]$.*

Proof: Note that D is the least model of $\mathfrak{D}$ and that by construction, $P(\bar{a})^{\mathcal{C}_\mathfrak{D}} = \mathbf{t}$ if and only if $P(\bar{a}) \in D$. The theorem now follows from Proposition 33. □

Thus, to prove that a query can be optimally answered using our approximate query answering techniques, we need to prove the squaredness of the query and the optimality of $\mathcal{C}_\mathfrak{D}$ on the (potentially small) set of negatively occurring predicates.

To get other concrete results from Theorem 12, we need sufficient conditions for the two assumptions under which it can be applied, namely:

1. Squaredness of queries, and
2. Optimality of predicate approximations.

We start with a syntactical condition for the former.

4.4.5 Proving Squaredness of Queries

Definition 4.18 (The set $\sigma^-_\mathcal{Q}$ and positive-free predicates) *For a query $\mathcal{Q}[\bar{x}]$, let*

$$\sigma^-_\mathcal{Q} = \{R \mid \exists Q \in^- \mathcal{Q}[\bar{x}] \text{ such that } R \preceq Q\}.$$

That is, $\sigma^-_\mathcal{Q}$ is the $\prec$-downward closure of the set of predicates that occur negatively in $\mathcal{Q}$. A predicate P is called positive free *in $\mathcal{Q}[\bar{x}]$ if $P \in \mathcal{R}(\Sigma) \setminus \sigma^-_\mathcal{Q}$.*

Observe that positive free predicates have only positive occurrences in $\mathcal{Q}[\bar{x}]$ and if $\mathcal{Q}[\bar{x}]$ is a positive query, then all database predicates are positive free. Let $\mathfrak{D}_\mathcal{Q}$ be the extension of $\mathfrak{D} = (D, \mathcal{L})$ obtained by adding $\mathcal{LCWA}(P(\bar{x}), \mathbf{t})$ to $\mathcal{L}$, for each predicate P that is positive free in $\mathcal{Q}[\bar{x}]$.

Example 4.20 *In Example 4.11, all the predicates are positive free for predicate $CarO(n, m, id)$, and none of them is positive free for $\neg CarO(n, m, id)$, since $CarO$ depends on Loc.*

Proposition 34 *If $\mathcal{Q}[\bar{x}]$ is squared in $\mathfrak{D}_\mathcal{Q}$, then $\mathcal{Q}[\bar{x}]$ is squared in $\mathfrak{D}$. For every predicate P, $P^{\mathcal{O}_{\mathfrak{D}_\mathcal{Q}}} = P^{\mathcal{O}_\mathfrak{D}}$ if P is positive free in $\mathcal{Q}[\bar{x}]$, and $P^{\mathcal{O}_{\mathfrak{D}_\mathcal{Q}}} = P^D$ otherwise*[6].

Definition 4.19 *Two conjunctions C_1 and C_2 are called* mutually exclusive *with respect to a three-valued structure $\mathcal{K}$, if C_1 has a conjunct whose negation is a conjunct of C_2, and that contains only predicates that are two-valued in $\mathcal{K}$.*

The propositions below provide conditions for squared queries.

Theorem 13 (Squaredness) *A query $\mathcal{Q}[\bar{x}]$ is squared in $\mathfrak{D}_\mathcal{Q}$ if it is of the form*

$$\forall \bar{y} \left(C_1 \vee \cdots \vee C_n \right)$$

where each C_i is a conjunction in which

1. *each non-literal conjunct C_{ij} is an arbitrary subformula containing only two-valued predicates in $\mathcal{O}_\mathfrak{D}$ and (positive occurrences of) positive free predicates in $\mathcal{Q}[\bar{x}]$,*
2. *each pair C_i, C_j $(i \neq j)$ is mutually exclusive with respect to $\mathcal{O}_\mathfrak{D}$.*

Since $\mathcal{C}_\mathfrak{D} \leqslant_\mathrm{p} \mathcal{O}_\mathfrak{D}$, the conditions of the theorem can be verified by using $\mathcal{C}_\mathfrak{D}$ instead of $\mathcal{O}_\mathfrak{D}$. I.e. by checking if non-literal conjuncts C_{ij} consist of two-valued predicates in $\mathcal{C}_\mathfrak{D}$, and if each pair C_i, C_j $(i \neq j)$ contains complementary conjuncts $\varphi[\bar{x}]$ and $\neg\varphi[\bar{x}]$, such that $\varphi[\bar{d}]^{\mathcal{C}_\mathfrak{D}}$ is two-valued, for all $\bar{d} \in Dom^n$. The latter is the case if φ consists of base predicates only.

The class of queries covered by this theorem comprises:

- Literals and conjunctions of literals,
- Positive formulas

The class of queries covered by Theorem 13 includes positive conjunctive queries. This class is widely considered a keystone in practical relational databases systems, and it has consistently been identified as the most common class of queries appearing in databases practice (Grohe et al. 2001). As such, how to efficiently answer positive conjunctive queries has been extensively studied in diverse contexts related to the closed world assumption: incomplete databases (Grahne 1989), data integration systems (Grahne and Mendelzon 1999a; Duschka et al. 2000; Lenzerini 2002), data exchange (Libkin 2006) and answering queries using views (Levy et al. 1995). This bulk of research has shown that, very often, answering conjunctive queries enjoy several positive computational properties that make them suitable for efficient implementation.

[6] In order to improve its readability, some of the proofs of this section have been moved to Section 4.7.

In terms of expressiveness, the syntactic class of queries derived from Theorem 13 generalizes the type of queries considered by Levy for locally complete databases (Levy 1996) (positive conjunctive queries). In addition, Theorem 13 also generalizes the class of queries allowed in Motro's formalism (Motro 1989) (positive conjunctive queries).

Example 4.21 *The queries $CarO(n, m, id)$ and $\neg CarO(n, m, id)$ in Example 4.11 are squared in $\mathfrak{D}$.*

To define an alternative semantical condition for squaredness, we need the following definition.

Definition 4.20 *A theory Γ is atomical if and only if for each 2-valued Herbrand structure I such that $\mathcal{O}_\Gamma \leqslant_{\mathrm{p}} I$, it holds that $I \models \Gamma$.*

Observe that every consistent atomical theory Γ that includes $\mathsf{DCA}(\Sigma) \wedge \mathsf{UNA}(\Sigma)$ is equivalent to the first-order theory consisting of $\mathsf{DCA}(\Sigma) \wedge \mathsf{UNA}(\Sigma)$ and the set of all ground literals entailed by Γ.

Proposition 35 *$\mathcal{Q}[\bar{x}]$ is squared in $\mathfrak{D}_\mathcal{Q}$ if $\mathfrak{D}_\mathcal{Q}$ is atomical and $\mathcal{Q}[\bar{x}]$ is a formula such that all the predicates with positive and negative occurrences in $\mathcal{Q}[\bar{x}]$ are two-valued in $\mathcal{C}_\mathfrak{D}$.*

The class of queries covered by this proposition allows arbitrary quantification. Since $\mathfrak{D}_\mathcal{Q}$ is atomical if $\mathcal{Q}[\bar{x}]$ is a positive query, positive queries are covered by this proposition as well.

Additionally, it has been shown (Denecker et al. 2008) that $\mathfrak{D}$ is atomical if the database has CWI on the windows of expertise. From the discussion on CWI information in Section 3.5.1, in particular from propositions 12 and 14, it appears quite natural that locally closed databases will often meet this condition. In particular, databases only consisting of windows of expertise with base predicates fulfill this requirement.

4.4.6 Proving Partial Optimality of $\mathcal{C}_\mathfrak{D}$:

The second assumption of Theorem 13 requires the optimality of predicate approximations.

The next result reduces the problem of proving optimality of $\mathcal{C}_\mathfrak{D}$ in a predicate P to the problem of proving optimality of $\mathcal{C}_\mathfrak{D}$ in predicates Q that occur negatively in Ψ_P.

Proposition 36 *Let $\mathfrak{D}$ be a locally closed database. Then $P^{\mathcal{C}_\mathfrak{D}} = P^{\mathcal{O}_\mathfrak{D}}$ if the following conditions are satisfied:*

- *$\Psi_P[\bar{x}]$ is squared in $\mathfrak{D}$,*

- *for every predicate Q such that $Q \preceq^{-} P$, P does not occur positively in Ψ_Q,*
- *for every predicate Q that has a negative occurrence in Ψ_P, $Q^{\mathcal{C}_\mathfrak{D}} = Q^{\mathcal{O}_\mathfrak{D}}$.*

By induction on the depth of the strict (well-founded) order $\preceq^{-}$ on $\{Q \mid Q \preceq^{-} P\}$, using Proposition 36, the following way of showing optimality on P is obtained:

Proposition 37 *Let $\mathfrak{D}$ be a locally closed database. Then $P^{\mathcal{C}_\mathfrak{D}} = P^{\mathcal{O}_\mathfrak{D}}$ if for every predicate Q such that $Q \preceq^{-} P$, it holds that $Q \not\prec^{-} Q$, $\Psi_Q[\bar{x}]$ is squared in $\mathfrak{D}$, and Q does not occur positively in Ψ_S for any $S \preceq^{-} Q$.*

Note that the conditions of the proposition are satisfied if P's window of expertise is a positive formula and does not contain P. It is also satisfied if for every $Q \preceq^{-} P$, $\Psi_Q[\bar{x}]$ is squared in $\mathfrak{D}$ and $Q \not\prec Q$, i.e. the full LCWA dependency graph does not cycle through Q.

4.4.7 Summary of the Results on Accuracy of Approximations

By Theorem 13 and Proposition 37 we get the following theorem for the accuracy of approximative query answering.

Theorem 14 *Let $\mathcal{Q}[\bar{x}]$ be a query that is squared in $\mathfrak{D}$ and suppose that for every predicate P that occurs negatively in $\mathcal{Q}$ it holds that $P \not\prec^{-} P$, $\Psi_P[\bar{x}]$ is squared in $\mathfrak{D}$, and P does not occur positively in Ψ_Q for any $Q \preceq^{-} P$. Then $Cert_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}[\bar{x}]) = Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$.*

Example 4.22 *Consider again the database instance of Examples 1.5 and 3.3, together with the following LCWAs:*

- $\mathcal{LCWA}(\mathsf{Tel}(x, y), \mathsf{Dept}(x, \mathsf{CS}))$,
- $\mathcal{LCWA}(\mathsf{Dep}(x, y),\ y = \mathsf{CS} \wedge \exists z \mathsf{Tel}(x, z))$.

The first assumption is considered in Example 3.3; the second assumption expresses complete knowledge of the database about all the people of the computer science department that have a telephone.

This locally closed database is circular, as the window of expertise in each assumption mentions an object predicate of the other assumption. Yet, Theorem 14 is applicable here:

- *The windows of expertise are positive formulas, hence the contained predicates are positive free. It follows that both are squared in $\mathfrak{D}$. Also, $\prec^{-}$ is the empty graph, hence it is acyclic. It follows that $\mathcal{C}_\mathfrak{D}$ is optimal.*

- *The queries* $\mathsf{Tel}(x, y)$ *and* $\neg\mathsf{Tel}(x, y)$ *are literals, hence they are squared. It follows that the approximate methods compute the optimal answers for certain and possible answers of this query.*

So let us compute the answers for $\mathsf{Tel}(x, y)$ *by our approximate methods, using the following set* $\Delta_{\mathcal{Q},\mathcal{L}}$ *of certain rules (implicit universal quantification is assumed here):*

$$\left\{\begin{array}{l} Q^c(x,y) \leftarrow \mathsf{Tel}^c(x,y). \\ Q^{c\neg}(x,y) \leftarrow \mathsf{Tel}^{c\neg}(x,y). \\ \mathsf{Tel}^c(x,y) \leftarrow \mathsf{Tel}(x,y). \\ \mathsf{Tel}^{c\neg}(x,y) \leftarrow \neg\mathsf{Tel}(x,y) \wedge \mathsf{Dept}^c(x,\mathsf{CS}). \\ \mathsf{Dept}^c(x,y) \leftarrow \mathsf{Dept}(x,y). \\ \mathsf{Dept}^{c\neg}(x,y) \leftarrow \neg\mathsf{Dept}(x,y) \wedge y = \mathsf{CS} \wedge \exists z \mathsf{Tel}^c(x,z). \end{array}\right\}$$

Interestingly, this set of rules is non-recursive. It follows that its least and greatest fixpoint coincide. As expected, the expression $[lfp_{Q^c,\Delta_{\mathcal{Q},\mathcal{L}}}](x, y)$ *for* $\mathcal{Q} = \mathsf{Tel}(x, y)$*, gives the certain answers:*

$$\mathsf{Tel}(\mathsf{LD}, 6531421),\ \mathsf{Tel}(\mathsf{LD}, 09\text{-}23314),$$
$$\mathsf{Tel}(\mathsf{BD}, 5985625),\ \mathsf{Tel}(\mathsf{TD}, 5845213),$$

which are the database tuples of Tel*.*

To obtain the certainly false tuples of the query, we unfold the expression $[lfp_{Q^{c\neg},\Delta_{\mathcal{Q},\mathcal{L}}}](x, y)$*, yielding the database query* $\neg\mathsf{Tel}(x, y) \wedge \mathsf{Dept}(x, \mathsf{CS})$*. Its answer is the following set:*

$$\{\mathsf{Tel}(\mathsf{BD}, y) \mid y \neq 5985625\} \cup \{\mathsf{Tel}(\mathsf{TD}, y) \mid y \neq 5845213\}.$$

The set of possible but uncertain answers is the complement of the union of the certain answers and the certainly false answers. This set is specified by the database query $\neg\mathsf{Tel}(x, y) \wedge (\mathsf{Tel}(x, y) \vee \neg\mathsf{Dept}(x, \mathsf{CS}))$*, which can be simplified to* $\neg\mathsf{Tel}(x, y) \wedge \neg\mathsf{Dept}(x, \mathsf{CS})$*. Its answer is the set:*

$$\{\mathsf{Tel}(\mathsf{LD}, y) \mid y \neq 6531421 \wedge y \neq 09\text{-}23314\} \cup \{\mathsf{Tel}(\mathsf{DF}, y)\}.$$

As noted above, Theorem 14 guarantees that these query answers are precise.

To summarize, the results in this section allow to prove the optimality of the approximate certain answers in the context of queries $\mathcal{Q}[\bar{x}]$ and (a subset of the) windows of expertise in the form of positive formulas or conjunctions of literals, etc. Even larger classes of queries can be optimally answered with respect to atomical locally closed databases (Proposition 35). Clearly, this is a rich class of queries and a rich class of databases, for example allowing many forms of cycles in the LCWA-dependency graph. In particular, these results show that optimality is achieved for the important class of queries in the form of conjunctions of literals and on hierarchically closed databases with conjunctions of literals in the windows of expertise.

4.5 Domain-Independent and Safe Queries

We now return to the issue of safety and domain independence of queries for possible and certain answers in the context of locally closed databases. The problem of domain independence of a query for certain answers in standard (finite) databases is co-recursively enumerable, and the same holds for locally closed databases, both for possible and certain answers[7].

A related and practical question is the safety of approximate query answering, as this question is linked to computing certain or possible answers using relational algebra. Let us call a query $\mathcal{Q}[\bar{x}]$ *safe for certain (respectively, possible) answers* if the fixpoint query $[lfp_{Q^c,\mathrm{Def}^c_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ (respectively, $\neg[lfp_{Q^{c\neg},\mathrm{Def}^c_{\mathcal{Q},\mathcal{L}}}](\bar{x})$) is safe. As we have already seen in several examples, queries to locally closed databases are often unsafe, especially positive queries for possible answers and negative queries for certain answers. This is also demonstrated in the following example:

Example 4.23 *In Example 4.11, the certain answers for $CarO(n, m, id)$ can be obtained directly from the database (remember that $R^c(CarO(n, m, id)) = CarO(n, m, id)$). Clearly, this is a safe query. On the other hand, the possible answers for the same query are given by $CarO(n, m, id) \vee \neg Loc(n, Bx))$, and this set is domain dependent since m, id are unconstrained in the second disjunct. Also the query for certain answers of $\neg CarO(n, m, id)$ is unsafe, as its answers are those of posing the formula $\neg CarO(n, m, id) \wedge Loc(n, Bx)$ to the database.*

More generally, take a locally closed database in which all windows of expertise are positive formulas (hence, it is hierarchically closed). Then $R^p(\mathcal{Q}[\bar{x}])$ is obtained from $\mathcal{Q}[\bar{x}]$ by substituting every database predicate atom $P(\bar{t})$ in $\mathcal{Q}[\bar{x}]$ by $P(\bar{t}) \vee \neg\Psi_P[\bar{t}]$. Since the second disjunct is a negative formula, the resulting query will often be unsafe.

Domain dependence for possible query answering seems to be almost inherent to all approaches for incomplete databases, since such databases specify no or only few properties of unnamed objects that might or might not belong to the domain. Only by imposing stringent restrictions on the forms of queries one can guarantee domain independence. Below are some suggestions on how to deal with this problem.

- The database could maintain a universal predicate[8] U, representing all elements of the active domain. This predicate would be a base predicate. Queries would have to be relativized with respect to this predicate[9].

[7] To see this, compute for arbitrary large finite domains all models with that domain, and the possible or certain answers of the query. Check if these possible or certain answers for that domain are the same as those for the active domain. If not, then domain dependence holds.

[8] In the context of a sorted database, there would be a universal predicate for each sort.

[9] This is the solution that we implemented in our prototype system in Section 5.3.

- Symbolic query answering methods that return queries with constrained variables may be developed to correctly represent query answers in arbitrary domains. For example, possible answers for $CarO(n, m, id)$ in Example 4.11 may be represented by the set $\{(PS, m, id) \mid (m, id) \neq (M, Q1)\}$.

- Strong restrictions on queries may be imposed for possible answers. One simple syntactic condition could be that each variable in the query and in a window of expertise is constrained by a base predicate atom. I.e. queries $\mathcal{Q}[\bar{x}]$ and windows of expertise $\Psi_P[\bar{x}]$ have the form

$$P_1(\bar{t}_1) \wedge \cdots \wedge P_n(\bar{t}_n) \wedge \varphi,$$

  such that $P_1, \ldots, P_n$ are base predicates and each free variable x_i occurs in at least one atom $P_j(\bar{t}_j)$. In addition, all the quantifications in queries and windows of expertise are of the form $\exists x(P(\bar{t}) \wedge \ldots)$ or $\forall x(P(\bar{t}) \supset \ldots)$, where $P(\bar{t})$ is again a base predicate containing x. It is clear that such queries can be solved using relational algebra and so they are domain independent.

The next example shows, finally, that unsafe queries in standard relation databases are not necessary unsafe in locally closed databases.

Example 4.24 *Consider the locally closed database $\mathfrak{D} = (D, \mathcal{L})$, where $D = \{R(a)\}$ and $\mathcal{L} = \{\mathcal{LCWA}(R(x), \mathbf{t}), \mathcal{LCWA}(P(x), R(x))\}$, and the query $\mathcal{Q}[x] = \neg P(x)$. Since there is complete knowledge on R, the set of certain answers for $\mathcal{Q}[x]$ is $\{a\}$. Note that $R^c(\neg P(x)) = \neg P(x) \wedge R(x)$, which is a safe query.*

4.6 Conclusions

In this chapter, we have presented approximate methods for query answering in locally complete databases. The general approximation-based approach was motivated by the intractability results of Chapter 3 of computing certain and possible answers. In consequence, an essential requirement for all these methods was, as eventually shown, that the query answering mechanism would be tractable. The results we obtained can be summarized as follows:

- In Section 4.3, we presented a polynomial time algorithm to compute approximations of certain and possible answers. The algorithm consisted of two main tasks: first computing an explicit three-valued structure that approximates all the models of the database, and then posing the queries to it. The construction of the approximation was based on a fixpoint operator over three-valued approximations of the database.

- To address a major drawback of storing a concrete three-valued structure, namely the need to recompute the structure every time the database is updated, in Section 4.3.1 we refined the naive method by presenting a transformation to fixpoint queries that allow us to leave the three-valued structure implicit. As we will show in the next chapter, this method is easily implementable in systems that support fixpoint logics or logic programming.

- For the special class of databases called hierarchically closed, we introduced a query answering mechanism whose result is a standard first-order query that, when posed to the original database, retrieves approximations of the possible or certain answers. Since first-order query answering can be implemented in SQL (Ullman 1988), this method permits the use of standard database management systems.

In order to assess the accuracy of the approximation techniques, we obtained a number of results that guarantee optimality of answers for important classes of queries and locally complete databases. For this analysis, we introduced the notion of squared query, which allowed us to split the study in two independent but related concepts, namely proving squaredness of queries and proving the optimality of the three-valued structure. The concrete results on accuracy are summarized below:

- In subsection 4.4.2, we identified and showed by examples the sources for which precision may be lost when querying three-valued approximation of databases. We singled out the following reasons that may lead to lack of precision.

  - Query evaluation under standard Kleene truth assignment may be too weak.
  - The computed three-valued approximation of the database is too weak or is only a coarse representation of the information on the database $\mathfrak{D}$.

- In subsection 4.4.4, we introduced the concept of squared query, and we showed how the analysis of the accuracy methods can be reduced to the study of squared queries.

- In addition to semantic conditions on queries and databases that guarantee optimal answers, in subsection 4.4.5 we provided concrete syntactical classes of queries and databases for which the approximate method provide optimal answers. In particular, we were able to prove squaredness of queries for conjunction of literals and positive formulas. This class of queries have been recognized as relevant from both practical and computational point of view.

Currently, we lack real life experience to evaluate in practice the precision methods presented in this chapter. On one hand, we are able to construct examples showing a total loss of precision of the approximate answers. On the other hand, an analysis of the optimality proofs suggests that in many other applications where the conditions do not hold, the approximate methods should still be quite precise.

The approximate methods based on simulation of three-valued structures and their tractability properties make them suitable for their implementation in real systems. In the next chapter we provide such an implementation, based on the model generator system IDP.

4.7 Proofs for the Accuracy Section

Proof of Proposition 34. First, we show the following lemma:

Lemma 2 *If $\mathcal{Q}[\bar{x}]$ is squared in $\mathfrak{D}_\mathcal{Q}$, then $\mathcal{Q}[\bar{x}]$ is squared in $\mathfrak{D}$.*

Proof. Note that the models of $\mathfrak{D}_\mathcal{Q}$ correspond exactly to the models M' of $\mathfrak{D}$ such that $P^{M'} = P^D$, for every positive free predicate P of $\mathcal{Q}[\bar{x}]$. Also, for every model M of $\mathfrak{D}$, the structure M' obtained by setting $P^{M'} = P^D$ if P is positive free and $P^{M'} = P^M$ otherwise, is a model of $\mathfrak{D}_\mathcal{Q}$. It follows that $\mathcal{Q}[\bar{d}]^{M'} \leq \mathcal{Q}[\bar{d}]^M$, as for all predicates P that occur negatively in $\mathcal{Q}[\bar{x}]$, it holds that $P^{M'} = P^M$ while predicates P with only positive occurrences have $P^{M'} \subseteq P^M$. Thus, $\mathfrak{D} \models \mathcal{Q}[\bar{d}]$ if and only if $\mathfrak{D}_\mathcal{Q} \models \mathcal{Q}[\bar{d}]$, and so $Cert_{\mathfrak{D}_\mathcal{Q}}(\mathcal{Q}[\bar{x}]) = Cert_\mathfrak{D}(\mathcal{Q}[\bar{x}])$.

Observe, also, that $\mathcal{O}_\mathfrak{D}$ is an approximation of $\mathfrak{D}_\mathcal{Q}$ that satisfies the condition on $\mathcal{K}$ in Theorem 12. It follows that $Cert_{\mathcal{O}_\mathfrak{D}}(\mathcal{Q}[\bar{x}]) = Cert_{\mathcal{O}_{\mathfrak{D}_\mathcal{Q}}}(\mathcal{Q}[\bar{x}])$ and so, since $\mathcal{Q}[\bar{x}]$ is squared in $\mathfrak{D}_\mathcal{Q}$, $Cert_{\mathcal{O}_\mathfrak{D}}(\mathcal{Q}[\bar{x}]) = Cert_{\mathfrak{D}_\mathcal{Q}}(\mathcal{Q}[\bar{x}])$. By what we have obtained above, it follows that $Cert_{\mathcal{O}_\mathfrak{D}}(\mathcal{Q}[\bar{x}]) = Cert_\mathfrak{D}(\mathcal{Q}[\bar{x}])$, which implies that $\mathcal{Q}[\bar{x}]$ is squared in $\mathfrak{D}$.

To complete the proof of Proposition 34, let $\mathcal{Q}[\bar{x}]$ be a query that is squared in $\mathfrak{D}_\mathcal{Q}$. By Lemma 2, $\mathcal{Q}[\bar{x}]$ is squared in $\mathfrak{D}$, and so by Proposition 33 for $\mathcal{K} = \mathcal{C}_\mathfrak{D}$ the theorem is obtained. □

We now turn to Theorem 13 and Proposition 35 that identify conditions for squardeness. For this, we need the following definitions:

Definition 4.21 *A query $\mathcal{Q}[\bar{x}]$ is* literal-based *in a satisfiable theory Γ containing $DCA \wedge UNA$ if and only if*

$$Cert_\Gamma(\mathcal{Q}[\bar{x}]) = \{\bar{d} \in HU^n \mid sv_{\mathcal{O}_\Gamma}(\mathcal{Q}[\bar{d}]) = \mathbf{t}\}.$$

$\mathcal{Q}[\bar{x}]$ is Kleene-precise *in a three-valued structure $\mathcal{K}$ with domain Dom if and only if*

$$Cert_\mathcal{K}(\mathcal{Q}[\bar{x}]) = \{\bar{d} \in Dom^n \mid sv_\mathcal{K}(\mathcal{Q}[\bar{d}]) = \mathbf{t}\}.$$

Again, in the sequel, if $\mathcal{Q}[\bar{x}]$ is literal-based in a theory $\Gamma = \mathcal{M}(\mathfrak{D})$ (see Definition 3.7), we shall sometimes say that $\mathcal{Q}[\bar{x}]$ is literal-based in $\mathfrak{D}$.

Note that since a supervaluation is more precise than standard Kleene truth assignment, it follows that the certain answers of a Kleene-precise query under standard Kleene truth assignment coincide with the supervaluation.

Lemma 3 *A query $\mathcal{Q}[\bar{x}]$ is squared in Γ if and only if $\mathcal{Q}[\bar{x}]$ is literal-based in Γ and Kleene-precise in $\mathcal{O}_\Gamma$.*

Proof: For every Γ and $\mathcal{Q}[\bar{x}]$, the following inequalities hold:

$$\begin{array}{rcl} Cert_{\mathcal{O}_\Gamma}(\mathcal{Q}[\bar{x}]) & \subseteq & \{\bar{d} \in HU^n \mid sv_{\mathcal{O}_\Gamma}(\mathcal{Q}[\bar{d}]) = \mathbf{t}\} \\ & \subseteq & Cert_\Gamma(\mathcal{Q}[\bar{x}]). \end{array}$$

If $\mathcal{Q}[\bar{x}]$ is literal-based in Γ and Kleene-precise in $\mathcal{O}_\Gamma$, the inequalities turn into equalities. Vice versa, if $\mathcal{Q}[\bar{x}]$ is squared, the three terms are equal, and it follows that the query is literal-based and Kleene-precise. □

By Lemma 3, then, for showing Theorem 13 and Proposition35, it is enough to show that $\mathcal{Q}[\bar{x}]$ is literal-based in $\mathfrak{D}_\mathcal{Q}$ and is Kleene-precise in $\mathcal{O}_{\mathfrak{D}_\mathcal{Q}}$.

Proof of Theorem 13. Let $\Gamma = \mathcal{M}(\mathfrak{D}_\mathcal{Q})$. Since Γ contains $\mathsf{DCA}(\Sigma) \wedge \mathsf{UNA}(\Sigma)$, the formula $\forall x \varphi[\bar{x}]$ will be literal-based in Γ if $\varphi[\bar{x}]$ is literal-based in Γ. Hence, it suffices to prove the proposition for the case that $\mathcal{Q}[\bar{x}]$ is quantifier free.

Now, given that all predicates of non-literal conjuncts C_{ij} are two-valued in $\mathcal{O}_\Gamma$, it is obvious that $\Gamma \models C_{ij}[\bar{d}]$ if and only if $sv_{\mathcal{O}_\Gamma}(C_{ij}[\bar{d}]) = \mathbf{t}$, if and only if $C_{ij}[\bar{d}]^{\mathcal{O}_\Gamma} = \mathbf{t}$ for each $\bar{d} \in HU^n$.

Since each pair $C_i[\bar{x}], C_j[\bar{x}]$ of $\mathcal{Q}[\bar{x}]$ is mutually exclusive in $\mathcal{O}_\Gamma$, it follows that for all $\bar{d}$, there is at most one conjunct C_i such that $\Gamma \not\models \neg C_{ij}[\bar{d}]$. It is then easy to verify that

$$\begin{array}{ll} \Gamma \models C_i[\bar{d}] & \text{iff } \Gamma \models C_{ij}[\bar{d}], \text{ for each conjunct } C_{ij} \text{ of } C_i, \\ & \text{iff } sv_{\mathcal{O}_\Gamma}(C_{ij}[\bar{d}]) = \mathbf{t} \text{ for each conjunct } C_{ij} \\ & \text{iff } (C_{ij}[\bar{d}])^{\mathcal{O}_\Gamma} = \mathbf{t} \text{ for each conjunct } C_{ij} \\ & \text{iff } sv_{\mathcal{O}_\Gamma}(C_i[\bar{d}]) = \mathbf{t} \\ & \text{iff } (C_i[\bar{d}])^{\mathcal{O}_\Gamma} = \mathbf{t}. \end{array}$$

Thus $\mathcal{Q}[\bar{x}]$ is literal-based in Γ and Kleene-precise in $\mathcal{O}_\Gamma$. □

Proof outline of Proposition 35. As $\mathfrak{D}_\mathcal{Q}$ is atomical, every $\mathcal{Q}[\bar{x}]$ is literal-based in it. By Lemma 3 it remains to show that $\mathcal{Q}[\bar{x}]$ is Kleene-precise in $\mathcal{O}_{\mathfrak{D}_\mathcal{Q}}$. All we need to do is to show that if $\mathcal{Q}[\bar{d}]^{\mathcal{O}_{\mathfrak{D}_\mathcal{Q}}} = \mathbf{u}$ then $sv_{\mathcal{O}_{\mathfrak{D}_\mathcal{Q}}}(\mathcal{Q}[\bar{d}]) = \mathbf{u}$. We construct two-valued structures K and K' from $\mathcal{O}_{\mathfrak{D}_\mathcal{Q}}$ where K maps each unknown atom $P(\bar{a})$ of $\mathcal{O}_{\mathfrak{D}_\mathcal{Q}}$ to $\mathbf{t}$ if P occurs positively in φ and to $\mathbf{f}$ otherwise; likewise, K' follows the inverse strategy and maps unknown atoms $P(\bar{a})$ to $\mathbf{f}$ if P occurs positively in φ and to $\mathbf{t}$ otherwise. By a routine induction, one

shows that if $\mathcal{Q}[\bar{d}]^{\mathcal{O}_{\mathfrak{D}_\mathcal{Q}}} = \mathbf{u}$, then $\mathcal{Q}[\bar{d}]^K = \mathbf{t}$ and $\mathcal{Q}[\bar{d}]^{K'} = \mathbf{f}$, and hence, $sv_{\mathcal{O}_{\mathfrak{D}_\mathcal{Q}}}(\mathcal{Q}[\bar{d}]) = \mathbf{u}$. □

Proof of Proposition 36. By the soundness of $\mathcal{C}_\mathfrak{D}$, it holds for every $\bar{a} \in Dom^n$, that if $P(\bar{a})^{\mathcal{C}_\mathfrak{D}} \neq \mathbf{u}$, then $P(\bar{a})^{\mathcal{C}_\mathfrak{D}} = P(\bar{a})^{\mathcal{O}_\mathfrak{D}}$. So, let us assume that $P(\bar{a})^{\mathcal{C}_\mathfrak{D}} = \mathbf{u}$. Observe that, in this case, $P(\bar{a}) \notin D$.

To show that $P(\bar{a})^{\mathcal{O}_\mathfrak{D}} = \mathbf{u}$, we need to construct two models M, M' of $\mathfrak{D}$ such that $M \models P(\bar{a})$ and $M' \models \neg P(\bar{a})$. Since $P(\bar{a}) \notin D$, we can take $M' = D$ which is indeed a model of $\mathfrak{D}$. Let us now construct M. By construction of $\mathcal{C}_\mathfrak{D}$, it holds that $P(\bar{a}) \notin D$ and $\Psi_P[\bar{a}]^{\mathcal{C}_\mathfrak{D}} \neq \mathbf{t}$. The sentence $\Psi_P[\bar{a}]$ is squared in $\mathfrak{D}$ and by assumption it holds that $Q^{\mathcal{C}_\mathfrak{D}} = Q^{\mathcal{O}_\mathfrak{D}}$ for each negatively occurring predicate Q in this formula. Therefore, the conditions of Theorem 12 are satisfied, so there is a model N of $\mathfrak{D}$ such that $N \models \neg\Psi_P[\bar{a}]$. If $N \models \neg P(\bar{a})$, then we can take $M = N$. Otherwise, we transform N into a model M in which $P(\bar{a})$ is false.

Consider the set $S_P = \{Q \in \mathcal{R}(\Sigma) |$ for some $Q' \preceq^- Q$, $P \in^+ \Psi_{Q'}\}$. It consists of all predicates in whose window of expertise P has a positive occurrence, and all predicates that negatively depend on these. By the condition of the proposition, it holds that $P \notin S_P$.

Define M as the structure obtained by modifying N as follows:

- $P^M = P^N \cup \{\bar{a}\}$, i.e. $P(\bar{a})$ is made true;
- $Q^M = Q^D$ for $Q \in S_P$.

This modification increases P and decreases all predicates of S_P; i.e. $P^N \leq P^M$, $Q^M \leq Q^N$ for $Q \in S_P$, and $Q^N = Q^M$ otherwise. Thus, formulas with only positive occurrences of P and only negative occurrences of predicates $Q \in S_P$ have a larger truth value in M than in N.

To verify that M is a model of $\mathfrak{D}$, it suffices to check that M satisfies all local closed-world assumptions. Consider any instance of a local closed-world assumption:

$$\varphi \equiv \neg\Psi_Q[\bar{d}] \vee \neg Q(\bar{d}) \vee (Q(\bar{d}) \in D)$$

Each of these formulas is satisfied in N. Let us verify that it is satisfied in M as well. There are four cases:

- $Q = P$ and $\bar{d} = \bar{a}$: in this case, $N \models \neg\Psi_P[\bar{a}]$. The formula $\neg\Psi_P[\bar{a}]$ contains only positive occurrences of P and only negative occurrences of predicates $Q \in S_P$, hence $\mathbf{t} = (\neg\Psi_P[\bar{a}])^N \leq (\neg\Psi_P[\bar{a}])^M$.
- $Q = P$ and $\bar{d} \neq \bar{a}$: we have $(\neg P(\bar{d}) \vee (P(\bar{d}) \in D))^N = (\neg P(\bar{d}) \vee (P(\bar{d}) \in D))^M$ and $\neg\Psi_P[\bar{a}]$ contains only positive occurrences of P and only negative occurrences of predicates $Q \in S_P$. It follows that $\mathbf{t} = \varphi^N \leq \varphi^M$.
- $Q \in S_P$: M satisfies $\neg Q(\bar{d}) \vee (Q(\bar{d}) \in D)$.

- $Q \notin S_P$ and $Q \neq P$: φ contains only positive occurrences of P and only negative occurrences of predicates of S_Q; hence $\mathbf{t} = \varphi^N \leq \varphi^M$. □

Chapter 5

Modal Queries and a Prototype for the LCWA

5.1 Introduction

In the previous chapter, we have proposed tractable methods to answer queries in locally complete databases. Answers were divided in two categories: certain and possible. Certain query answers are those tuples that are definitely true according to the knowledge of the database instance together with the local closed-world assumptions. Possible answers correspond to the dual, i.e. answers that are possibly true or, equivalently, that are certainly not false. Queries to extract these two types of answers provided a rich framework to retrieve information from locally complete databases. There are situations, however, in which a combination of both, queries for certain and possible answers in a single expression is needed in order to retrieve the desired answers. The following example illustrates one such a situation.

Example 5.1 *A patient (Veronique Claes) visits her personal doctor complaining about severe headaches during the night. During the visit, the doctor finds out that the patient is being treated for two unrelated pathologies: an herniated lumbar disc and angina. The practitioner suspects that one of the drugs that has been prescribed to treat these conditions might be the cause of the migraine.*

To investigate the source of the headaches, the doctor counts upon two sources of information. The first one is his personal database storing the past records of all the drugs he has prescribed to patients. For Veronique Claes, this information is displayed in Table 5.1. Prescription predicate indicates the name of a patient, the drug that has been prescribed to treat certain conditions and the date of the prescription.

The second source of information is a public locally closed pharmacological repository that stores information about certain drugs and their side effects.

Prescription			
Patient	*Drug*	*Condition*	*DatePres.*
Veronique Claes	CH563	Herniated disc	10.10.08
Veronique Claes	MG675	Herniated disc	10.10.08
Veronique Claes	LD465	Herniated disc	17.10.08
Veronique Claes	LH194	Angina	12.06.08
Veronique Claes	LH244	Angina	12.06.08
Veronique Claes	HP354	Angina	14.06.08

Table 5.1: A database of a patient and prescribed medicines.

The database instance of Drug Class (Table 5.2) stores the classification of certain drugs according to their composition (or chemical type). The set of local closed-world assumptions for $SideEffect$ summarizes the results of a pharmacological research about side effects of classes of drugs. The results are shown below. The first expression ($\mathcal{LCWA}_1$), for instance, expresses that it is known for steroids whether they induce migraine as a side effect. The database instance for $SideEffect$ is empty.

Drug Class	
Drug	*Class*
CH563	Steroid
MG675	Analgesic
LD465	Steroid
LH194	Steroid
LH244	Antipyretic
HP354	Analgesic

Table 5.2: Database of drugs and their classification.

$$\mathcal{LCWA}_1(SideEffect(x,y), DrugClass(x, Steroid) \wedge y = Migraine)$$
$$\mathcal{LCWA}_2(SideEffect(x,y), DrugClass(x, Analgesic) \wedge y = Migraine)$$
$$\mathcal{LCWA}_3(SideEffect(x,y), DrugClass(x, Antipyretic) \wedge y = Diarrhea)$$
$$\mathcal{LCWA}_4(SideEffect(x,y), DrugClass(x, Antipyretic) \wedge y = Arrhythmia)$$

To find possible causes of the migraine in this patient, the practitioner needs a query as follows:

$$\mathcal{Q}[\bar{x}] : \mathbf{C}Prescription(Veronique\ Claes, x, y, z) \wedge \mathbf{P}SideEffect(x, Migraine)$$

The first conjunct of $\mathcal{Q}[\bar{x}]$ asks for drugs x that are certainly taken by Veronique Claes. The second conjunct asks for drugs that possibly cause migraine. The whole query then asks for drugs taken by Veronique Claes that may be the cause of the migraine.

As answers to this query we would expect the following set:

$$Ans(\mathcal{Q}[\bar{x}]) = \{(LH244, Angina, 12.06.08)\}.$$

Based on this information, the doctor may choose to discontinue the prescription of $LH244$.

Query $\mathcal{Q}[\bar{x}]$ above is modal: predicates $Prescription$ and $SideEffect$ are within the scope of modalities **C** and **P**, that stand for certainly and possibly true, respectively. These types of queries cannot be expressed in terms of a single query for retrieving either possible or certain answers.

The limitation of first-order logic for expressing modalities has already been identified in the context of incomplete databases, and it has motivated the introduction of more expressive languages that could allow such a combination of modalities. In this section, we describe one such a formalism in the context of the local closed-world assumption. The language we choose is Levesque's modal formalism $KFOPCE$ (First-Order Predicate Calculus with Equality and modal operator **K**) (Levesque 1982). $KFOPCE$ is a variant with parameters[1] of the well-known modal logic S45 (Chellas 1980). The semantics of $KFOPCE$ is defined in terms of possible worlds.

In the setting of locally complete databases, we want to apply the logic of $KFOPCE$ as follows. Given a database $\mathfrak{D} = (D, \mathcal{L})$, the meaning of $\mathfrak{D}$ represents the set of possible worlds according to D and $\mathcal{L}$, and the real world is one of them. $KFOPCE$ queries are then posed with respect to this set, from which the answers are extracted. In this scenario, queries for computing certain or possible answers can be viewed as special cases of modal queries. It will not come as a surprise then that queries for exclusively certain or possible answers are easily expressed by $KFOPCE$ formulas.

A well-known consequence of increasing language expressiveness is that the reasoning tasks become computationally more complex. As we showed in Chapter 3, reasoning in locally complete databases was already an expensive task (confer Proposition 17). The extension of the language with modal operators makes query answering at least as expensive as in the first-order case.

To reduce the complexity of reasoning with locally complete databases, we presented, in Chapter 4, several methods to approximate answers using three-valued structures. In this chapter, we follow the same approach but we now apply it to queries in the language $KFOPCE$. Towards this goal we adapt some of the approximation techniques developed in the previous chapter to the new, more expressive modal setting. As a result, we obtain answers in

[1] Pairwise distinct symbols that define the domain over which quantifiers range.

a tractable way for a much richer query language. To assess the soundness of answers under this approach, we provide classes of queries for which our method always produces correct answers.

In the second part of this chapter we present a simple prototype to answer modal queries from locally complete databases. The system is based on the approximative techniques developed in this and the previous chapter. The reasoning task is supported by the model expansion system IDP (Mariën et al. 2006).

This chapter is organized as follows. In Section 5.2, we introduce the basic concepts of Levesque's logic of $KFOPCE$. In Section 5.2.2, we adapt this logic to the three-valued setting and define query answering of modal expressions. In Section 5.2.3, we apply the modal framework to locally complete databases and show how to answer queries with it. In Section 5.3, we present the prototype. In Sections 5.4 and 5.5, we conclude this chapter with a summary of the results and a brief bibliographical note.

5.2 A Modal Language for the LCWA

We are interested in a query language that could allow us to express queries as the one stated in Example 5.1. There, the query is *modal* in the sense that it contains modalities **C** and **P** of database predicates $Prescription$ and $SideEffect$. In colloquial terms, modality **C** refers to what is *certainly true* about a formula ϕ. Its dual modality, **P**, refers to what is *possibly true* about ϕ. **C** and **P** are related in the following standard way:

$$\mathbf{P}\phi \equiv \neg\mathbf{C}\neg\phi.$$

In words, knowing that ϕ is possibly true is equivalent to not knowing that the negation of ϕ is certainly true. In the following section we study a logic that captures these intuitions.

5.2.1 The Logic of $KFOPCE$

In this section we review Levesque's $KFOPCE$ logic (Levesque 1982; Reiter 1992) and we argue that this formalism is a suitable language for querying locally complete databases.

$KFOPCE$ is a first-order modal language with equality and the modal operator **K**, which stands for "known" or "believed". A distinctive feature of $KFOPCE$ is the assumption of a fixed domain of so called parameters. The parameters represent a single universal domain of discourse[2]. $FOPCE$ is the language $KFOPCE$ without the modal operator **K**.

[2]In the context of the local closed-world assumption, the set of parameters corresponds the Herbrand universe of the database.

A knowledge base (or database) is specified as a set of $FOPCE$ (First-Order Predicate Calculus with Equality) sentences. An atomic sentence is an atom of the form $P(p_1, \ldots, p_n)$, where P is an $n-$ary predicate symbol and $p_1, \ldots, p_n$ are parameters. A world is any set of atomic sentences that includes $p = p$ for each parameter p and that does not include $p_1 = p_2$ for different parameters p_1 and p_2[3]. The semantics of $KFOPCE$ is defined in terms of possible worlds. If we characterize a knowledge base by a set of worlds $\mathfrak{W}$, we define that $\mathbf{K}\phi$ is true in the database represented by $\mathfrak{W}$ if and only if ϕ is true in every element of $\mathfrak{W}$.

Below, this intuition is formalized.

Definition 5.1 *The truth value of a sentence of $KFOPCE$ in a world W and a set of worlds $\mathfrak{W}$ is defined as follows:*

1. *If ϕ is an atom, ϕ is true in W and $\mathfrak{W}$ if and only if $\phi \in W$.*
2. *$\neg\phi$ is true in W and $\mathfrak{W}$ if and only if ϕ is not true in W and $\mathfrak{W}$.*
3. *$\phi_1 \wedge \phi_2$ is true in W and $\mathfrak{W}$ if and only if ϕ_1 and ϕ_2 are both true in W and $\mathfrak{W}$.*
4. *$\forall x(\phi[\bar{x}])$ is true in W and $\mathfrak{W}$ if and only if for every set of parameters $\overline{p}$, $\phi[\overline{p}]$ is true in W and $\mathfrak{W}$.*
5. *$\exists x(\phi[\bar{x}])$ is true in W and $\mathfrak{W}$ if and only if there is a parameter $\overline{p}$, such that $\phi[\overline{p}]$ is true in W and $\mathfrak{W}$.*
6. *$\mathbf{K}\phi$ is true in W and $\mathfrak{W}$ if and only if for every $S \in \mathfrak{W}$, ϕ is true in S and $\mathfrak{W}$.*

We denote by $(\mathfrak{W}, W) \models \phi$ when ϕ is true in $\mathfrak{W}$ and W. If ϕ is modal, i.e. every subformula in ϕ is under the scope of $\mathbf{K}$, then W does not matter and we write $\mathfrak{W} \models \phi$. Conversely, if $\mathbf{K}$ does not occur in ϕ, i.e. ϕ is an objective *formula, then the set of worlds does not matter and we write $W \models \phi$.*

With the exception of the special treatment of parameters, the above semantics is identical to that of the modal logic $KD45$ or weak $S5$ (see e.g. (Chellas 1980)).

When W is a world and $\mathfrak{W}$ is a set of worlds, the pair $(\mathfrak{W}, W)$ is a *model* of a set of $KFOPCE$ sentences if and only if each sentence is true in W and $\mathfrak{W}$.

Example 5.2 *Consider the following incomplete $FOPCE$ knowledge base consisting of two ground atoms:*

$$\Gamma = \{Dept(Bart\ Delvaux, CS),\\ Dept(David\ Finner, Bio)\}$$

[3]This is equivalent with the notion of structure we have used so far.

The following corresponds to some $KFOPCE$ queries posed to this knowledge base:

- $\mathcal{Q}_1$*:* $\mathbf{K}Dept(Bart\ Delvaux, CS)$*. Does* Γ *know that Bart Delvaux belongs to CS?*
- $\mathcal{Q}_2 : \mathbf{K}\neg Dept(David\ Finner, CS)$*: Does* Γ *know that David Finner does not belong to CS?*
- $\mathcal{Q}_3 : \mathbf{K}\exists x Dept(x, Eco)$*: Does* Γ *know whether there is someone belonging to the economy department?*
- $\mathcal{Q}_4 : \neg\mathbf{K}\neg Dept(David\ Finner, CS)$*: According to what* Γ *knows, is it possible that David Finner is a collaborator of the CS department?*
- $\mathcal{Q}_5 : \neg\mathbf{K}Dept(TomDemans, Bio)$ *According to what* Γ *knows, is it possible that Tom Demans is not a member of the biology department?*

Consider now the following worlds and sets of worlds based on the knowledge base vocabulary Σ*.*

$$W_1 = \{Dept(Bart\ Delvaux, CS)\},$$
$$W_2 = \{Dept(David\ Finner, Bio)\},$$
$$W_3 = \{Dept(Bart\ Delvaux, CS), Dept(David\ Finner, Bio)\},$$

$\mathfrak{W}_1 = \{W_2\}$ *and* $\mathfrak{W}_2 = \{W_3\}$*. The following corresponds to the truth evaluation of some formulas in worlds* W_1 *and* W_2 *and sets of worlds* $\mathfrak{W}_1, \mathfrak{W}_2$*:*

- $W_1 \models Dept(BartDelvaux, CS)$? *Answer: Yes.*
- $W_2 \models \neg CarO(BartDelvaux, CS)$? *Answer: Yes.*
- $\mathfrak{W}_1 \models \mathbf{K}Dept(Lien\ Desmet, Bio)$? *Answer: No.*
- $\mathfrak{W}_2 \models \neg\mathbf{K}\neg Dept(Lien\ Desmet, Bio)$? *Answer: Yes.*

We now proceed to define answers to $KFOPCE$ queries on a $FOPCE$ knowledge base:

Definition 5.2 (Answer to a query) *Consider a first-order vocabulary* Σ*. Let*

- Γ *be a set of* $FOPCE$ *sentences,*
- $\mathcal{Q}[\bar{x}]$ *a* $KFOPCE$ *formula with free variables* $\bar{x}$*,*
- $\mathfrak{W}(\Gamma)$ *be the set of Herbrand models of* Γ*,*

all based on Σ. *A tuple* $\bar{t}$ *of parameters is an* answer *to* $\mathcal{Q}$ *if and only if for all* $W \in \mathfrak{W}(\Gamma)$, $\mathcal{Q}[\bar{t}]$ *is true in* W *and* $\mathfrak{W}(\Gamma)$. *We denote by* $Ans_\Gamma \mathcal{Q}[\bar{x}]$ *the set of answers of* $\mathcal{Q}[\bar{x}]$ *with respect to* Γ.

Example 5.3 *The following expressions formalize some of the queries of Example 5.2 and their answers according to the semantics of* $KFOPCE$.

- $\mathcal{Q}_1[\bar{x}] : \mathbf{K}\neg Dept(David\ Finner, x)$. $Ans_\Gamma \mathcal{Q}_1[x] = \emptyset$. *Indeed, since no constraint is imposed on the number of departments a person may be affiliated to, it is possible, according to* Γ, *that David Finner belongs to any department. Thus, there is no element* x *for which* $Dept(David\ Finner, x)$ *is certainly false.*
- $\mathcal{Q}_2[\bar{x}] : \mathbf{K}\exists x Dept(x, Eco)$. $Ans_\Gamma \mathcal{Q}_2[x] =$ No. *Only the atoms* $Dept(Bart\ Delvaux, CS)$ *and* $Dept(David\ Finner, Bio)$ *must be true in every model of the knowledge base* Γ. *Hence, according to* Γ, *it is not certainly true that there are members of the economy department.*

Queries for certain and possible answers are related to $KFOPCE$ query answers as follows:

Proposition 38 *Let* $\mathfrak{D} = (D, \mathcal{L})$ *be a locally complete database based on vocabulary* Σ *and* $\mathcal{Q}[\bar{x}]$ *a query posed to it. It holds that:*

$$\begin{aligned} Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) &= \{\bar{t} \mid \mathfrak{D} \models Q[\bar{t}]\} \\ &= \{\bar{t} \mid mod(\mathfrak{D}) \models \mathbf{K}Q[\bar{t}]\} \\ Poss_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) &= \{\bar{t} \mid \mathfrak{D} \not\models \neg Q[\bar{t}]\} \\ &= \{\bar{t} \mid mod(\mathfrak{D}) \models \neg\mathbf{K}\neg Q[\bar{t}]\}, \end{aligned}$$

where $mod(\mathfrak{D})$ *denotes all the models of* $\mathfrak{D}$.

A straight forward application of the last proposition is the simulation of queries for certain or possible answers using modal $KFOPCE$ expressions. The formula $\mathbf{K}\varphi[\bar{x}]$ is equivalent to $\text{Cert}_{\mathfrak{D}}(\varphi[\bar{x}])$ for obtaining certain answers. $\neg\mathbf{K}\neg\varphi[\bar{x}]$ is equivalent to $\text{Poss}_{\mathfrak{D}}(\varphi[\bar{x}])$ for obtaining possible answers.

In the context of locally complete databases, we will only consider queries in which each predicate symbol is within the scope of (only) one modal operator. In other words, queries without nested modalities. For instance, queries such as $\mathbf{K}\neg\mathbf{K}\phi$ will not be considered.

$KFOPCE$ is strictly more expressive than $FOPCE$ (Reiter 1992). When the set of parameters is finite, $KFOPCE$ is decidable for query answering,[4] but not tractable (Rosati 2000). As in the case of query answering in general locally complete databases[5], this fact motivates the use of three-valued approximations of $FOPCE$ knowledge bases, in particular of locally complete databases theories, to answer $KFOPCE$ queries in tractable ways.

[4] There exists an algorithm that determines in finite time whether a tuple $\bar{t}$ of parameters is an answer to a query or not.

[5] As in Proposition 17.

5.2.2 $KFOPCE$ and Three-Valued Logic

A *formula is modal* if every predicate symbol is within the scope of a modal operator. We can now extend the Kleene truth evaluation defined in Section 4.2.1 to $KFOPCE$ modal formulas:

- $(\mathbf{K}\varphi)^{\mathcal{K}} = \mathbf{t}$ if $\varphi^{\mathcal{K}} = \mathbf{t}$; else $(\mathbf{K}\varphi)^{\mathcal{K}} = \mathbf{f}$.

Truth evaluation of modal formulas in all cases returns either true or false:

Proposition 39 *If φ is a modal formula (without free variables), then $\varphi^{\mathcal{K}}$ is two-valued.*

Example 5.4 *Consider some of the three-valued structures of Example 4.2. Recall $\mathcal{R}(\Sigma) = \{P, Q\}$, $\mathcal{C}(\Sigma) = \{a\}$, $Dom(\mathcal{K}_i) = \{a\}$, $a^{\mathcal{K}_i} = a$, $i = 1, \ldots, 9$, and*

$$\mathcal{K}_1 = \{P^{\mathcal{K}_1}(a) = \mathbf{u},\ Q^{\mathcal{K}_1}(a) = \mathbf{u}\} \qquad \mathcal{K}_4 = \{P^{\mathcal{K}_4}(a) = \mathbf{u},\ Q^{\mathcal{K}_4}(a) = \mathbf{t}\}$$
$$\mathcal{K}_5 = \{P^{\mathcal{K}_5}(a) = \mathbf{f},\ Q^{\mathcal{K}_5}(a) = \mathbf{u}\} \qquad \mathcal{K}_9 = \{P^{\mathcal{K}_9}(a) = \mathbf{f},\ Q^{\mathcal{K}_9}(a) = \mathbf{t}\}$$

The following corresponds to the evaluation of some $KFOPCE$ formulas from the indicated three-valued structures:

$$(\mathbf{K}(P(a) \wedge Q(a)))^{\mathcal{K}_1} = \mathbf{f} \qquad (\mathbf{K}(Q(a) \vee \neg Q(a)))^{\mathcal{K}_5} = \mathbf{f}$$
$$(\neg\mathbf{K}\neg Q(a))^{\mathcal{K}_4} = \mathbf{t} \qquad (\neg\mathbf{K}\neg(\exists x P(x)))^{\mathcal{K}_9} = \mathbf{f}$$

Similar as in the case of the standard Kleene evaluation, expression $(\mathbf{K}(Q(a) \vee \neg Q(a)))^{\mathcal{K}_1}$ evaluates to false, even though $Q(a) \vee \neg Q(a)$ is a valid formula. This is expected, since $KFOPCE$ three-valued evaluation relies on Kleene truth assignment which, as we discussed in the previous chapter, is weaker than first-order semantics.

Definition 5.3 *Given a three-valued Σ-interpretation $\mathcal{K}$ and a $KFOPCE$ query $\mathcal{Q}[\bar{x}]$ in Σ, we define answers for $\mathcal{Q}[\bar{x}]$ in $\mathcal{K}$ as follows:*

- *$\bar{t}$ is an answer in $\mathcal{K}$ for $\mathcal{Q}[\bar{x}]$, iff $\mathcal{Q}[\bar{t}]^{\mathcal{K}} = \mathbf{t}$.*

We denote by $Ans^3_{\mathcal{K}}(\mathcal{Q}[\bar{x}])$ the set of answers of $\mathcal{Q}[\bar{x}]$ in $\mathcal{K}$.

We are after using three-valued approximations of a $FOPCE$ theory Γ to answer *efficiently* $KFOPCE$ queries posed to Γ. Next we analyze this scenario for the case of locally complete databases.

5.2.3 $KFOPCE$ and Locally Complete Databases

Given a locally closed database $\mathfrak{D}$ and a $KFOPCE$ query Q, we follow the same principle introduced in the previous chapter to answer Q with respect to $\mathfrak{D}$: we first construct a three-valued approximation $\mathcal{C}_{\mathfrak{D}}$ of $\mathfrak{D}$ using $\mathcal{A}pp_{\mathfrak{D}}$, and then we pose the query Q against $\mathcal{C}_{\mathfrak{D}}$. As in Theorem 7, we expect that the answers so obtained are sound. The following theorem shows that, under certain conditions, that is the case.

Theorem 15 *Let $\mathfrak{D}$ be a locally complete database based on vocabulary Σ and $\mathcal{Q}[\bar{x}]$ a KFOPCE query. Let $\mathcal{C}_{\mathfrak{D}}$ be the approximation of $\mathfrak{D}$ according to Definition 4.7. If for all subformulas $\mathbf{K}\psi$ that occur negatively in $\mathcal{Q}[\bar{x}]$ it is the case that $Cert_{\mathcal{C}_{\mathfrak{D}}}(\psi[\bar{x}]) = Cert_{\mathfrak{D}}(\psi[\bar{x}])$, then*

$$Ans^3_{\mathcal{C}_{\mathfrak{D}}}(\mathcal{Q}[\bar{x}]) \subseteq Ans_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]).$$

Proof: Let's start by recalling Corollary 3, which states that $Cert_{\mathcal{C}_{\mathfrak{D}}}(\mathcal{Q}[\bar{x}]) \subseteq \mathrm{Cert}_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) \subseteq \mathrm{Poss}_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) \subseteq Poss_{\mathcal{C}_{\mathfrak{D}}}(\mathcal{Q}[\bar{x}])$. It follows from this corollary together with Proposition 38 that sub-queries ψ under the scope of positive occurrences of $\mathbf{K}$ are under-approximated by $\mathrm{Cert}_{\mathfrak{D}}(\psi[\bar{x}])$. To complete the proof we have to show additionally that negative occurrences of modalities are also under-approximated. For this, we use the hypothesis. Since by the hypothesis, for sub-queries ψ of $\mathcal{Q}[\bar{x}]$ within the scope of a negative modality it is the case that $Poss_{\mathcal{C}_{\mathfrak{D}}}(\neg\psi[\bar{x}]) = Poss_{\mathfrak{D}}(\neg\psi[\bar{x}])$, it follows that possible answers are under-approximated as well. Thus, as subformulas under positive modalities are under stimated and negative ones are exact, by a monotonicity of first-order logic it then follows that $Ans^3_{\mathcal{C}_{\mathfrak{D}}}(\mathcal{Q}[\bar{x}]) \subseteq Ans_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$. □

This theorem states that sound answers are guaranteed whenever the query contains only positive occurrences of $\mathbf{K}$. To assert soundness in cases where the query contains also sub-queries $\psi[\bar{x}]$ under the scope of negative occurrences of $\mathbf{K}$, we must check whether $Cert_{\mathcal{C}_{\mathfrak{D}}}(\psi[\bar{x}]) = Cert_{\mathfrak{D}}(\psi[\bar{x}])$ holds. Depending on the structure of $\psi[\bar{x}]$, this can be done by applying Propositions 36 and 37. A direct application of these results yields that whenever there is closed-world information in the sub-query ψ, the condition $Cert_{\mathcal{C}_{\mathfrak{D}}}(\psi[\bar{x}]) = Cert_{\mathfrak{D}}(\psi[\bar{x}])$ is satisfied and the method returns correct answers.

Another way to guarantee soundness of answers, is by replacing in the query subformulas with negative occurrences of $\mathbf{K}$ by $\mathbf{t}$.

Corollary 5 *Let $\mathfrak{D}$ be a locally complete database based on vocabulary Σ, and assume $\mathcal{Q}[\bar{x}]$ is a KFOPCE query. After replacing all subformulas $\mathbf{K}\psi$ which occur negatively in $\mathcal{Q}[\bar{x}]$ by $\mathbf{t}$, it holds that*

$$Ans^3_{\mathcal{C}_{\mathfrak{D}}}(\mathcal{Q}[\bar{x}]) \subseteq Ans_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]).$$

Proof: Indeed, if negative occurrences of a modal operator are replaced by $\mathbf{t}$, then the negation turns these occurrences into false in $\mathcal{Q}[\bar{x}]$. Since the positive occurrences of $\mathbf{K}$ in $\mathcal{Q}[\bar{x}]$ are under-approximated and all the others are false, the resulting query is under-approximated as well. □

We can now apply the approximation techniques of the previous chapter that allow us to compute answers to $KFOPCE$ queries while leaving implicit in a fixpoint query computed approximation.

Definition 5.4 ($\Delta^t_{\mathcal{Q},\mathcal{L}}$) *Let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database over a vocabulary Σ, and let $\mathcal{Q}[\bar{x}]$ be a KFOPCE query. We extend Σ with symbols*

$P_i^c, P_i^{c\neg}$ for every predicate $P_i \in \Sigma$. In addition to the predicate variables P_i^c and $P_i^{c\neg}$, we introduce a variable Q whose arity is the number of free variables of $\mathcal{Q}[\bar{x}]$, and define $\Delta^t_{\mathcal{Q},\mathcal{L}}$, as the following collection:

- *A rule for the query:*

  $$Q(\bar{x}) \leftarrow \mathcal{Q}[\bar{x}]^t,$$

  where $\mathcal{Q}[\bar{x}]^t$ is the expression resulting from replacing every subformula of the form $\mathbf{K}\varphi$ by φ^c.

- *two rules for each database predicate P_i:*

  $$P_i^c(\bar{x}_i) \leftarrow P_i(\bar{x}_i)$$

  $$P_i^{c\neg}(\bar{x}_i) \leftarrow \neg P_i(\bar{x}_i) \wedge (\Psi_{P_i}[\bar{x}_i])^c \quad \textit{(where } \Psi_{P_i} \textit{ is the window of expertise of } P_i\textit{).}$$

Define the answer query for $\mathcal{Q}[\bar{x}]$ as $[lfp_{Q,\Delta^t_{\mathcal{Q},\mathcal{L}}}](\bar{x})$.

Example 5.5 *Consider again the locally complete database of Example 5.2 and the query $\mathcal{Q}$ posed to it. To get the answers for $\mathcal{Q}$, we first compute the fixpoint query $\Delta^t_{\mathcal{Q},\mathcal{L}}$ according to Definition 5.4. The resulting query is shown below.*

$$\mathcal{Q}[\bar{x}] : \mathbf{C}Prescription(Veronique\ Claes, x, y, z) \wedge \mathbf{P}SideEffect(x, Migraine)$$

$$\Delta_{\mathcal{Q},\mathcal{L}} = \left\{ \begin{array}{l} Q(x) \leftarrow Prescription^c(Veronique\ Claes, x, y, z) \wedge \\ \neg SideEffect^{c\neg}(x, Migraine). \\ \\ Prescription^c(x, y, u, v) \leftarrow Prescription(x, y, u, v). \\ Prescription^{c\neg}(x, y, u, v) \leftarrow \neg Prescrition(x, y, u, v) \wedge \\ x = Veronique\ Claes. \\ \\ SideEffect^c(x, y) \leftarrow SideEffect(x, y). \\ SideEffect^{c\neg}(x, y) \leftarrow \neg SideEffect(x, y) \wedge \Psi. \end{array} \right\}$$

Where

$$\begin{aligned} \Psi \equiv\ & (DrugClass(x, Steroid) \wedge y = Migraine) \vee \\ & DrugClass(x, Analgesic) \wedge y = Migraine) \\ & DrugClass(x, Antipyretic) \wedge y = Diarrhea) \\ & DrugClass(x, Antipyretic) \wedge y = Arrhythmia) \end{aligned}$$

By first unfolding the query using the of $\Delta_{\mathcal{Q},\mathcal{L}}$ and then by posing it against the database instance D, we obtain:

$$Ans(\mathcal{Q}[\bar{x}]) = \{(LH244, Angina, 12.06.08)\}.$$

In the following theorem we state soundness of answers obtained with the above described fixpoint transformation. $[lfp_{Q,\Delta^t_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ with respect to $\mathcal{C}_{\mathfrak{D}}$.

Theorem 16 *Given a locally closed database $\mathfrak{D} = (D, \mathcal{L})$ over Σ, a $KFOPCE$ query $\mathcal{Q}[\bar{x}]$ and the relations $(\mathcal{R}^t_{\mathcal{Q}}, \mathcal{R}^c_1, \mathcal{R}^{\neg c}_1, \ldots, \mathcal{R}^c_n, \mathcal{R}^{\neg c}_n)$ defined by $\Delta^t_{\mathcal{Q},\mathcal{L}}$ in D. The following equalities hold:*

(a) $\mathcal{R}^c_i = \{\bar{d} \mid P_i(\bar{d})^{\mathcal{C}_{\mathfrak{D}}} = \mathbf{t}\}$, *for each* $1 \leq i \leq n$,

(b) $\mathcal{R}^{c\neg}_i = \{\bar{d} \mid P_i(\bar{d})^{\mathcal{C}_{\mathfrak{D}}} = \mathbf{f}\}$, *for each* $1 \leq i \leq n$, *and*

(c) $\mathcal{R}^t_{\mathcal{Q}} = \{\bar{d} \mid \mathcal{Q}[\bar{d}]^{\mathcal{C}_{\mathfrak{D}}} = \mathbf{t}\}$.

Proof: Cases (a) and (b) are proved in Theorem 8. To prove (c) we use a similar argument as in Theorem 8, by comparison of the fixpoint computations of the operator $\mathcal{A}pp_{\mathfrak{D}}$ that constructs $\mathcal{C}_{\mathfrak{D}}$ and the operator $\Gamma_{\Delta^t_{\mathcal{Q},\mathcal{L}}}$. Recall the operator $\mathcal{A}pp_{\mathfrak{D}}$ constructs a sequence of three-valued Σ-structures $\mathcal{K}_\alpha$, and $\Gamma_{\Delta^t_{\mathcal{Q},\mathcal{L}}}$ a sequence of tuples of relations, or equivalently, a sequence of Σ'-structures I_α where Σ' contains the new predicates P^c_i and $P^{c\neg}_i$. When $\mathcal{C}_{\mathfrak{D}}$ reaches a fixpoint, say at α (i.e. $\mathcal{K}_\alpha = \mathcal{C}_{\mathfrak{D}}$), then $\Gamma_{\Delta^t_{\mathcal{Q},\mathcal{L}}}$ reaches a fixpoint on all predicates P^c_i and $P^{c\neg}_i$ at α, and the fixpoint of $\Gamma_{\Delta^t_{\mathcal{Q},\mathcal{L}}}$ simulates $\mathcal{C}_{\mathfrak{D}}$. After one more iteration, $\Gamma_{\Delta^t_{\mathcal{Q},\mathcal{L}}}$ reaches a fixpoint also on the predicates Q . It holds that $\bar{d} \in \mathcal{R}^t_{\mathcal{Q}}$ if and only if $(\mathcal{Q}[\bar{d}]^c)^{I_\alpha} = \mathbf{t}$ if and only if $(\mathcal{Q}[\bar{d}])^{\mathcal{K}_\alpha} = \mathbf{t}$ (by Proposition 21). □

Recalling the argument presented after Theorem 8, the bottom up computation of $[lfp_{Q,\Delta^t_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ is equivalent as first computing $\mathcal{C}_{\mathfrak{D}}$ and then evaluating $\mathcal{Q}[\bar{x}]$. The latter can be done in polynomial time in the size of the database (see Proposition 27). As a consequence, the computation of answers for $[lfp_{Q,\Delta^t_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ can also be carried out in polynomial time (in the size of the database domain).

5.2.4 Discussion on $KFOPCE$ Queries

The need for combining queries for retrieving certain and possible answers in a single expression has motivated the extension of the initial first-order query language to the more expressive $KFOPCE$ modal formalism. To fence against intractability of query answering under $KFOPCE$, we have adapted the approximation techniques developed for first-order queries to the $KFOPCE$ setting. In the first-order case, the approximation approach was based on the (implicit) construction of a three-valued structure approximating all models of a database $\mathfrak{D}$. Queries were then posed against the approximation structure. To apply the approximation framework to $KFOPCE$ queries, we have defined the semantics of both $KFOPCE$ formulas and query answers under the three-valued setting. This was done in Section 5.2.2 and Definition 5.3, respectively. With the semantic machinery in order, we proceeded to show

that answers obtained by using approximations were, under certain conditions, correct (Proposition 15). Since the explicit computation of a three-valued structure is in general impractical (recall the discussion opening Section 4.3.1), we showed with a working example how to apply the query answering techniques based on fixpoint logic of Section 4.3.1 to the $KFOPCE$ setting. Under this approach answers to $KFOPCE$ queries can be retrieved in a tractable way while leaving the three-valued structure implicit in the fixpoint query.

In contrast to the first-order case, where sound answers were unconditionally guaranteed using the approximations, in the case of $KFOPCE$ queries the situation is slightly more complicated: soundness depends on the structure of the query. Intuitively, this is due to the fact that in $KFOPCE$ sub-queries for certain answers are under approximated while sub-queries for possible answers are over approximated. The composed $KFOPCE$ query may then be under or over approximated. Fortunately, the results obtained in Section 4.4 can be applied to this scenario for checking conditions in which the approximation framework delivers sound answers.

To assess completeness of query answering under $KFOPCE$, for queries that contain only positive or only negative occurrences of modal predicates, the accuracy results of Section 4.4 apply. Completeness of general $KFOPCE$ query answers remains an open problem. A thorough study of $KFOPCE$-completeness will probably demand a study by cases as the one presented in Section 4.4.

5.3 An IDP-Based Prototype of the Local Closed-World Assumption

In this section, we present a simple prototype for query answering under the LCWA. The system is supported by the ID-Logic system IDP (Mariën et al. 2006).

The prototype works as follows. First, the user specifies an input file with the encoding of a locally complete database together with a $KFOPCE$ query. This file is then translated by an internal script into the fixpoint query described in Definition 5.4. Next, the IDP system finds a model of the fixpoint expression and the input database. The answers are then extracted from the computed model.

5.3.1 System Overview

The user specifies an input file with the following components:

- 1. A database instance D.
- 2. A set $\mathcal{L}$ of local closed-world assumptions.

- 3. A query $\mathcal{Q}[\bar{x}]$ in the language of $KFOPCE$.

The expressions in the input file follow the syntax of the IDP solver.[6] We illustrate the composition of each of these modules with a variant of Example 4.11.

- **Definition of the domain, types and database schema.** IDP is a typed language. Thus, every attribute of a predicate must be assigned to a type. Every type is defined by its domain. In the working example:

```
Given:
          type Person = {JohnSmith;PeterSteward;MaryClark}
          type Place = {Queens;Bronx;Manhattan}
          type CarID = {Bx5462;Bc5242;Qn5452}
          type Car= {Mercedes320;Volvo320;BMW}
          CarOwner(Person,Car,CarID)
          Location(Person,Place)
```

- **The database.** The database instance is specified by an enumeration of the atoms in D. Every constant in the database must be previously defined in the declaration of types.

```
Database:
{
          CarOwner(PeterSteward,Mercedes320,Qn5452).
          CarOwner(JohnSmith,Volvo320,Bc5242).
          CarOwner(MaryClark,BMW,Bx5462).

          Location(PeterSteward,Queens).
          Location(MaryClark,Bronx).
          Location(JohnSmith,Bronx).
}
```

- **Local closed-world assumptions.** The window of expertise of the LCWA follows the syntax of first-order IDP expressions. Thus, the expressions

$$\mathcal{LCWA}(CarOwner(x,y,z), Location(x, Bronx))$$

and

$$\mathcal{LCWA}(Location(x,y), (y = Bronx \wedge \exists z, uCarOwner(x,z,u))$$

are translated as:

[6] See http://www.cs.kuleuven.be/~ dtai/krr/software/idp.html for a detailed specification of the IDP system syntax.

```
Closure:
{
        LCWA[CarOwner(x,y,z);Location(x,u)&u=Bronx].
        LCWA[Location(x,y);y=Bronx & ?z u:CarOwner(x,y,u)].
}
```

- **The Query.** The query $Q[\bar{x}]$ is a $KFOPCE$ expression defined in the environment `Query`. Modalities $\mathbf{C}$ and $\mathbf{P}$ in $Q[\bar{x}]$ are specified by symbols $*$ and $\#$, respectively.

```
Query:
            type = Query(Person,Car,CarID)
            Query(x,y,z) = *[~CarOwner(x,y,z)]
```

The input file is pre-processed as follows[7]: `$ Perl lcwa inputfile.lcw`. The script `lcwa` produces a `.idp` extension file. This file is then processed by the IDP system with the command `$ IDP inputfile.idp`, which retrieves the answers to the original query.[8]

The underlying semantical foundations of the query answering mechanism of this prototype are those of Chapter 4. I.e., queries are posed against an implicit three-valued approximation of all the models of the database. Thus, as a consequence of Theorem 7, the system returns sound answers for queries that retrieve solely either possible or certain answers. Following the accuracy results of Section 4.4, for an important class of queries and databases, the system also retrieves complete answers. For general $KFOPCE$ queries, soundness Theorem 16 applies.

The basic functionality of the prototype is easily extended in at least two important directions.

First, following the discussion of Section 4.3.5, it is possible to incorporate functional dependencies (see Definition 4.13) into the system. The implementation of this kind of constraints amounts to encode each dependency as an additional rule (as described in Example 4.15) in the fixpoint query. The soundness of this transformation follows from Theorem 11.

A second extension comprises *views*. Intuitively, a view is a predicate *definition* (in deductive database terminology called *intensional predicate*) in terms of the other predicates of the database schema. Another (equivalent) perspective on views is that they are pre-computed queries. A view definition in the input file could look as follows:

```
Views:
Type = V(Person,Car,CarID)
{
```

[7] Available at http://www.cs.kuleuven.be/alvaro/lcwa.pl.

[8] The complete input file and its execution with IDP is presented in Appendix A.

```
        V(x,y,z) <- CarOwner(x,y,z) & Location(x,Queens).
}
```

Providing the right semantics to views is here slightly more complicated than in the case of functional dependencies. One alternative consists of restricting the type of views to non-recursive formulas. In such case, the view can be rewritten as a logically equivalent first-order formula by replacing the symbol `<-` by $\equiv$. The view `V` above, for instance, could be captured in first-order logic as follows:

$$\forall x, y, z \; V(x, y, z) \equiv CarOwner(x, y, z) \wedge Location(x, Queens).$$

If the view definition is recursive (recall Example 2.8), then first-order logic does not suffice. In such a case, the logical semantics of the view together with the database can be given in terms of an ID-Logic theory. Such knowledge base would consist of two components: a module with definitions encoding the views, and a module with assertions corresponding to the first-order logical semantics of the locally-closed database.

We lack the experience of evaluating the prototype in real life contexts. However, it is not difficult to see an important drawback when relying on a model generator system as IDP to perform query answering; namely, the encoding of a potentially large relational database together with a fixpoint query into a single input file. In practice, this is unfeasible. To tackle this problem, one alternative is to extend the prototype with an interface for communication with external relational database management systems (DBMS). Such a module would allow the system to access the data without comprising the functionality of the database.

The key concept behind choosing IDP as the supporting system of our prototype was *expressive power*. Indeed, as IDP provides the full functionality of ID-Logic, it is possible to encode arbitrary first-order formulas in the windows of expertise. However, in scenarios where less expressive languages are required, systems based on different principles might be good alternatives to implement the closed-world assumption. A Prolog encoding of the fixpoint query is one of those alternatives.

Indeed, a completely different approach to the problem of database interfacing is to encode the fixpoint query directly into a Prolog system. A good candidate for such an implementation is XSB Prolog (Sagonas et al. 1994). An important feature of XSB is the support of an ODBC driver for connecting with external database systems. In addition, XSB supports the well-founded semantics, one of the sustaining formalisms of the fixpoint queries we are interested in.

A disadvantage of encoding fixpoint queries in XSB with respect to IDP is its limited expressiveness: universal quantification in the bodies is not supported. We expect, however, that formulas with such shape would be rare in practice, and that in general an XSB-based implementation would perform well. For the

case of databases with windows of expertise consisting of base predicates, our tests seem to confirm this intuition. In Table 5.3, we show the execution times of XSB for two queries, $\mathcal{Q}_1$ and $\mathcal{Q}_2$, each consisting of two positive literals (one literal for retrieving possible answers, the other for retrieving certainly false answers) in conjunction ($\mathcal{Q}_1$) and disjunction ($\mathcal{Q}_2$), respectively. The window of expertise of each (original) predicate consists of a disjunction of a (variable) number of equalities and inequalities. The most left column of each table indicates the size of the windows of expertise ($\mathcal{L}$) of each predicate. The size of the database domain ($|D|$) in terms of number of atoms is indicated on top of each table.

As expected, the results indicate that running time scales up with the size of the windows of expertise and the size of the domain. In this test, the differences in the structure of the two queries do not seem to play an important role in the performance of the system.

$\mathcal{Q}_1$				
$\mathcal{L}\backslash\|D\|$	10^3	5×10^3	10^4	10^5
1	0.01	0.01	0.07	0.25
3	0.01	0.03	0.09	0.50
5	0.02	0.05	0.12	0.75
20	0.05	0.17	0.27	3.08
50	0.07	0.29	0.66	3.25

$\mathcal{Q}_2$				
$\mathcal{L}\backslash\|D\|$	10^3	5×10^3	10^4	10^5
1	0.01	0.01	0.01	0.55
3	0.01	0.03	0.12	0.99
5	0.02	0.07	0.15	1.84
20	0.05	0.22	0.31	3.21
50	0.08	0.39	0.68	4.21

Table 5.3: LCWA queries executed on XSB: CPU running time (in secs.).

Unexpectedly perhaps, tested on the same databases and queries, the prototype based on IDP showed a similar performance as XSB. From this data we can conclude that on small or moderate size databases with windows of expertise consisting of base predicates, both systems can be of practical utility.

5.4 Summary of the Results

We have divided this chapter in two main parts:

- In the first part, we extended the local closed-world assumption query language with modalities. This extension was motivated by the need for combining queries for certain and possible answers in a single expression. The logical foundations of the language were provided by Levesque's $KFOPCE$ logic, whose semantics is defined in terms of possible worlds. Since query answering under $KFOPCE$ is not tractable, we showed how to adapt the approximation techniques developed in Chapter 4 to the modal setting. We proved that, under certain conditions, the answers obtained using approximations were sound.

- In the second part, we presented a simple prototype for reasoning with the local closed-world assumption. We demonstrated its use with one of the working examples. The system relies on compiling fixpoint queries for computing approximative answers into a model generator. The results obtained for soundness and completeness of query answering using approximations can also be applied to answers obtained with the prototype.

5.5 Bibliographical Note

Modal logic has played an important role in computer science in general and in databases theory in particular (see, for instance, (Fagin et al. 1995)). The standard reference for a mathematical treatment of modal logic is the book by Chellas (Chellas 1980). A thorough analysis of the particular modal logic $KFOPCE$ and its application to incomplete knowledge bases can be found in Levesque's seminal paper (Levesque 1982). An application of $KFOPCE$ to relational databases with integrity constraints was proposed by Reiter (Reiter 1992). In that work, Reiter argues that integrity constraints are best viewed as true sentences about the knowledge on the database rather than axioms about the underlying domain. A detailed study about the computational complexity of various reasoning tasks under $KFOPCE$ is provided by Rosati (Rosati 2000). In the distributed database settings, modal logic has been applied in (Franconi et al. 2003) and (Calvanese et al. 2004). Common to both these approaches is the idea of modeling what the sources know about their domain with the modal operator **K**.

Chapter 6

Conclusions

The need to weaken the closed-world assumption in database systems and the ability to efficiently reason with partially complete databases is an important goal that has already been considered in several contexts. In this thesis, we extended the notion of local closed-world assumption to the first-order case, studied the complexity of query answering in locally closed databases, provided a modal extension for the standard first-order query language and implemented a prototype for query answering.

The contribution of this thesis is therefore:

1. A logical reconstruction of a theory for locally complete databases.
2. The formulation of efficient methods to reason with it.
3. The application of a rich modal formalism for querying locally complete databases.
4. The development of the essential functionality of a working system for reasoning with locally complete databases.

We have introduced a flexible and expressive generalization of Levy's method to express local completeness in either isolated or distributed databases. The semantics is given in a model-theoretic way using standard first-order logic. In order to compare our approach to related works, we have reformulated our semantics using, among others, circumscription and extended logic programming.

At the reasoning level, we have studied the tasks of retrieving possible or certain answers and checked whether a database has complete world information on queries. As under the general setting these tasks turned out to be computationally expensive, we have developed methods that, for restricted classes of databases and queries, retrieve answers efficiently. For this, we have introduced a logical framework based on three-valued logic to approximate answers.

In order to answer queries from implicit, auxiliary three-valued structures, we presented methods based on fixpoint computations. For an important class of queries and databases, we have shown that our methods are complete.

We believe a contribution of a different kind lies in the methodology by which we obtained the results above. A common mode of operation in database theory is to consider certain sublanguages (e.g., conjunctive queries) and design efficient algorithms for them. The strategy followed here is quite different. We presented tractable, but approximate methods for full first-order queries and local closed world assumptions. These methods are not perfect but may still be useful. E.g., it may be useful to get a subset of objects for which a certain query is certainly satisfied, or it may be useful to get a superset of objects for which a possible query might be satisfied. In a next phase, we then showed that these methods are optimal for certain subformulas and thus obtain tractable and optimal solutions for this class. In this process, the use of three-valued semantics plays an important role.

Motivated by the need of combining expressions for obtaining possible and certain answers in a single query, we applied Levesque's modal logic $KFOPCE$ as a query language to the setting of locally complete databases. Since query answering in $KFOPCE$ is intractable, we adapted the approximation techniques developed for standard first-order queries to $KFOPCE$. The result is an expressive query language for which we can retrieve approximate answers in a tractable way.

We finished this thesis by presenting a simple prototype for query answering under the local closed-world assumption. The system has its foundations in the approximation techniques developed in Chapter 4. The implementation is based on the ID-Logic solver IDP and provides additional support for views and the extended modal query language $KFOPCE$.

We conclude this thesis with some suggestions for future research:

- **Approximation methods for general first-order theories.** The approximation techniques presented in Chapter 4 were tailored for locally complete databases. However, the same principle of approximating answers could be applied to more general types of knowledge bases. Although promising steps have already been taken in that direction (Wittocx et al. 2008), much research is still needed.

- **Extensions of the LCWA to other related domains.** We have studied the local closed-world assumption in the setting of standard relational databases. Given the prevalent presence of semi-structured and non-structured data on the web (XML, HTML, web services), it would be interesting and natural to extend the notion of local completeness to less structured data models. A direct application of this would be the possibility of expressing local completeness over distributed but integrated web sources. Users could then be certain that (part of) a website contains *all* the information about a certain domain.

- **Real-life evaluation of the LCWA.** At the moment, we lack the experience to evaluate in practice the approximate methods of Chapter 4 and the implementation of Chapter 5. Although we have provided completeness results for a large class of queries and databases, critical evaluation of the usefulness of the LCWA in real life applications is still needed. Given their distributed nature, we believe that databases in the bioinformatics domain might be good candidates to perform this evaluation.

Appendix A

Apendix

A.1 Perl Script Output File

```
Given:
True
False
          type Person = {JohnSmith;PeterSteward;MaryClark}
          type Place = {Queens;Bronx;Queen;Manhattan}
          type Plaque = {Bx5462;Bc5242;Qn5452}
          type Car= {Mercedes320;Volvo320;BMW}

Find:
          Query(Person,Car,Plaque)

Declare:
          CarOwners(Person,Car,Plaque)
          CarOwnersCert(Person,Car,Plaque)
          CarOwnersCertFa(Person,Car,Plaque)

          Location(Person,Place)
          LocationCert(Person,Place)
          LocationCertFa(Person,Place)

          V(Person,Car,Plaque)
          VCert(Person,Car,Plaque)
          VCertFa(Person,Car,Plaque)

          R2(Person)
          R2Cert(Person)
```

```
          R2CertFa(Person)

Satisfying:
{
  VCert(x,y,z) <- CarOwnersCert(x,y,z) & LocationCert(x,Queens).
  VCertFa(x,y,z) <- CarOwnersCertFa(x,y,z) |
                            LocationCertFa(x,Queens).
}
{
          R2Cert(x) <- LocationCert(x,Bronx).
          R2CertFa(x) <- LocationCertFa(x,Bronx).
}
{
  CarOwnersCert(x0,x1,x2) <-  CarOwners(x0,x1,x2).
  CarOwnersCertFa(x,y,z) <- ~CarOwners(x,y,z) &
                                     LocationCert(x,u)&u=Bronx.
}
{
  LocationCert(x0,x1) <-  Location(x0,x1).
  LocationCertFa(x,y) <- ~Location(x,y) &
                            y=Bronx&!x y:CarOwnersCert(x,y,m).
}
{
  Query(x,y,z) <-  CarOwnersCertFa(x,y,z).
}
{
  CarOwners(PeterSteward,Mercedes320,Qn5452).
  CarOwners(JohnSmith,Volvo320,Bc5242).
  CarOwners(MaryClark,BMW,Bx5462).

          Location(PeterSteward,Queens).
          Location(MaryClark,Bronx).
          Location(JohnSmith,Bronx).
}
Data:
True = { () }
False = { }
```

A.2 IDP Execution Output File

```
This is MiniSat 2.0 beta
WARNING: for repeatability, setting FPU to use double precision
```

```
====================[ Problem Statistics ]==================
|                                                           |
| Reading ECNF file.                                        |
|    May contain inductive definitions.                     |
|    May contain exists unique statements.                  |
|    May contain aggregate expressions.                     |
| Number of exists unique statements:     0                 |
|    (there will be no exists unique propagations)          |
| Number of aggregate exprs.:    0                          |
|    (there will be no aggregate propagations)              |
| Number of rules          : 0                              |
| Number of recursive rules: 0                              |
|    (there will be no definitional propagations)           |
|  Parsing time:         0.00         s                     |
================[ Search Statistics ]======================
| Conflicts |  ORIGINAL         |    LEARNT     | Progress |
|           |Vars  Clauses L.   |L. C.   Lit/Cl |          |
============================================================
|   0       |  0   0     0      |  0   0    nan |  0.000 % |
Verified 0 original clauses.
SATISFIABLE
============================================================
restarts              : 1
conflicts             : 0          (0 /sec)
decisions             : 1          (0.00 % random) (250 /sec)
propagations          : 126        (31500 /sec)
conflict literals     : 0          ( nan % deleted)
Memory used           : 3.52 MB
CPU time              : 0.004 s

Query =
   {
    JohnSmith,BMW,Bc5242; JohnSmith,BMW,Bx5462;
    JohnSmith,BMW,Qn5452; JohnSmith,Mercedes320,Bc5242;
    JohnSmith,Mercedes320,Bx5462; JohnSmith,Mercedes320,Qn5452;
    JohnSmith,Volvo320,Bx5462; JohnSmith,Volvo320,Qn5452;
    MaryClark,BMW,Bc5242; MaryClark,BMW,Qn5452;
    MaryClark,Mercedes320,Bc5242; MaryClark,Mercedes320,Bx5462;
    MaryClark,Mercedes320,Qn5452; MaryClark,Volvo320,Bc5242;
    MaryClark,Volvo320,Bx5462; MaryClark,Volvo320,Qn5452}
```

References

Abiteboul, S. and O. Duschka (1998). Complexity of answering queries using materialized views. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington*, pp. 254–263.

Abiteboul, S., R. Hull, and V. Vianu (1995). *Foundations of Databases*. Addisin-Wesley Publishing Company.

Arieli, O., M. Denecker, and M. Bruynooghe (2007). Distance semantics for database repair. *Annals of Mathematics and Artificial Intelligence 50*(3-4), 389–415.

Bancilhon, F., D. Maier, Y. Sagiv, and J. D. Ullman (1986). Magic sets and other strange ways to implement logic programs. In *Proceedings of the Fifth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, March 24-26, 1986, Cambridge, Massachusetts*, pp. 1–15.

Bertossi, L. E. (1996). *Lógica para ciencias de la computacion*. Ediciones Universidad Catlica de Chile.

Bertossi, L. E. (2006). Consistent query answering in databases. *SIGMOD Record 35*(2), 68–76.

Bertossi, L. E. and L. Bravo (2004). Query answering in peer-to-peer data exchange systems. In *Current Trends in Database Technology - EDBT 2004 Workshops, EDBT 2004 Workshops PhD, DataX, PIM, P2P&DB, and ClustWeb, Heraklion, Crete, Greece, March 14-18, 2004, Revised Selected Papers*, pp. 476–485.

Boolos, G. and R. Jeffrey (1999). *Computability and Logic*. Cambridge University Press.

Brachman, C. and H. Levesque (2004). *Knowledge Representation and Reasoning (The Morgan Kaufmann Series in Artificial Intelligence)*. Morgan Kaufmann.

Bravo, L. and L. E. Bertossi (2006). Semantically correct query answers in the presence of null values. In *Current Trends in Database Technology - EDBT 2006, EDBT 2006 Workshops PhD, DataX, IIDB, IIHA,*

ICSNW, QLQP, PIM, PaRMA, and Reactivity on the Web, Munich, Germany, March 26-31, 2006, Revised Selected Papers, Volume 4254 of *Lecture Notes in Computer Science*. Springer.

Calvanese, D., G. D. Giacomo, M. Lenzerini, and R. Rosati (2004). Logical foundations of peer-to-peer data integration. In *Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 14-16, 2004, Paris, France*, pp. 241–251.

Chaudhuri, S. (1993). Finding nonrecursive envelopes for datalog predicates. In *Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 25-28, 1993, Washington, DC*, pp. 135–146.

Chaudhuri, S. and P. G. Kolaitis (1994). Can datalog be approximated? In *Proceedings of the Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 24-26, 1994, Minneapolis, Minnesota*, pp. 86–96.

Chellas, B. (1980). *Modal Logic: An Introduction*. Cambridge University Press.

Davey, B. and H. Priestley (2002). *Introduction to lattices and order (second edition)*. Cambridge university press.

Denecker, M. (2000, April 9-11). Extending classical logic with inductive definitions. In C. Baral and M. Truszczyński (Eds.), *8th Intl. Workshop on Non-Monotonic Reasoning (NMR'2000)*, Breckendridge, Colorado, USA, pp. 1–15.

Denecker, M., M. Bruynooghe, and V. Marek (2001, October). Logic programming revisited: Logic programs as inductive definitions. *ACM Transactions on Computational Logic 2*(4), 623–654.

Denecker, M., A. Cortés-Calabuig, O. Arieli, and M. Bruynooghe (2008). Accuracy of fixpoint methods for approximate query answering in locally complete databases. In preparation.

Denecker, M., V. Marek, and M. Truszczynski (2004). Ultimate approximation and its application in nonmonotonic knowledge representation systems. *Information and Computation 192*(1), 84–121.

Denecker, M. and E. Ternovska (2004). Inductive situation calculus. In D. Dubois and C. Welty (Eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)*, pp. 545–553.

Denecker, M. and E. Ternovska (2007). Inductive situation calculus. *Artificial Intelligence 171*(5-6), 332–360.

Doherty, P., W. Lukaszewicz, and A. Szalas (2000). Efficient reasoning using the local closed-world assumption. In *Proceedings of the Artificial*

Intelligence: Methodology, Systems, and Applications, 9th International Conference, AIMSA, Varna, Bulgaria, Volume 2407, pp. 49–58. LNCS.

Doherty, P., M. Magnusson, and A. Szalas (2006). Approximate databases: a support tool for approximate reasoning. *Journal of Applied Non-Classical Logics 16*(1-2), 87–118.

Duschka, O., M. Genesereth, and A. Levy (2000). Recursive query plans for data integration. In *Journal of Logic Programming*, Volume 1, pp. 49–73.

Duschka, O. M. (1997). Query optimization using local completeness. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island. AAAI Press / The MIT Press, 1997, ISBN 0-262-51095-2*, pp. 249–255.

Ebbinghaus, H.-D. and J. Flum (2005). *Finite Model Theory (2nd edition)*. Springer.

Enderton, H. B. (1972). *A Mathematical Introduction to Logic*. Academic Press.

Etzioni, O., K. Golden, and D. Weld (1997). Sound and efficient closed-world reasoning for planning. *Artificial Intelligence 89*(1-2), 113–148.

Fagin, R., J. Halpern, Y. Moses, and M. Vardi (1995). *Reasoning About Knowledge*. MIT Press.

Franconi, E., G. M. Kuper, A. Lopatenko, and L. Serafini (2003). A robust logical and computational characterisation of peer-to-peer database systems. In *Databases, Information Systems, and Peer-to-Peer Computing, First International Workshop, DBISP2P, Berlin Germany, September 7-8, 2003, Revised Papers*, pp. 64–76.

Garey, M. and D. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Gelfond, M. and V. Lifschitz (1988). The stable model semantics for logic programming. In *International Joint Conference and Symposium on Logic Programming (JICSLP'88)*, pp. 1070–1080. MIT Press.

Gelfond, M. and V. Lifschitz (1991). Classical negation in logic programs and disjunctive databases. *New Generation Computing 9*, 365–387.

Grahne, G. (1989). Horn tables - an efficient tool for handling incomplete information in databases. In *Proceedings of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, March 29-31, 1989, Philadelphia, Pennsylvania*, pp. 75–82.

Grahne, G. (2002). Information integration and incomplete information. *IEEE Data Eng. Bull. 25*(3), 46–52.

Grahne, G. and A. Mendelzon (1999a). Tableau techniques for querying information sources through global schemas. In *Proceedings of 7th International Conference on Database Theory ICDT*, Volume 1540, pp. 332–347. LNCS.

Grahne, G. and A. O. Mendelzon (1999b). Tableau techniques for querying information sources through global schemas. In *ICDT*, pp. 332–347.

Grahne, G. and A. Thomo (2001). Approximate reasoning in semistructured data. In *Proceedings of the 8th International Workshop on Knowledge Representation meets Databases (KRDB 2001), Rome, Italy, September 15, 2001*.

Grohe, M., T. Schwentick, and L. Segoufin (2001). When is the evaluation of conjunctive queries tractable? In *STOC*, pp. 657–666.

Halevy, A. Y., Z. G. Ives, D. Suciu, and I. Tatarinov (2003). Schema mediation in peer data management systems. In *Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India*, pp. 505–.

Halpern, J. Y., R. Harper, N. Immerman, P. G. Kolaitis, M. Y. Vardi, and V. Vianu (2001). On the unusual effectiveness of logic in computer science. *Bulletin of Symbolic Logic 7*(2), 213–236.

Kleene, S. (1952). *Introduction to Metamathematics*, Volume 1 of *Bibliotheca Mathematica*. Princeton, NJ *&* Groningen/Amsterdam: Van Nostrand *&* Wolters-Noordhoff/North-Holland.

Lenzerini, M. (2002). Data integration: A theoretical perspective. In *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*, pp. 233–246.

Levesque, H. (1990). All I Know: A Study in Autoepistemic Logic. *Artificial Intelligence 42*, 263–309.

Levesque, H. J. (1982). The logic of incomplete knowledge bases. In *On Conceptual Modelling (Intervale)*, pp. 165–189.

Levy, A. (1996). Obtaining complete answers from incomplete databases. In *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pp. 402–412.

Levy, A., A. Rajaraman, and O. J.J. (1996). Querying heterogeneous information sources using source descriptions. In *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, Volume 1, pp. 251–262.

Levy, A. Y., A. O. Mendelzon, Y. Sagiv, and D. Srivastava (1995). Answering queries using views. In *Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 22-25, 1995, San Jose, California*, pp. 95–104.

Levy, A. Y. and Y. Sagiv (1993). Queries independent of updates. In *19th International Conference on Very Large Data Bases, August 24-27, 1993, Dublin, Ireland, Proceedings*, pp. 171–181.

Libkin, L. (1995). *Elements of Finite Model Theory*. Springer.

Libkin, L. (1998). Models of approximation in databases. *Theoretical Computer Science 190*(2), 167–210.

Libkin, L. (2006). Data exchange and incomplete information. In *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*, pp. 60–69.

Lifschitz, V. (1994). Circumscription. In *Handbook of Logic in AI and Logic Programming*, Volume 3, pp. 298–352. Oxford University Press.

Lloyd, J. (1987). *Foundations of Logic Programming*. Springer-Verlag.

Loyer, Y. and U. Straccia (2005). Any-world assumptions in logic programming. *Theoretical Computer Science 342*(2-3), 351–381.

Mariën, M., J. Wittocx, and M. Denecker (2006). The IDP framework for declarative problem solving. In *Search and Logic: Answer Set Programming and SAT*, pp. 19–34. URL = http://www.cs.kuleuven.ac.be/cgi-bin-dtai/publ_info.pl?id=42379.

McCarthy, J. (1980). Circumscription - a form of nonmonotonic reasoning. *Artifical Intelligence 13*, 27–39.

McCarthy, J. (1986). Applications of Circumscription to Formalizing Common-Sense Knowledge. *Artifical Intelligence 28*, 89–116.

McCarthy, J. (1990). Applications of circumscription to formalizing common sense knowledge. In V. Lifschitz (Ed.), *Formalizing Common Sense: Papers by John McCarthy*, pp. 198–225. Ablex Publishing Corporation.

Mendelson, E. (1972). *Introduction to Mathematical Logic*. D. Van Nostrand company.

Minker, J. (1982). On indefinite databases and the closed world assumption. In *LNCS 138*, pp. 292–308. Springer-Verlag.

Motro, A. (1989). Integrity = validity + completeness. *ACM Transactions on Database Systems 14*(4), 480–502.

Nilsson, U. and J. Maluszyński (2000). *Logic, programming and prolog (second edition)*. John Wiley & Sons Ltd.

Pan, J. Z. and E. Thomas (2007). Approximating owl-dl ontologies. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pp. 1434–1439.

Pottinger, R. and A. Levy (2000). A scalable algorithm for answering queries using views. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, Volume 1, pp. 484–495.

Reiter, R. (1978). On closed world data bases. In H. Gallaire and J. Minker (Eds.), *Logic and Data Bases*, pp. 55–76. Plenum Press, New York.

Reiter, R. (1982). Towards a logical reconstruction of relational database theory. In *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages, Book resulting from the Intervale Workshop*, pp. 191–233.

Reiter, R. (1986). A sound and sometimes complete query evaluation algorithm for relational databases with null values. *Journal of the ACM 33*(2), 349–370.

Reiter, R. (1987). Nonmonotonic reasoning. In *Ann. Rev. Comput. Sci.*, Volume 2, pp. 147–182. Annual Reviews Inc.

Reiter, R. (1992). What should a database know? *J. Log. Program. 14*(1&2), 127–153.

Rosati, R. (2000). On the decidability and complexity of reasoning about only knowing. *Artif. Intell. 116*(1-2), 193–215.

Russell, S. and P. Norvig (2002). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Sagonas, K. F., T. Swift, and D. S. Warren (1994). XSB as an efficient deductive database engine. In R. T. Snodgrass and M. Winslett (Eds.), *SIGMOD Conference*, pp. 442–453. ACM Press.

Stuckenschmidt, H. and F. van Harmelen (2002). Approximating terminological queries. In *Flexible Query Answering Systems, 5th International Conference, FQAS 2002, Copenhagen, Denmark, October 27-29, 2002, Proceedings*, pp. 329–343.

Subrahmanian, V. S. (1992). Paraconsistent disjunctive deductive databases. *Theoretical Computer Science 93*(1), 115–141.

Swift, T. (1999). Tabling for non-monotonic programming. *Ann. Math. Artif. Intell. 25*(3-4), 201–240.

Trakhtenbrot, B. (1963). Impossibility of an algorithm for the decision problem in finite classes. *American Mathematical Society Transaction 3*(2), 1–5.

Ullman, J. D. (1988). *Principles of Database and Knowledge-Base Systems, Volume I*. Computer Science Press.

van Emden, M. and R. Kowalski (1976). The semantics of Predicate Logic as a Programming Language. *Journal of the ACM 23*(4), 733–742.

van Fraassen, B. C. (1966). Singular Terms, truth-Value Gaps, and Free Logic. *The journal of Philosophy 63*(17), 481–495.

Van Gelder, A. (1993). The alternating fixpoint of logic programs with negation. *Journal of Computer and System Sciences 47*(1), 185–221.

Van Gelder, A., K. A. Ross, and J. S. Schlipf (1991). The well-founded semantics for general logic programs. *Journal of the ACM 38*(3), 620–650.

Van Nuffelen, B., A. Cortés-Calabuig, M. Denecker, O. Arieli, and M. Bruynooghe (2004). Data integration using ID-logic. In *Proceedings of the Advanced Information Systems Engineering, 16th International Conference, CAiSE, Riga, Latvia, June 7-11*, Volume 3084, pp. 67–81. LNCS.

Vardi, M. (1982). The complexity of relational query languages. In *14th ACM Symposium on Theory of Computing*, pp. 137–145.

Wittocx, J., M. Mariën, and M. Denecker (2008). Approximate reasoning in first-order logic theories. In *Proceedingsof the 11th International Conference on Principles of Knowledge Representation and Reasoning, 2008*. AAAI Press.

List of Symbols

$\mathcal{C}_{\mathfrak{D}}$	Constructed three-valued approximation, page 91
$\mathcal{A}pp_{\mathfrak{D}}$	Operator in LCWA, page 90
$\mathcal{O}_{\Gamma}$	Optimal approximation, page 88
$\mathcal{C}(\Sigma)$	Set of constants of Σ, page 24
$\Delta^{c}_{\mathcal{Q},\mathcal{L}}$	Rules for certain answers, page 96
$\Delta^{p}_{\mathcal{Q},\mathcal{L}}$	Rules for possible answers, page 97
Δ	ID-Logic definition, page 33
$\prec$	Transitive closure full LCWA dependency graph, page 110
$\mathfrak{D}_{\mathcal{Q}}$	Database D extended with LCWAs for positive free predicates in $\mathcal{Q}$, page 114
$\mathfrak{L}_{\Sigma}$	Collection of two valued Herbrand interpretations of Σ, page 84
$\mathfrak{L}^{c}_{\Sigma}$	Collection of three-valued Herbrand interpretations of Σ, page 84
$\leq$	Order or partial order relation, page 18
$\leqslant_{\mathrm{p}}$	Precision order, page 83
$\leqslant$	Truth order, page 83
$\mathcal{A}$	Assertional knowledge, page 34
$\mathcal{LSA}$	Local soundness assumption, page 53
$\mathcal{L}(P)$	Propositional language based on P, page 22
$\mathcal{M}_D(\theta)$	Logical semantics of a LCWA θ in D, page 44
$\mathcal{O}$	Set of atoms, complete object assumption, page 51

$\mathfrak{C}(\mathfrak{D})$	Circumscriptive semantics of $\mathfrak{D}$, page 55
$\mathfrak{M}$	Mediator-based system, page 76
$\mathcal{M}_{st}(D)$	Logical semantics of D, page 39
$\prec^{\text{-}}$	transitive closure of the LCWA-dependency graph, page 110
$\preceq$	Reflexive closure of $\prec$, page 110
$\preceq^{-}$	Reflexive closure of $\prec^{\text{-}}$, page 110
$\Psi[\bar{x}]$	Windows of expertise of a LCWA, page 42
$\mathcal{Q}_l[\bar{x}]$	Lower bound Motro's algorithm, page 69
$\mathcal{Q}_u[\bar{x}]$	Upper bound Motro's algorithm, page 69
$\mathcal{R}(\Sigma)$	Set of relations of Σ, page 24
$\sigma^{-}_{\mathcal{Q}}$	$\prec$-downward closure of the set of predicates that occur negatively in $\mathcal{Q}$, page 114
$sv_{\mathcal{K}}(\varphi)$	Supervaluation of φ in $\mathcal{K}$, page 87
$\mathcal{THREE}$	Truth values of three-valued semantics, page 83
$\mathfrak{W}$	Set of worlds, page 131
D	Database instance, page 38
Dom^D	Active domain of D, page 38
$HB(\Sigma)$	Herbrand base of Σ, page 26
$HU(\Sigma)$	Herbrand universe of Σ, page 26
$InfS$	Infimum of S, page 19
Max_Q	Maximal element of a set Q, page 18
$mod(\mathfrak{D})$	All the models of $\mathfrak{D}$, page 133
$Open(D)$	Open predicates of D, page 34
P_i^c	Certainly true predicate, page 96
P_i^p	Possibly true predicate, page 96
$P_i^{c\neg}$	Certainly false predicate, page 96
$P_i^{p\neg}$	Possibly false predicate, page 96

R^c	Query rewriting function for certain answers, page 103
R^p	Query rewriting function for possible answers, page 103
S^l	Set of all lower bounds of S, page 19
S^u	Set of all upper bounds of S, page 19
$SupS$	Supremum of S, page 19
T_p	Immediate consequence operator, page 32
C	Certainly true, page 130
K	Modal operator $KFOPCE$, page 130
P	Possibly true, page 130
CWI	Closed-world information, page 59

Publications List

Contributions at international conferences published in proceedings (LNCS/LNAI or AAAI)

- A. Cortés-Calabuig, M. Denecker, O. Arieli and M. Bruynooghe, *Accuracy and efficiency of fixpoint methods for approximate query answering in locally complete databases*, 11th International Conference on Principles of Knowledge Representation and Reasoning, KR'08, pages 81-91, AAAI Press, 2008.

- A. Cortés-Calabuig, M. Denecker, O. Arieli and M. Bruynooghe, *Approximate query answering in locally complete databases*, 22nd National Conference on Artificial Intelligence, AAAI'07, pages 397-402, AAAI Press, 2007.

- A. Cortés-Calabuig, M. Denecker, O. Arieli and M. Bruynooghe, *Representation of partial knowledge and query answering in locally complete databases*, 13th Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'06, LNCS 4246, pages 407-421, Springer, 2006.

- A. Cortés-Calabuig, M. Denecker, O. Arieli, B. Van Nuffelen and M. Bruynooghe, *On the local closed-word assumption of data-sources*, 8th International Conference on Logic Programming and Non-monotonic Reasoning, LPNMR'05, LNAI 3662, pages 145-157, Springer, 2005.

- B. Van Nuffelen, O. Arieli, A. Cortés-Calabuig and M. Bruynooghe, *An ID-logic formalization of the composition of autonomous databases*, 8th International Conference on Logic Programming and Non-monotonic Reasoning, LPNMR'05, LNAI 3662, pages 132-144, Springer, 2005.

- B. Van Nuffelen, A. Cortés-Calabuig, M. Denecker, O. Arieli and M. Bruynooghe, *Data integration using ID-logic*, 16th Conference on Advanced Information System Engineering, CAISE'04, LNCS 3084, pages 67-81, Springer, 2004.

• L. Bertossi, J. Chomicki, A. Cortés-Calabuig and C. Gutiérrez, *Consistent answers from integrated data sources*, 5th International Conference on Flexible Query Answering Systems, FQAS'02, LNCS 2522, pages 71-85, Springer, 2002.

Biography

Álvaro Cortés was born on June 12, 1975 in Santiago, Chile. After finishing high school at Compañía de María Apoquindo School in 1993, he studied civil engineering at Universidad Católica de Chile (UC). In 2000, he obtained the 'licentiate' in sciences. In 2002 he graduated as a civil engineer with a specialization in industrial management. Between 2001 and 2002, he completed a master degree in theoretical computer science under the supervision of Leopoldo Bertossi. The title of his thesis was *Consistent Answers from Integrated Data Sources*.

Between 2002 and 2003, he was a part-time professor of databases at UC. At the same time, he worked as an independent consultant for the private sector.

In October 2003, he joined the Declarative Languages and Artificial Intelligence group of the Katholieke Universiteit Leuven, and started a Ph.D on incomplete databases under the supervision of Marc Denecker and Maurice Bruynooghe.