

MODEL THEORY AND CALCULUS FOR THE DESCRIPTION
LOGIC DL-LITE _{\mathcal{F}}

A DISSERTATION
SUBMITTED TO THE FACULTY OF COMPUTER SCIENCE
OF THE FREE UNIVERSITY OF BOZEN-BOLZANO
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Evgeny Kharlamov
October 2006

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Master of Science in Computer Science.

(Prof. Diego Calvanese) First Supervisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Master of Science in Computer Science.

(Prof. Werner Nutt) Second Supervisor

Abstract

The idea of using ontologies as a conceptual view over data repositories is becoming more and more popular. In practice, ontologies mediate between the repositories and a user [Len02], which allows the user to query the repositories by posing queries to the ontologies. Since data in the repositories is typically of a very large volume (much larger than the ontology itself), query answering should be efficient on the data. At present the standard ontology language is OWL¹, but none of its variants (OWL Full/DL/Lite) satisfies this efficiency requirement. In order to solve this problem, a new ontology language, called *DL-Lite* [CDGL⁺05b], was proposed; the language is a fragment of OWL Lite and allows for efficient query answering. In fact, *DL-Lite* is a family of various Description Logics [CGL⁺06], where *DL-Lite_F* and *DL-Lite_R* are the ones best studied to date.

We investigate two aspects of *DL-Lite_F*: properties of models of *DL-Lite_F* knowledge bases (KBs) and a calculus for the logic. As it is shown in [CGL⁺06], in general *DL-Lite_F* KBs have neither finite nor least (wrt inclusion) models. We construct examples to illustrate these phenomena. We introduce the notion of a universal model for a KB (which is based on the notion of a universal solution proposed in [FKMP05]) and show that any satisfiable *DL-Lite_F* KB has a universal model. We also show that a chase of a knowledge base is a universal model.

Answering queries over a KB requires in principle to evaluate the query over all models of the KB. We show that for answering conjunctive queries (CQs) over a *DL-Lite_F* KB it suffices to evaluate it only over a universal model of the KB, and consequently, over a chase of the KB². In general the chases of a KB may be infinite. We identify (using results from [FKMP05]) a class of KBs for which all chases are finite and, moreover, CQs can be answered in polynomial time.

However, it turns out that, also for KBs that have an infinite chase, it is possible to answer CQs in finite time. Calvanese et al. do this in [CDGL⁺05b] by presenting a query rewriting algorithm (with LOGSPACE data complexity) that takes as input a KB and a CQ and rewrites the query to a union of CQs which is then evaluated over the data stored in the KB in order to return all answers. We do this by defining a calculus that takes as input a KB and a CQ and allows one to derive all answers. We also show how the algorithm presented by Calvanese et al. in [CDGL⁺05b] can be obtained by imposing a specific control strategy on the calculus.

Finally, we present a way proposed in [CGL⁺06] to reduce several other reasoning tasks for DLs to answering CQs, so that it shows the calculus and the algorithm can be applied to a wide range of inferences.

¹<http://www.w3.org/TR/owl-features>

²The fact that evaluation of CQs over a *DL-Lite_F* KB can be reduced to their evaluation over a chase of the KB was presented first in [CGL⁺06].

Acknowledgements

I would like to thank my supervisors, Prof. Diego Calvanese and Prof. Werner Nutt, for many insightful conversations during the development of the ideas in this thesis, and for helpful comments on the text.

My deepest thanks to Prof. Enrico Franconi for his significant help with my studies. Thanks to Andrei Lopatenko for several very interesting discussions and suggestions. I thank Federica Cumer and Sylvia Epp for explaining and helping with bureaucratic stuff during my studies at FUB and TU Dresden.

This work could not have been done without the intellectual and social environment created by the members of the KRDB research center in the CS Faculty at FUB : Alessandro Artale, Raffaella Bernardi, Andrea Calí, Paolo Dongilli, Rosella Gennari, Marijke Keet, Manuel Kirschner, Davide Martinenghi, Mariano Rodriguez Muro, Vladisav Ryzhikov, Sergio Tessaris, Camilo Thorne.

I am very grateful for the financial support provided by Prof. Steffen Hölldobler as a Erasmus Mundus scholarship. The scholarship made my studies at TU Dresden and at FUB possible.

Contents

Abstract	iii
Acknowledgements	iv
1 Introduction	1
1.1 Information Integration	1
1.1.1 Virtual vs. Materialized Data Integration	2
1.1.2 Procedural vs. Declarative Data Integration	2
1.2 Ontologies	4
1.2.1 Usage of Ontologies for Modelling	4
1.3 DL Based Ontology Languages for Data Integration	6
1.3.1 DL-Lite Family	6
1.3.2 Reasoning in DL-Lite \mathcal{F}	7
1.3.3 Maximality of DL-Lite \mathcal{F} and DL-Lite \mathcal{R}	7
1.4 Contribution of Thesis	8
1.5 Organization of Thesis	8
1.6 Example of Using Ontologies in Information Integration	9
2 DL-Lite\mathcal{F}	11
2.1 Syntax	11
2.2 Semantics	13
2.3 Reasoning in DL-Lite \mathcal{F}	15
2.3.1 Query Answering for DL-Lite \mathcal{F}	15
3 Model-Theoretical Properties of DL-Lite\mathcal{F} Knowledge Bases	19
3.1 Lack of Finite Model Property	19
3.2 Lack of Least Model Property	20
3.3 Universal Model Property	21
3.3.1 Universal Models	21
3.3.2 Chase: Canonical Generation of Universal Model	23
3.3.3 Properties of Chase Sequences	26
3.3.4 Separation Theorem	30
3.3.5 Chase is a Universal Model	33

4	Conjunctive Query Answering for DL-Lite_{\mathcal{F}}	36
4.1	Conjunction Queries (CQs)	36
4.2	Model Theoretical Approach to Answering CQs	37
4.2.1	Chase of Polynomial Depth	39
4.3	Proof Theoretical Approach to Answering CQs	40
4.3.1	Extended Horn Clause (EHC) Logic	40
4.3.2	Translation from DL-Lite _{\mathcal{F}} Assertions and CQs to EHCs	41
4.3.3	Calculus for EHC Logic	43
4.3.4	Soundness of EHC Calculus.	49
4.3.5	Completeness of EHC Calculus	53
4.3.6	Algorithm to Implement EHC Calculus	58
5	Reduction of Reasoning in DL-Lite_{\mathcal{F}} to CQ Answering	60
5.1	KB Satisfiability	60
5.2	Instance checking	61
5.3	Subsumption Checking	61
5.4	Equivalence Checking	62
6	Conclusions	63
	Bibliography	65

Chapter 1

Introduction

1.1 Information Integration

Information integration has attracted a lot of attention in recent years. The key problem in this field is to access, relate and combine data from multiple sources. This problem is the basis of distributed databases, cooperative information systems, and data warehousing, which are important areas in the software development industry [Wie96, KL95, ZGMHW95, Hul97].

There are two main stages in the evolution of researches in information integration. The first one is known as *schema integration* and the second one as *data integration*. Schema integration was investigated in the 1980s in the context of database design. The goal was to design, for a given collection of independent database schemas, a global schema that reflects all the given ones [BLN86]. Data integration concerns a more general problem, which involves not only schemas but also the actual data in the integration process. In this context the goal is, for a given collection of data sources (schemas plus actual data),

1. to provide an integrated and reconciled *view* (called *global view* in the following) of the actual data from the sources, taking into account the autonomy of the sources [Ull97],
2. to use the global view in order to *manipulate* the data in the data sources, i.e. to query or/and update the data.

Systems that perform data manipulation using this kind of approach are called *information integration systems*. A general architecture of such systems is given in Figure 1.1. Data integration that involves the (integrated and reconciled) views for answering queries, but not for updating the sources is known as *read-only* integration. In this work we deal with read-only integration only.

There are two orthogonal ways to classify approaches to data integration. The first classification exploits the fact that the global view can be materialized or not. This corresponds to so-called *materialized* and *virtual* integrations, respectively. The second classification exploits the fact that the integration can have a single global view, which describes the whole data in the sources at once, or many small views, where each of them is adopted to a specific query. This corresponds to so-called *declarative* and *procedural* integrations, respectively. In the next two sections we discuss the classifications in more detail.

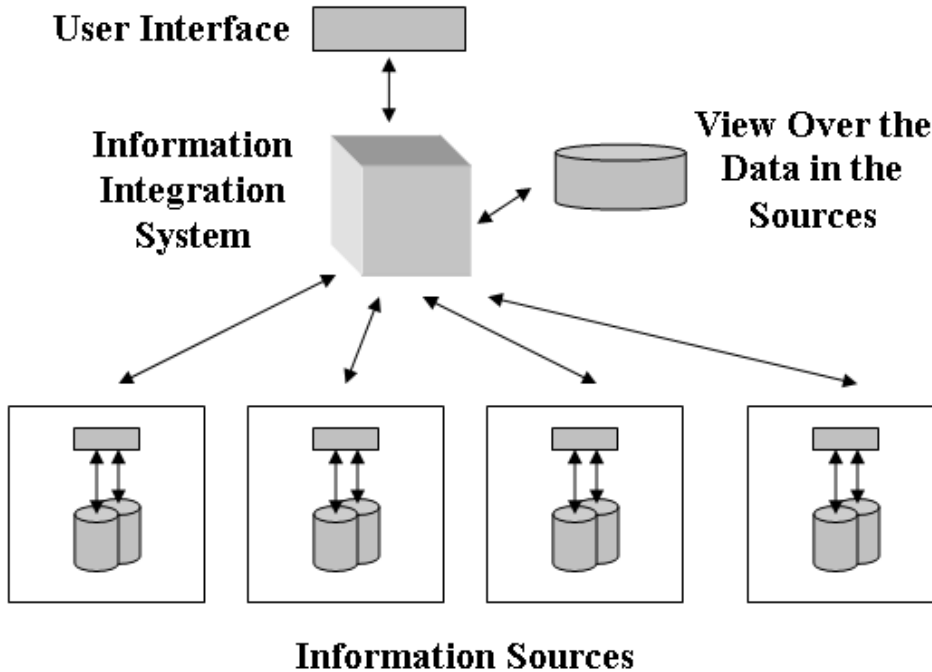


Figure 1.1: General architecture of information integration systems.

1.1.1 Virtual vs. Materialized Data Integration

In virtual integration the view of the data in the sources is not materialized, i.e., the view is a collection of descriptions written in some language that represents relationships between data elements in different sources. This view can involve an external vocabulary for describing the data elements and the relationship between them. The data integration systems that exploit this kind of views act as an interface between the user and the data sources [SL90, HBP94]. It is used in multidatabases, distributed databases, and more generally open systems. Query answering for virtual integration is generally costly, because it requires accessing the sources.

In materialized integration the global view is a replication of the data from the sources [GM95, Inn96]. In this case the data integration system maintains the replication and provide a querying interface for the user. It is used, for example, both in information system re-engineering and data warehousing. Query answering for materialized integration is generally more efficient, because it does not require accessing the sources, whereas maintaining the materialized views is costly, especially when the views must be up-to-date with respect to the updates at the sources (view refreshment).

In this work we deal with virtual data integration only.

1.1.2 Procedural vs. Declarative Data Integration

In procedural data integration, the views are created in an ad-hoc manner wrt a set of predefined querying needs. In this case the basic approach to design data integration system is to provide suitable software modules for the systems, which access the sources in order to fulfill the predefined querying needs. There are several data integration (both virtual and materialized) projects, such as

TSIMMIS [CGMH⁺94, Ull97], *Squirrel* [ZHK96, HZ96], and WHIPS [HGMW⁺95, WGL⁺96], that follow the procedural idea. They do not require an explicit notion of integrated data schema, and rely on two kinds of software components:

- *wrappers* that encapsulate sources, converting the underlying data objects to a common data model and
- *mediators* [Wie92] that
 - obtain information from one or more wrappers or other mediators,
 - refine this information by integrating and resolving conflicts among the pieces of information from the different sources and
 - provide the resulting information either to the user or to other mediators.

The basic idea is to have one mediator for every query pattern required by the user, and generally there is no constraint on the consistency of the results of different mediators.

In the declarative approach to data integration, the global view is a *unified representation* of the data in all the sources, moreover, it is a model of the domain of the data described in a suitable language. In this case data integration systems perform query answering in the following way:

- first, they consult the unified representation regarding relationships between the terms (e.g. names of the tables, attributes, etc.) used by the user in the query and the ones used in the sources,
- second, they create a so-called *query plan* that reflects an optimal, from the point of view of the system, way to access the data and to combine the retrieved result,
- third, they access the data sources using suitable mechanisms.

Note that the unified representation is a model of the domain of the data stored in the sources, but not of the sources themselves. For instance, in the data integration of several sources with information about car sellers, the representation is a model of the car selling domain, and this model not necessary depends on the particular data stored in the sources. This independence provides a crucial advantage of the declarative approach over the procedural one: although building a unified representation may be costly, the representation or the domain model is a reusable component of the data integration systems.

The declarative approach is used in *Carnot* [CHS91, HJK⁺93], SIMS [ACHK93, AKS96], *Information Manifold* [LSK95, KLSS95, LRO96], and DWQ [JLVV99, CDGL⁺98a, CDGL⁺98b].

In this work we deal with declarative virtual data integration only. In the next two sections of this chapter we first consider a general approach to use ontology languages for modelling data and ontologies as a unified representation formalism; second, we consider a specific data integration framework, where an ontology language called *DL-Lite_F* (see details in Section 1.2) is used for the modelling and discuss complexity of query answering and other reasoning tasks for this framework.

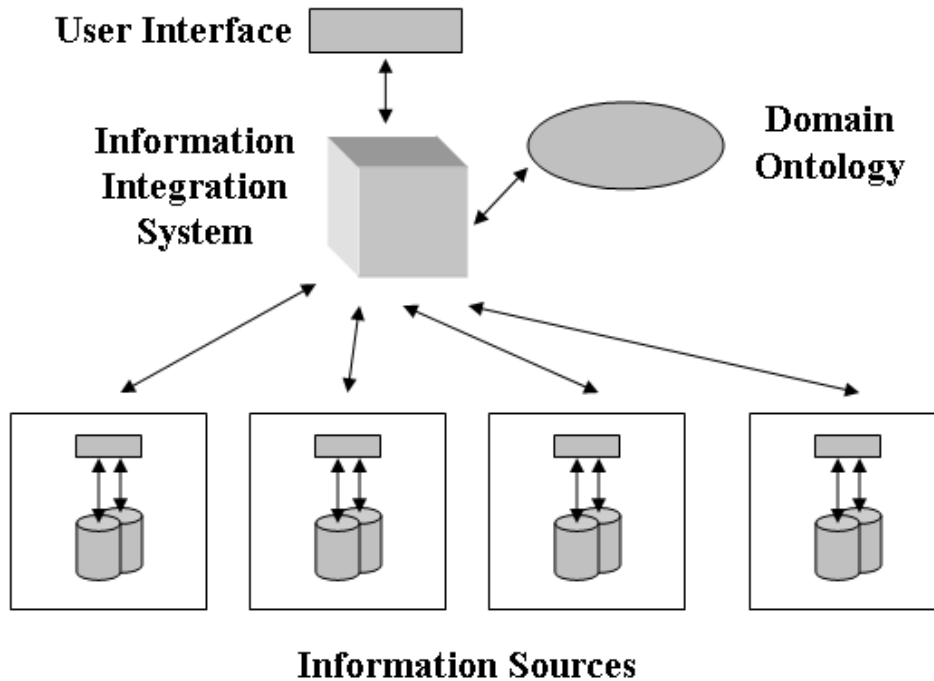


Figure 1.2: General architecture of information integration systems that use ontologies as a view over the data in the sources.

1.2 Ontologies

Ontology languages are formal languages that were specifically designed to describe data in unambiguous and machine processable way. A feature of ontology languages is that they represent the data in terms of classes (also called entities or concepts) and relationships between them. Since ontology languages use classes and relations to represent data, they allow one not only to describe particular elements of the domain of interest, but also to specify a conceptualization, i.e. an abstract view, of the domain [Gru93]. A conceptualization of the domain written in an ontology language is called *domain ontology* or simply *ontology*. For instance, ER diagrams in databases and information systems, or UML class diagrams in Software Engineering can be considered as ontologies. Ontologies can be also used in data integration as unified representations of the domain, and ontology languages can be used to write these representations. A general architecture of a data integration system where ontologies are used as unified representations is given in Figure 1.2.

Let us consider an evolution of ontologies as modelling tools in Computer Science.

1.2.1 Usage of Ontologies for Modelling

A real interest in ontology languages and ontologies in Computer Science started to emerge in the 1980s [dB03] in the areas of *Knowledge Engineering* (KE) and *Knowledge Representation* (KR). In these areas researches used the term *knowledge representation languages* instead of ontology languages. In the 1980s several knowledge representation systems such as KL-ONE [BS85] and CLASSIC [BBMR89] were developed, where subsets of first-order logic that later become known as

Description Logics (DL) [BCM⁺03] were used as ontology languages. Later on in the 1990s systems were developed such as LOOM¹, *Ontolingua* [Gru92] which used different DLs as ontology languages. *Ontolingua* allowed one to develop, manage and exchange ontologies. This system was also able to inter-operate with different ontology languages from different systems like KIF, KL-ONE, LOOM and CLASSIC, using the KIF language as an interchange language.

Almost in parallel with KE and KR, at the beginning of the 1990s, the term ontologies was first introduced in Computer Science by researchers in *Artificial Intelligence* and ontologies became a popular research topic. There were several well-known ontology based systems developed in this period, for instance, the WordNet ontology [Fel98], which models the English language. It contains a (natural language) description of English words and specifies three types of relationships between words, namely synonym and hyper-/hypo-nym (more/less general terms) relationships. The scope of WordNet is very broad but the level of detail is very low. Since the descriptions of the words are in natural language only, they are hardly machine-processable. Another example of an early ontology is CYC [Len95]. Again it is a very broad scope ontology, which attempts to capture all common-sense knowledge of human beings (i.e., it contains terms like space, time) by describing it with high level of details. CYC contains many formal relationships between different terms and is machine-processable.

Later, in the 1990s, there were first attempts to apply ontologies to the World Wide Web. For instance, in the SHOE project [HH00] ontologies were used to annotate web pages (through their embedding in HTML documents) in order to facilitate the process of information retrieval. In the *Ontobroker* project [FDES98] and its successor *On2broker* [FAD⁺00], that were carried out in the late 1990s, ontologies were used not only to annotate web pages but also to formulate queries and derive answers.

Starting from the beginning of the 2000s the idea of the so-called *Semantic Web* [BLHL01] appeared. The Semantic Web is an attempt to improve the World Wide Web from the current one, where the information is mostly presented in natural language and therefore can be understood by humans but not by machines, to a Web of a new generation, where the information will be presented in a machine readable and understandable form. An important European research project on *Knowledge Management* using the Semantic Web technology was the On-To-Knowledge project², which ran from 2000 until 2002. The focus of the project was knowledge management in large and distributed organizations using Semantic Web technology, and mainly ontologies. One of the outcomes of the project is the OIL³ (Ontology Inference Layer) [HFB⁺00] ontology language. OIL appeared in 2001 and became a basis for the DAML+OIL [Hor02] ontology language, which in turn became a basis for the ontology language OWL (Web Ontology Language) [BvHH⁺04]. OWL has three types, namely OWL Full, OWL-DL and OWL-Lite, which were developed by the Web Ontology workgroup at W3C (World Wide Web Consortium) and standardized by W3C. Nowadays, OWL has a broad support in both industry and academics and it is considered as *the* ontology language for the Semantic Web.

In the next section we first consider limitations in using of DL-based ontology languages in data integration and then discuss the *DL-Lite_F* ontology language, which was developed to overcome

¹<http://www.isi.edu/isd/LOOM/LOOM-HOME.html>

²<http://www.ontoknowledge.org/>

³<http://www.ontoknowledge.org/oil/>

these limitations.

1.3 DL Based Ontology Languages for Data Integration

The usage of ontology languages for data integration to specify a unified representation or a conceptual view over a given collection of sources always faces the problem of finding the right trade-off between *expressive power* of the language and *computational complexity* of different reasoning tasks for integration. If one considers ontology languages that are based on DLs, then the reasoning tasks include query answering, instance checking, subsumption, etc.

Let us now discuss this trade-off for ontology languages expressed in arbitrary DLs and then for ontology languages expressed in the so-called *DL-Lite* family of DLs, which will be introduced in detail later.

Research carried out in the past on investigating the trade-off between expressivity and complexity [BCM⁺03] shows that many DLs with efficient, PTIME, reasoning algorithms lack the modelling power to capture basic ontology languages, while most DLs with sufficient modelling power suffer from intractability, i.e. EXP TIME complexity of reasoning [BCM⁺03, BB03].

In the context of data integration even POL TIME complexity of reasoning turns out to be too high. It is the case because in several important applications, where the use of ontologies as conceptual views is advocated nowadays, the overall size of data sources is huge (e.g., from millions to trillions of instances). For instance, in the Semantic Web data sources are Web repositories and they may be numerous or incorporate very large data sets. Hence, in this context, neither the data integration systems which are equipped with expressive ontology languages and EXP TIME (in the size of the data sources) reasoners (e.g Fact⁴, Racer⁵), nor the ones, which are equipped with not very expressive ontology languages but relatively fast reasoners, are suitable for data integration needs. In order to solve this problem a new family of DLs, called the *DL-Lite* family was proposed.

1.3.1 DL-Lite Family

The *DL-Lite* family [CDGL⁺05b, CGL⁺06] is a family of DLs, in fact a fragment of OWL Lite, specifically tailored to capture basic ontology languages, while keeping many reasoning tasks tractable. In particular, the family has LOGSPACE (in the size of the data sources) complexity of all basic reasoning task for DLs and the same complexity of answering unions of conjunctive queries. There are three DL logics in the *DL-Lite* family. The basic one, so-called *DL-Lite_{core}*, was introduced first in [CDGL⁺05b]. The language of the logic allows one to express (cyclic) IS-A assertions on concepts, disjointness between concepts, role-typing, participation constraints, i.e., assertions stating that all instances of a concept participate to a specified role, and non-participation constraints. The other two logics are extensions of the basic one. The first extension is called *DL-Lite_F* and it extends *DL-Lite_{core}* with functionality restrictions on roles. The second one is called *DL-Lite_R* and it extends *DL-Lite_{core}* with IS-A and disjointness assertions between roles. Both extensions are expressible

⁴<http://www.cs.man.ac.uk/~horrocks/FaCT/>

⁵<http://www.sts.tu-harburg.de/~r.f.moeller/racer/>

enough to capture a variety of representation languages widely used in Computer Science. For instance, $DL-Lite_{\mathcal{R}}$ is a strict superset of RDFS⁶, and $DL-Lite_{\mathcal{F}}$ captures the fundamental constructs of the Entity-Relationship [AHV95] conceptual data model, and basic of UML class diagrams⁷ used in object-oriented formalisms.

1.3.2 Reasoning in $DL-Lite_{\mathcal{F}}$

As we already discussed above, the most important reasoning task in data integration is query answering. In a data integration setting where the global ontology is specified in $DL-Lite_{\mathcal{F}}$ and the data sources are stored as relational databases it has been shown that the query answering can be performed as a two step process [CDGL⁺05c]:

1. first step: to reformulate the query posed to the data integration system using the ontology only; the reformulation returns a union of conjunctive queries;
2. second step: to evaluate the union of conjunctive queries over the sources by an SQL engine.

Since the evaluation in the second step can be done using SQL engines, it exploits advantages of well established query optimization strategies. The complexity of the evaluation measured in the size of the data sources is LOGSPACE, because the first step does not depend on the data in the sources, and the second one is the evaluation of an SQL query over a databases, which is LOGSPACE in the size of the data [AHV95], and, consequently, the whole query answering process is in LOGSPACE in the size of the data.

As for the other reasoning tasks for DLs, it is shown in [CDGL⁺05a] that they can be reduced to conjunctive query answering and, consequently, performed in LOGSPACE in the size of the data.

The idea of taking advantage of well-established query optimization strategies supported by current industrial strength relational technology was one of the motivations behind several research works done on CLASSIC in the 1980s [BBMR89]. Based on this idea a reasoning system for $DL-Lite_{\mathcal{F}}$, called QuOnto [ACDG⁺05] has been developed.

1.3.3 Maximality of $DL-Lite_{\mathcal{F}}$ and $DL-Lite_{\mathcal{R}}$

As it is shown in [CDGL⁺05a] the logics of the DL-Lite family are essentially the maximal DLs for which conjunctive query answering can be computed in LOGSPACE, by allowing one to delegate query evaluation to an SQL engine. It is shown that even slight extensions of the $DL-Lite_{\mathcal{F}}$ and $DL-Lite_{\mathcal{R}}$ languages increase the complexity of query answering (in the size of the data) to NLOGSPACE-hard and CONP-hard, thus losing the possibility of reformulating queries in SQL. Actually for the extended languages where query answering is PTIME, it could still be feasible in practice, but the database technologies needed (Datalog engines) are not as well developed as standard SQL-based DBMSs.

⁶<http://www.w3.org/TR/rdf-schema/>

⁷<http://www.omg.org/uml/>

1.4 Contribution of Thesis

Conjunctive queries constitutes a large fragment of SQL, but there are more complex queries which are commonly used in practice and still expressible in SQL [AHV95], e.g. conjunctive queries with negation, comparisons and even aggregated queries. An important further investigation of $DL-Lite_{\mathcal{F}}$ is to understand whether it is possible to use wider fragments of SQL for query answering still keeping delegation of the query evaluation to SQL-based DBMSs. This task requires for better understanding of the theoretical properties of $DL-Lite_{\mathcal{F}}$ logic and this work is aimed to provide such an understanding.

We investigate two aspects of $DL-Lite_{\mathcal{F}}$: properties of models of $DL-Lite_{\mathcal{F}}$ knowledge bases (KBs) and a calculus for the logic. As shown in [CGL⁺06], in general $DL-Lite_{\mathcal{F}}$ KBs have neither finite nor least (wrt inclusion) models. We construct examples to illustrate these phenomena. We introduce the notion of a universal model for a KB (which is based on the notion of a universal solution proposed in [FKMP05]) show that every satisfiable $DL-Lite_{\mathcal{F}}$ KB has a universal model. We also show that a chase of a knowledge base is a universal model.

Query answering over a KB, using an arbitrary query language, requires in principle to evaluate the query over all models of the KB. We show that for answering conjunctive queries (CQs) over a $DL-Lite_{\mathcal{F}}$ KB it suffices to evaluate it only over a universal model of the KB, and consequently, over a chase of the KB⁸. In general the chases of a KB may be infinite. We identify (using results from [FKMP05]) a class of KBs for which all chases are finite and, moreover, CQs can be answered in polynomial time.

However, it turns out that, also for KBs that have an infinite chase, it is possible to answer CQs in finite time. Calvanese et al. do this in [CDGL⁺05b] by presenting a query rewriting algorithm (with LOGSPACE data complexity) that takes as input a KB and a CQ and rewrite the query to a union of CQs which is then evaluated over the data (ABox in terms of DLs) stored in the KB in order to return all answers. We do this by defining a calculus that takes as input a KB and a CQ and allows one to derive all answers. We show how the algorithm presented by Calvanese et al. in [CDGL⁺05b] can be obtained by imposing a specific control strategy on the calculus. Finally, we present a way proposed in [CGL⁺06] to reduce several other reasoning tasks for DLs to answering CQs, which shows the calculus and the algorithm can be applied to a wide range of inferences.

1.5 Organization of Thesis

The material of this thesis is organized in four chapters. Chapter 2 covers syntax, semantics and reasoning tasks for $DL-Lite_{\mathcal{F}}$. In Chapter 2 we also define query answering and data complexity for query answering.

Chapter 3 covers model-theoretical properties of $DL-Lite_{\mathcal{F}}$ KBs: we show lack of finite models (Section 3.1) and least models (Section 3.2) for arbitrarily $DL-Lite_{\mathcal{F}}$ KBs. In Section 3.3.1 we introduce so-called universal models for KBs over arbitrary DLs. In section 3.3.2 we present a chase for $DL-Lite_{\mathcal{F}}$ KBs and then in Section 3.3.4, using the chase and homomorphisms between interpretations, we prove a Separation Theorem, which shows that it is possible to separate evaluation

⁸The fact that evaluation of CQs over a $DL-Lite_{\mathcal{F}}$ KB can be reduced to their evaluation over a chase of the KB was first presented in [CGL⁺06].

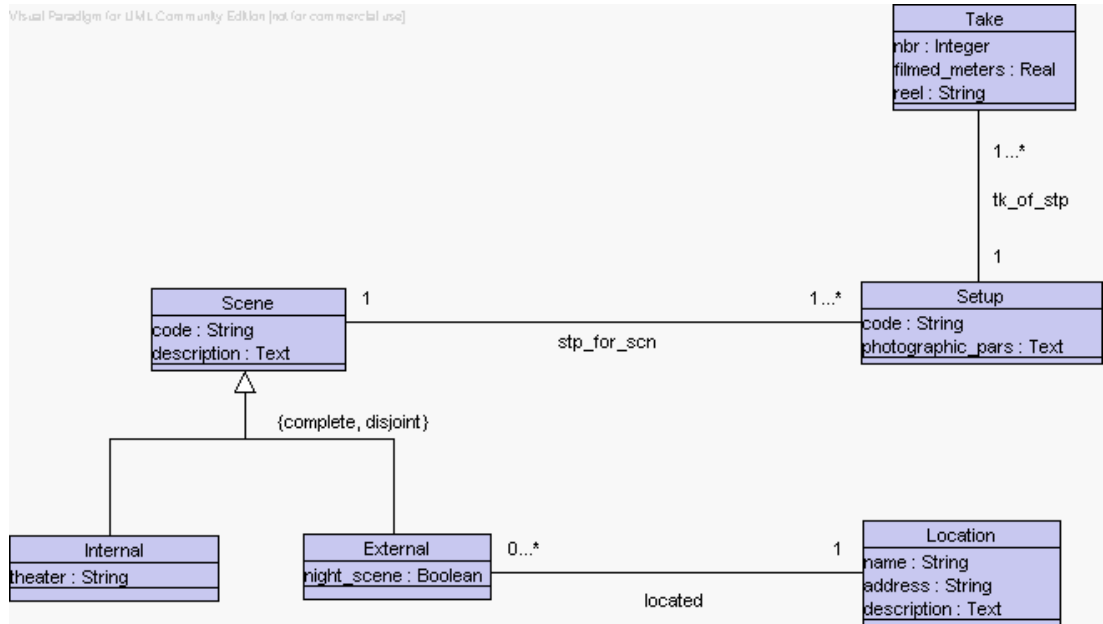


Figure 1.3: The UML class diagram represents the ontology for filmed scenes for realizing a movie.

of CQs over $DL-Lite_{\mathcal{F}}$ KBs in two steps as it was discussed above. In fact, the theorem was proved first in [CGL⁺06], here we re-prove it using another technique. In the last section of Chapter 3, we show that any satisfiable KB satisfies universal model property and that any chase of a KB is a universal model.

Chapter 4 introduces CQs and then two orthogonal approaches for answering CQs posed over $DL-Lite_{\mathcal{F}}$ KBs: a model theoretical (in Section 4.2) and a proof theoretical one (in Section 4.3). We introduce (in Section 4.2.1) the class of so-called weakly-acyclic $DL-Lite_{\mathcal{F}}$ KBs, for which the model theoretical approach is applicable and show complexity of reasoning for these KBs. In order to apply the approach one needs to embed $DL-Lite_{\mathcal{F}}$ in the so-called Extended Horn Logic (which is introduced in Section 4.3.1) and then apply a calculus (which is introduced in Section 4.3.3) to the result of the embedding. The embedding itself is introduced in Section 4.3.2. In Sections 4.3.4 and 4.3.5, we show respectively that the calculus is sound and complete wrt to answering CQs, i.e., for a given KB and a CQ the calculus allows one to derive all answers for the query over the KB. In Section 4.3.6 we present a way to implement the calculus.

Chapter 5 covers reductions from different reasoning tasks for $DL-Lite_{\mathcal{F}}$ to conjunctive query answering over $DL-Lite_{\mathcal{F}}$ KBs.

1.6 Example of Using Ontologies in Information Integration

Here we consider an example of constructing an ontology using several languages, namely, UML-class diagrams, natural language and $DL-Lite_{\mathcal{F}}$ (formally the language will be introduced in Chapter 2). The UML-class diagram is presented on Figure 1.3 and represents the ontology for filmed scenes for realizing a movie. Let us make descriptions of some fragments of the diagram first in natural language and then in $DL-Lite_{\mathcal{F}}$.

The class Scene is identified by a code (a string) and a text description. Scenes are filmed from different positions (at least one), which are called *setups* and can be either internal or external. In the diagram the class Scene is completely covered by the union of classes Internal and External, but we can not express it in $DL-Lite_{\mathcal{F}}$, since completeness is not expressible in this language. Let us write this part of the ontology in $DL-Lite_{\mathcal{F}}$ (see details of translation from UML to $DL-Lite_{\mathcal{F}}$ in [CCDGL01, BCDG01]):

$$\begin{aligned} \exists \text{Scene_code} &\sqsubseteq \text{Scene} \\ \exists \text{Scene_code}^- &\sqsubseteq \text{String} \\ \text{func}(\text{Scene_code}) & \end{aligned}$$

$$\begin{aligned} \exists \text{Scene_description} &\sqsubseteq \text{Scene} \\ \exists \text{Scene_description}^- &\sqsubseteq \text{Text} \\ \text{func}(\text{Scene_description}) & \end{aligned}$$

$$\begin{aligned} \text{Internal} &\sqsubseteq \text{Scene} \\ \text{External} &\sqsubseteq \text{Scene} \end{aligned}$$

where Scene_code, Scene_description denote the attributes “code” and “description” of the class Scene, respectively.

The func(stp_for_scn) relationship is expressed in $DL-Lite_{\mathcal{F}}$ as follows:

$$\begin{aligned} \exists \text{stp_for_scn} &\sqsubseteq \text{Scene} \\ \exists \text{stp_for_scn}^- &\sqsubseteq \text{Setup} \\ \text{Scene} &\sqsubseteq \exists \text{stp_for_scn} \\ \text{Setup} &\sqsubseteq \exists \text{stp_for_scn}^- \end{aligned}$$

$$\text{func}(\text{stp_for_scn})$$

The others classes with their attributes and the relationships can be expressed in $DL-Lite_{\mathcal{F}}$ similarly.

Chapter 2

DL-Lite \mathcal{F}

In this chapter we introduce $DL\text{-Lite}_{\mathcal{F}}$, a description logic (DL) from the $DL\text{-Lite}$ family introduced in [CGL⁺06]. We start with the syntax of the language, then continue with its semantics and present some reasoning tasks for the logic.

2.1 Syntax

The $DL\text{-Lite}_{\mathcal{F}}$ language, as any DL language, is to represent a domain of interest in terms of concepts (denoting sets of objects) and roles (denoting binary relations between objects). Any well-formed set of statements (formulae) written in $DL\text{-Lite}_{\mathcal{F}}$ is a description of a domain of interest. We start with the syntax of $DL\text{-Lite}_{\mathcal{F}}$ and give some (semantical) intuitions behind it.

To define the $DL\text{-Lite}_{\mathcal{F}}$ language we need to introduce several disjoint sets of symbols. Let $AC = \{A_1, \dots, A_{|AC|}\}$ be a finite set of *atomic concepts*, $AR = \{R_1, \dots, R_{|AR|}\}$ be a finite set of *atomic roles*. Let $Const$ be a countable set of *constants*, and Var be a countable set of *variables*. We also need two disjoint sets of constructors, namely $\{\exists, \neg, \sqcap, \neg\}$ and $\{\sqsubseteq, func\}$. The former set of constructors will be used to define *concepts* (terms) of the language and the latter one to define *assertions* (formulae) of the language. We use Σ to denote a functional free signature $AC \cup AR \cup Const$.

Definition 1. *Using standard DL notation, we (inductively) define DL-Lite \mathcal{F} concepts as words over the alphabet $AC \cup AR \cup \{\exists, \neg, \sqcap, \neg\}$ in the following way:*

$$\begin{array}{ll}
 \text{basic concept } B \longrightarrow & A \mid \text{atomic concept} \\
 & \exists R \mid \text{existential quantification on a role} \\
 & \exists R^{-} \mid \text{existential quantification on the role's inverse} \\
 \\
 \text{(general) concept } C \longrightarrow & B \mid \text{basic concept} \\
 & (\neg B) \mid \text{negation of a basic concept} \\
 & (C_1 \sqcap C_2) \mid \text{intersection of (general) concepts}
 \end{array}$$

where $A \in AC$ and $R \in AR$.

Due to the fact that negation is allowed in front of basic concepts only, we will eliminate brackets around negated concepts. We will also eliminate brackets around general concepts if it does not lead

to ambiguity. We denote the *set of all concepts* over given sets of atomic concepts AC and atomic roles AR as $Con(AC, AR)$. We write Con if it is clear from the context over which sets of atomic concepts and roles it is constructed.

Note, that in terms of \mathcal{FOL} concepts can be considered as terms. The difference between terms and concepts is that the former ones denote elements of the domain whereas the latter ones denote sets of elements of the domain. In this sense concepts are close to unary predicate symbols. In the same manner as \mathcal{FOL} terms are used to construct formulae, $DL-Lite_{\mathcal{F}}$ concepts (together with roles) are used to construct assertions. There are two types of assertions, so called *intensional* and *extensional* ones. Assertions of the first type describe the intensional knowledge about a domain of interest and those of the second type describe the intensional knowledge. We call them as usual *TBox assertions* and *ABox assertions* respectively. We formally define TBox assertions in the following way.

Definition 2. *TBox assertions are defined as words over the alphabet $AC \cup AR \cup \{\exists, \neg, \sqcap, \neg\} \cup \{\sqsubseteq, \text{func}\}$. Well formed assertions are words of the form:*

$$\begin{array}{ll} B \sqsubseteq C & \text{inclusion assertion} \\ (\text{func } R), (\text{func } R^-) & \text{functionality assertions} \end{array}$$

where B, R, C are a basic concept, an atomic role and a concept respectively.

An inclusion assertion expresses the fact that a basic concept is subsumed by a general concept, while a functionality assertion expresses the (global) functionality of a role, or of the inverse of a role.

Note that negation can appear on the right hand side of inclusion assertions only. The meaning of an inclusion $B \sqsubseteq \neg B'$ is the concepts B and B' are disjoint. Actually, the only reason why we introduce negation in the $DL-Lite_{\mathcal{F}}$ language is to have the ability to express disjointness of concepts. Moreover, disjointness is the only property expressible in $DL-Lite_{\mathcal{F}}$ by means of negation. Let us now define ABox assertions of the $DL-Lite_{\mathcal{F}}$ language.

Definition 3. *ABox assertions are defined as words over the alphabet $AC \cup AR \cup Const$. Well formed assertions are words of the form:*

$$A(a), R(a, b) \quad \text{membership assertion}$$

where A, R are an atomic concept and an atomic role respectively; a, b are constants.

These membership assertions state respectively that the object denoted by a is an instance of the atomic concept A , and that the pair of objects denoted by (a, b) is an instance of the role R .

We denote a set of TBox assertions as \mathcal{T} and a set of ABox assertions as \mathcal{A} . A $DL-Lite_{\mathcal{F}}$ knowledge base (KB) \mathcal{K} is defined as a pair $\langle \mathcal{T}, \mathcal{A} \rangle$ of the TBox \mathcal{T} and the ABox \mathcal{A} .

We will not consider membership assertions involving general concepts in this work. However, it can be shown that, with respect to the forms of reasoning considered in this work, presence of a membership assertion $C(a)$ in a KB can be simulated by adding $A \sqsubseteq C$ to the KB's TBox, and $A(a)$ to the KB's ABox, where A is an atomic concept that does not appear in the original KB.

Note that $DL-Lite_{\mathcal{F}}$ is a weaker version of OWL-Lite, the less expressive sublanguage of OWL¹. OWL Lite presents some constructs (e.g., cardinality restrictions on roles) that are non expressible in $DL-Lite_{\mathcal{F}}$.

2.2 Semantics

The semantics of $DL-Lite_{\mathcal{F}}$ is given in terms of interpretations over a fixed infinite countable *domain* Δ . We assume that the set $Const$ is under the *unique name assumption*, i.e. different constants denote different objects. Moreover for each object we can always find a constant denoting it. In other words, we have bijection between the set of constants and the domain or (in terms of [LL01]) we have *standard names*, and we will not distinguish between the alphabet of constants $Const$ and the domain Δ . Therefore, statements like "a constant c in Δ " should not lead to misunderstanding.

Recall that the signature $\Sigma = AC \cup AR \cup Const$ consists of sets of unary (AC), binary (AR) predicate symbols and constants ($Const$).

Definition 4. An interpretation $I = (\Delta, \cdot^I)$ (over Σ) for the set of $DL-Lite_{\mathcal{F}}$ atomic concepts AC and atomic roles AR is defined as a first order structure (over Σ) with Δ as a domain and an interpretation function described as follows:

$$\begin{aligned} A^I &\subseteq \Delta \\ R^I &\subseteq \Delta \times \Delta \end{aligned}$$

To define semantics for the set of concepts Con we extend the interpretation function \cdot^I following Definition 1 of concepts as follows:

$$\begin{array}{ll} \text{Extension of } \cdot^I: & A^I \subseteq \Delta \\ \text{(basic concepts)} & (\exists R)^I = \{c \mid \exists c'.(c, c') \in R^I\} \\ & (\exists R^-)^I = \{c \mid \exists c'.(c', c) \in R^I\} \\ \\ \text{Extension of } \cdot^I: & B^I = B^I \\ \text{(general concepts)} & (\neg B)^I = \Delta \setminus B^I \\ & (C_1 \sqcap C_2)^I = C_1^I \cap C_2^I \end{array}$$

We call the constant c' that appears in the definition above as *filler*. As we already said before, concepts are interpreted as sets of elements from the domain and roles as binary relations on the domain elements. Since the binary operator \sqcap is interpreted as intersection it is polyadic and all brackets that occur in concepts can be eliminated. Let us now define *truth valuation* for $DL-Lite_{\mathcal{F}}$ assertions.

¹<http://www.w3.org/TR/owl-features>

Definition 5. Let I be an interpretation (over Σ), then

$$\begin{aligned}
I \models B \sqsubseteq C & \quad \text{iff} \quad B^I \subseteq C^I \\
I \models (\text{func } R) & \quad \text{iff} \quad \text{for all constants } c, c', c'' \text{ s.t. } (c, c') \in R^I \text{ and } (c, c'') \in R^I \text{ holds } c' = c'' \\
I \models (\text{func } R^-) & \quad \text{iff} \quad \text{for all constants } c, c', c'' \text{ s.t. } (c', c) \in R^I \text{ and } (c'', c) \in R^I \text{ holds } c' = c'' \\
I \models A(a) & \quad \text{iff} \quad a \in A^I \\
I \models R(a, b) & \quad \text{iff} \quad (a, b) \in R^I
\end{aligned}$$

In Definition 5, one should read $I \models \phi$, where ϕ is an assertion, as “ ϕ is true in the interpretation I ”. If ϕ is true in the interpretation I , then we say that I is a *model* for ϕ . We say that an *assertion is satisfiable* if there is a model for it. We say that I is a model for a KB \mathcal{K} if it is a model for all its assertions. A *knowledge base \mathcal{K} is satisfiable* if there exists a model for it. We say that two assertions or two KBs are *equivalent* if they have the same set of models. For a given KB \mathcal{K} we denote the set of all models of \mathcal{K} as $Mod(\mathcal{K})$.

In terms of database theory (see, for instance, [AHV95]), a KB can be seen as an incomplete BD, where the ABox is a given BD and the TBox is a set of dependencies. Any model of the KB is a possible completion of the DB (ABox) wrt the dependencies (TBox).

Note that if a KB contains either TBox or ABox assertions only, then the KB is always satisfiable. A reason is that in the former case a trivial (empty) interpretation of all concepts occurring in the KB over any domain is always a model for the KB. In the latter case a *Herbrand base* over the symbols occurring in the KB is a model for the knowledge base.

Example 1 Let us consider two examples of unsatisfiable knowledge bases. The first KB \mathcal{K}_1 is the following:

$$\mathcal{K}_1 = \{A(a), A'(a), A \sqsubseteq \neg A'\}.$$

Obviously there is no model for \mathcal{K}_1 . The reason is that on the one hand concepts A and A' are disjoint and on the other hand they contain the same element a . In this example the *conflict* between concepts/assertions of the KB was given explicitly, that is, the disjointness of two concepts that have a common element is given as an inclusion assertion in \mathcal{K} .

As one can see from the second KB:

$$\mathcal{K}_2 = \{A(a), A'(a), A \sqsubseteq A'', A'' \sqsubseteq \neg A'\},$$

a conflict in the KB is implicit, that is, it is not explicitly stated but in fact can be easily checked using query answering (see Section 5 for efficient satisfiability checking).

Let us consider inclusion assertions of a special kind. We call inclusion assertions of the form $B \sqsubseteq B'$ or $B \sqsubseteq \neg B'$, where B and B' are basic concepts, as *positive (linear)* or *negative (linear)* assertions respectively. We say that a KB \mathcal{K} is *linearized* if each its assertion is either a membership assertion or a functionality assertion or a linear inclusion assertion. It is easy to check that any

inclusion assertion is equivalent to an assertion of the form:

$$B \sqsubseteq B_1 \sqcap \dots \sqcap B_n \sqcap \neg B'_1 \sqcap \dots \sqcap \neg B'_m,$$

where B , all B_i and B'_j are basic concepts. This assertion viewed as a KB is obviously equivalent to the following KB:

$$\{B \sqsubseteq B_1, \dots, B \sqsubseteq B_n, B \sqsubseteq \neg B'_1, \dots, B \sqsubseteq \neg B'_m\}.$$

Hence, each KB is equivalent to a linearized KB and later on without lose of generality we consider only linearized KBs.

Note that functional and negative inclusion assertions correspond (see, for instance, [AHV95]) to so-called *equality-* and *inequality-generating-dependencies* respectively. For the further convenience, let us define for a given KB \mathcal{K} the *active domain* of \mathcal{K} , denoted as $adom(\mathcal{K})$, to be the set of all constants occurring in \mathcal{K} . It is clear that the active domain of \mathcal{K} does not depend on its TBox, for there are no constants occurring in the TBox assertions. We define $sch(\mathcal{K})$ as the set of all atomic concepts and roles occurring in \mathcal{K} .

2.3 Reasoning in DL-Lite_F

Syntax and semantics for DL logics allow us to describe knowledge about a domain of discourse formally (due to the syntax) and unambiguously (due to the semantics). After the domain is described one is interested in reasoning about the domain. In this section we present several reasoning tasks for *DL-Lite_F* KBs, namely, KB satisfiability, instance checking, subsumption checking, equivalence checking and query answering.

KB satisfiability. For a given KB \mathcal{K} check whether \mathcal{K} is satisfiable.

Instance checking for a concept/role. For a given constant a /pair of constants a and b , concept C /role R and a KB \mathcal{K} , check whether $I \models C(a)/R(a, b)$ for every model $I \models \mathcal{K}$.

Subsumption checking. For given concepts C, D and a KB \mathcal{K} , check whether $I \models C \sqsubseteq D$ for every model $I \models \mathcal{K}$.

Equivalence checking. For given concepts C, D and a KB \mathcal{K} , check whether $I \models C \sqsubseteq D$ and $I \models D \sqsubseteq C$ for every model $I \models \mathcal{K}$.

In Chapter 5 we present efficient procedures for the reasoning tasks above.

2.3.1 Query Answering for DL-Lite_F

In this section we define further reasoning tasks for *DL-Lite_F* logic, namely query answering and query containment.

We define a query q as a *FOL* formula $\phi(\vec{x}; \vec{y})$, where \vec{x} is the vector of all free variables of ϕ

and \vec{y} is the (disjoint from \vec{x}) vector of all bound variables of ϕ . We call variables in \vec{x} *distinguished variables* for the query q and variables in \vec{y} *non-distinguished variables* for the query q . We denote the union of variables from \vec{x} and \vec{y} as $var(q)$. We use a denotation

$$q(\vec{x}) \leftarrow \phi(\vec{x}; \vec{y})$$

to express the fact that a query q has distinguished variables \vec{x} and is defined as a formula $\phi(\vec{x}; \vec{y})$. For our convenience, following logic programming notation, we call the formula $\phi(\vec{x}; \vec{y})$ as *body* of the query q and use the notation $body(q)$ or $body(\vec{x}; \vec{y})$ as an equivalent one to $\phi(\vec{x}; \vec{y})$. Using this notation a query q can be written as:

$$q(\vec{x}) \leftarrow body(\vec{x}; \vec{y}).$$

We say that a query q is over a KB \mathcal{K} if $body(q)$ satisfies the following conditions: there are no functional symbols occurring there; all the predicate symbols occurring there are from $sch(\mathcal{K})$; and all constants occurring there are from $adom(\mathcal{K})$.

Let us define *query answering* for a query over a model of a given KB.

Definition 6. *Given a KB \mathcal{K} , a model I of \mathcal{K} , a query $q(\vec{x}) \leftarrow \phi(\vec{x}; \vec{y})$ over \mathcal{K} , the answer set of q over I is defined as the following set.*

$$q(I, \mathcal{K}) = \{\gamma(\vec{x}) \mid I, \gamma \models \phi(\vec{x}; \vec{y})\}$$

where $\gamma : Var \rightarrow \Delta$.

In Definition 6, the mapping γ is defined on variables Var , but it can be naturally extended to tuples of variables. One should view $\gamma(\vec{x})$ as application of the extended γ . We say that the set $q(I, \mathcal{K})$ is the result of the *answering* of the query over a model I of a given KB \mathcal{K} .

We defined how to answer a query over a single model of a KB. Let us now define how to answer a query over the KB itself. It is not an obvious task, for there is an ambiguity arising from the fact that, as one can see from the Sections 3.2, and 3.1, there may be many (not isomorphic and possibly infinite) models I for a KB \mathcal{K} and, as a result, different such models I may produce different answers $q(I, \mathcal{K})$. This conceptual difficulty was first encountered in the context of incomplete databases, where one has to find the “right” answers to a query posed against a set of “possible” databases (see, for instance, [vdM98]). An incomplete database can be thought of as the set of all databases that satisfy a certain specification, that is, all databases that are “possible” for the given specification. In this sense, a KB \mathcal{K} can be seen as a specification (that describes a set of constrains/assertions) that is to be satisfied; at the same time the set of all models of \mathcal{K} can be seen as an incomplete database. Now, suppose that a query is posed against an incomplete database. There is general agreement that in this context, the “right” answers are the *certain answers*, that is, the answers that occur in the intersection of all $q(I)$ ’s, as I varies over all “possible” databases. This notion makes good sense for *DL-Lite_F* as well, where, as discussed above, the “possible” databases are models of the KB. It also has the benefit that the query semantics is independent of the specific model of the KB but depends only on the KB itself. We thus adopt the certain answers as the

semantics of query answering for *DL-Lite_F* and investigate the complexity of computing the certain answers for *DL-Lite_F* KBs. A related important question is whether the certain answers of a query can be computed by query evaluation in a “good” model of the KB. Let us now formally define the discussed reasoning task for *DL-Lite_F*, i.e. query answering for a query over a KB.

Query answering. For a given query q and a KB \mathcal{K} , return the set of *certain answers* $q(\mathcal{K})$ defined as follows:

$$q(\mathcal{K}) = \{\vec{c} \mid \vec{c} \in \text{adom}\mathcal{K}^{|\vec{c}|}, \text{ and } \vec{c} \in q(I, \mathcal{K}) \text{ for every model } I \models \mathcal{K}\}.$$

We call the set $q(\mathcal{K})$ the *answer set* for q over \mathcal{K} . Note that if \mathcal{K} is unsatisfiable, then $q(\mathcal{K})$ is equal to $\Delta^{|\vec{x}|}$. The next reasoning task is *query containment*.

Query containment. For two given queries q_1, q_2 and a KB \mathcal{K} , check whether

$$q_1(I, \mathcal{K}) \subseteq q_2(I, \mathcal{K})$$

holds for every model $I \models \mathcal{K}$.

Note that for a given query $q(\vec{x}) \leftarrow \phi(\vec{x}; \vec{y})$ and a KB \mathcal{K} , query answering is searching for all tuples \vec{c} of constants from $\text{adom}(\mathcal{K})$ such that the following entailment holds:

$$\mathcal{K} \models \phi(\vec{c}; \vec{y}).$$

Let us now define complexity of query answering. We define it as the complexity of the following decision problem:

For a given KB \mathcal{K} , a query q and a vector \vec{c} of constants from $\text{adom}(\mathcal{K})$ decide whether $\vec{c} \in q(\mathcal{K})$

The most relevant parameter to measure complexity is the size (number of membership assertions) of the \mathcal{K} 's ABox. Complexity measured using this parameter is called *data complexity* (it was first introduced in [Var82]). Of course one can use the size of the KB \mathcal{K} (the size of its ABox plus the size of its TBox, i.e. the number of basic concepts occurring in its TBox) as a parameter to measure complexity. This kind of complexity is called *combined complexity*. Data complexity is relevant when the size of the KB's ABox much bigger than the size of the KB's TBox, which is actually the case in most practical applications. Combined complexity is relevant when an ABox and a TBox have comparable sizes.

Let us now consider a decision problem which is equivalent to the one introduced above:

Check that for a given KB \mathcal{K} , a query $q(\vec{x}) \leftarrow \phi(\vec{x}; \vec{y})$ and a vector \vec{c} of constants from $\text{adom}(\mathcal{K})$, the entailment $\mathcal{K} \models \phi(\vec{c}; \vec{y})$ holds.

To decide the entailment one needs to check that all models of \mathcal{K} satisfy $\phi(\vec{c}; \vec{y})$. As one can see from Sections 3.2, and 3.1, there may be an infinite number of (possibly infinite) models of \mathcal{K} . Hence, it is not clear at all whether query answering is decidable and if it does then what is the complexity

of the decision procedure. It turns out that for particular classes of queries, namely, for *conjunctive queries* the problem is LOGSPACE decidable in data complexity (see details Section 4.3.6).

Chapter 3

Model-Theoretical Properties of DL-Lite \mathcal{F} Knowledge Bases

In this chapter we consider three properties of KBs, namely, *finite model property*, *least model property* and *universal model property*.

3.1 Lack of Finite Model Property

We say that a set of assertions Γ meets *finite model property* (FMP) if Γ is satisfiable and there is a finite model I for Γ . Unfortunately, there are *DL-Lite \mathcal{F}* knowledge bases that do not meet FMP. To illustrate the phenomena let us consider axiomatization of forests of trees with infinite depth and unbounded branching factor. We first present the axiomatization in a natural language and then express it in *DL-Lite \mathcal{F}* . The axioms are the following:

- (1) Every root node has a child.
- (2) A root node does not have a parent.
- (3) Every node that has a parent, has a child.
- (4) Every node has at most one parent.

These axioms can be translated in the following *DL-Lite \mathcal{F}* TBox \mathcal{T} :

- (1) $Root \sqsubseteq \exists Child$
- (2) $Root \sqsubseteq \neg \exists Child^-$
- (3) $\exists Child^- \sqsubseteq \exists Child$
- (4) $func(Child^-)$

To obtain a single tree of an infinite depth with a root element a one need to add the following fact to the above forests' axioms:

- (5) an element a is a root.

This fact corresponds to the ABox \mathcal{A} :

(5) *Root*(a)

It is easy to check that the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable and there is no finite model for \mathcal{K} . Indeed, assume that an interpretation I is a model for \mathcal{K} . Due to the assertion (5), a is a *Root* element in I . Due to (1), there is a child element, say n , of the element a and, due to (2), n is different from a . Due to the assertion (3), there is a child of the element n in I , say m , such that, due to (4), m is different from n . Applying assertions (3) and (4) to m and its descendants we obtain an infinite chain of elements connected through the binary relation *Child*. This implies that I should “contain” an infinite chain of descendants of the element a . Hence, I is infinite. Note, that if one disallows functionality assertions in *DL-Lite_F*, it will lead to a DL language that meets FMP.

3.2 Lack of Least Model Property

We say that a set Γ of assertions meets *least model property* (LMP) if Γ is satisfiable and there is a (least) model I for Γ , which is contained in any other model of Γ . Note, that a model I for Γ is contained in a model I' for Γ if both models have the same signature and for any symbol P in the signature the containment $P^I \subseteq P^{I'}$ holds. The least model for Γ is in fact an intersection between all Γ 's models, where the intersection of a set of models is defined as the maximal (wrt containment) model that is contained in all the models from the set.

Unfortunately, the LMP does not hold for an arbitrary KB \mathcal{K} . The following example gives intuitions why this happens.

Example 2 Let us consider the following KB:

$$\mathcal{K} = \{A(a), A \sqsubseteq \exists R\}.$$

Interpretations $I_1 = \langle \Delta, \cdot^{I_1} \rangle$, where $A^{I_1} = \{a\}$, $R^{I_1} = \{\langle a, b \rangle\}$ (b is an element from Δ) and $I_2 = \langle \Delta, \cdot^{I_2} \rangle$, where $A^{I_2} = \{a\}$, $R^{I_2} = \{\langle a, c \rangle\}$ (c is an element from Δ) are models for \mathcal{K} . However, the intersection of them $I_1 \cap I_2 = \langle \Delta, \cdot^{I_1 \cap I_2} \rangle$, where $A^{I_1 \cap I_2} = a$, $R^{I_1 \cap I_2} = \emptyset$, obviously is not a model for \mathcal{K} . Therefore, there is no least model for the observed DB \mathcal{K} .

Intuitively the reason why in the example the intersection of two models of \mathcal{K} is not a model of \mathcal{K} is that the existential quantification on the role R does not fix the exact filler of R . Hence one can choose any element of the domain as a filler and obtain a model for \mathcal{K} . Similar problem appears when one tries to find a least model for a disjunction of formulae.

An interesting observation is that if a KB consists of an ABox only, then the least model always exists. The least model is simply a Herbrand model that coincides with the ABox viewing as a collection of ground atoms.

Let us now look on the two models from the Example 2. more carefully. Are they really different one from another? If we are able to distinguish elements from $adom(\mathcal{K})$ only, then the models look quite similar. If one define a mapping that is an identity on the constant a , maps the constants b, c to the same constant d and if one extend the mapping from constants to models, then both models will be mapped into the same one. In the Section 3.3 we define this mapping formally and define

a *universal model* using the mapping. This will allow us to define a *universal model property* for *DL-Lite_F* knowledge bases. In particular the models from the example are isomorphic and each of them is universal for the KB \mathcal{K} .

3.3 Universal Model Property

In this section we define *universal models* and investigate properties of such models. We also define *universal model property* for KBs and show that any satisfiable *DL-Lite_F* KB meets this property. In particular we show that a *chase* of a *DL-Lite_F* KB is a universal model of the KB.

3.3.1 Universal Models

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a knowledge base. We define a set $ln(\mathcal{K})$ of *labelled nulls* for \mathcal{K} as the set of all constants that do not occur in \mathcal{K} (actually, in \mathcal{A} because constants never occur in TBoxes). The word “nulls”, as in the case of databases (compare to *null values* in [AHV95], page 488), reflects the fact that the constants in $ln(\mathcal{K})$ are “unknown” for the KB \mathcal{K} , that is, they are not explicitly listed in the ABox. The word “labelled” reflects the fact that we are able to distinguish one null from another. Obviously the sets $adom(\mathcal{K})$ and $ln(\mathcal{K})$ are disjoint and their union covers the set $Const$. For a given interpretation I , we denote as $ln(\mathcal{K}, I)$ a subset of $ln(\mathcal{K})$ such that any ground atomic fact $P(\vec{c})$ over Σ that contains a constant from $ln(\mathcal{K}, I)$ is false in I . The set $ln(\mathcal{K}, I)$ essentially contains labelled nulls for \mathcal{K} which are not presented in I viewed as a collection of facts. We call $ln(\mathcal{K}, I)$ as a set of *fresh labelled nulls* for I .

Definition 7. *Let \mathcal{K}_1 and \mathcal{K}_2 be two knowledge bases such that $adom(\mathcal{K}_1) \subseteq adom(\mathcal{K}_2)$; I_1 and I_2 be models of the KBs respectively.*

1. *A homomorphism $h : I_1 \rightarrow I_2$ is a mapping from $Const$ to $Const$ such that: (i) $h(c) = c$, for every $c \in adom(\mathcal{K}_1)$; (ii) for an atomic concept A in $sch(\mathcal{K}_1)$ and a constant c if $I_1 \models A(c)$, then $I_2 \models A(h(c))$ (ii) for an atomic role R in $sch(\mathcal{K}_1)$ and a pair of constants (c_1, c_2) if $I_1 \models R(c_1, c_2)$, then $I_2 \models R(h(c_1), h(c_2))$ (where, $h(c_1, c_2) = (h(c_1), h(c_2))$).*
2. *I_1 is homomorphically equivalent to I_2 if there is a homomorphism $h : I_1 \rightarrow I_2$ and a homomorphism $h' : I_2 \rightarrow I_1$.*

In the definition above we put the restriction $adom(\mathcal{K}_1) \subseteq adom(\mathcal{K}_2)$, because, following the definition of homomorphism between two tableaux [AHV95], we want constants from $adom(\mathcal{K}_1)$ not to be mapped to constants from $adom(\mathcal{K}_2)$ but not to labelled nulls from $ln(\mathcal{K}_2, I_2)$.

Let us now define a *universal model*, which is a crucial notion for this work.

Definition 8. (*Universal model*). *Let \mathcal{K} be a knowledge base. A universal model for \mathcal{K} is such a model $UI \models \mathcal{K}$ that for every model $I \models \mathcal{K}$, there exists a homomorphism $h : UI \rightarrow I$.*

Example 3. Consider the Example 2 again. The models I_1 and I_2 are homomorphically equivalent. A homomorphism from I_1 to I_2 is $h = \{a \mapsto a, b \mapsto c\}$. A homomorphism from I_2 to I_1 is $h' = \{a \mapsto a, c \mapsto b\}$. Consider a model I_3 for \mathcal{K} , such that $A^{I_3} = \{a, c\}$ and $R^{I_3} = \{(a, b), (c, c')\}$. There is an obvious homomorphism from I_1 to I_3 which is the identity function. A homomorphism

from I_3 to I_1 is $h = \{a \mapsto a, b \mapsto b, c \mapsto a, c' \mapsto b\}$. So, the models I_1 and I_3 are homomorphically equivalent as well. It is easy to show that I_1 is a universal model. The reason is that any other model of \mathcal{K} , say I' , should obviously assign $A(a)$ to true and should also assign $R(a, x)$ to true, for some $x \in \text{In}(\mathcal{K})$. Hence, a mapping that maps a to itself and b to x is an homomorphism from I_1 to I' . Using the fact that I_2 and I_3 are homomorphically equivalent to I_1 we obtain that models I_1 , I_2 and I_3 are universal. Actually, the fact that I_1 and I_2 are universal is quite intuitive, both models “contains” not more and not less information than it is actually in the KB. The fact that I_3 is a universal model is not that intuitive. To separate intuitive universal models from non intuitive ones, we will introduce (see Section ??) a notion of *least universal model*, which generalize a notion of least modal.

Now let us consider a model for \mathcal{K} which is not a universal one. Let I_4 be a model for \mathcal{K} , such that $A^{I_4} = \{a, c\}$ and $R^{I_4} = \{(a, c)\}$. There is no way to construct a homomorphism from I_4 to I_1 , because any homomorphism should map c to a (due to the (i) property of Definition 7 of the homomorphism), hence a pair (a, c) will be mapped to (a, a) , but this pair does not belong to R^{I_1} , that is in contradiction with the (ii) property of Definition 7. Hence, I_4 can not be homomorphically mapped to one of the models of \mathcal{K} and, consequently, it is not a universal model.

Definition 9. We say that a KB in some DL language meets a universal model property if there is a universal model for the KB.

The next result shows good properties of universal models.

Proposition 1. Let \mathcal{T} be a TBox.

1. If \mathcal{A} is an ABox and UI_1 and UI_2 are two universal models for $\langle \mathcal{T}, \mathcal{A} \rangle$, then UI_1 and UI_2 are homomorphically equivalent.
2. Let \mathcal{A}_1 and \mathcal{A}_2 be two ABoxes; UI_1 a universal model for $\langle \mathcal{T}, \mathcal{A}_1 \rangle$; UI_2 a universal model for $\langle \mathcal{T}, \mathcal{A}_2 \rangle$. An inclusion $\text{Mod}(\langle \mathcal{T}, \mathcal{A}_1 \rangle) \subseteq \text{Mod}(\langle \mathcal{T}, \mathcal{A}_2 \rangle)$ holds if and only if there is a homomorphism $h : UI_2 \rightarrow UI_1$. Consequently, $\text{Mod}(\langle \mathcal{T}, \mathcal{A}_1 \rangle) = \text{Mod}(\langle \mathcal{T}, \mathcal{A}_2 \rangle)$ if and only if their universal models UI_1 and UI_2 are homomorphically equivalent.

Proof. The first part of the proposition holds by definition of the universal models. The “only if” part of the second part holds because if the inclusion $\text{Mod}(\langle \mathcal{T}, \mathcal{A}_1 \rangle) \subseteq \text{Mod}(\langle \mathcal{T}, \mathcal{A}_2 \rangle)$ holds, then UI_1 is a model for $\langle \mathcal{T}, \mathcal{A}_2 \rangle$. Consequently, by definition of the universal model, there is a homomorphism from UI_2 to UI_1 . To proof the “if” part assume that there is a homomorphism h such that $h : UI_2 \rightarrow UI_1$. Assume also that I is a model for $\langle \mathcal{T}, \mathcal{A}_1 \rangle$. We are to show that I is a model for $\langle \mathcal{T}, \mathcal{A}_2 \rangle$. Because I is a model for $\langle \mathcal{T}, \mathcal{A}_1 \rangle$, the interpretation I satisfies \mathcal{T} . Hence, we remain to show that $I \models \mathcal{A}_2$. Because I is a model for $\langle \mathcal{T}, \mathcal{A}_1 \rangle$, there is a homomorphism $h' : UI_1 \rightarrow I$. Due to the fact that a composition of two homomorphisms is a homomorphism, a mapping $h' \circ h$ is a homomorphism from UI_2 to I . Hence, for every atomic fact $P(\vec{a})$ from \mathcal{A}_2 , we have that $UI_2 \models P(\vec{a})$ and consequently $I \models P(h' \circ h(\vec{a}))$. Note that $P(h' \circ h(\vec{a})) = P(\vec{a})$, because all constants occurring in \vec{a} are from $\text{adom}(\langle \mathcal{T}, \mathcal{A}_2 \rangle)$. Hence, $I \models P(\vec{a})$ for any $P(\vec{a})$ in \mathcal{A}_2 and this implies $I \models \langle \mathcal{T}, \mathcal{A}_2 \rangle$. \square

The first part of Proposition 1 states that universal models are unique up to homomorphic equivalence. The second part implies that if for a given TBox \mathcal{T} , UI is a universal model for two KBs $\langle \mathcal{T}, \mathcal{A}_1 \rangle$ and $\langle \mathcal{T}, \mathcal{A}_2 \rangle$, then $Mod(\langle \mathcal{T}, \mathcal{A}_1 \rangle) = Mod(\langle \mathcal{T}, \mathcal{A}_2 \rangle)$. Thus, in a certain sense, each universal model precisely embodies the space of all models (completions) of a given ABox which are valid wrt to a given TBox.

In the second part of Proposition 1 we consider two KBs with the same TBoxes. If we relax this condition and consider two KBs $\langle \mathcal{T}_1, \mathcal{A}_1 \rangle$ and $\langle \mathcal{T}_2, \mathcal{A}_2 \rangle$ such that $\mathcal{T}_1 \neq \mathcal{T}_2$, then an existence of a homomorphism $h : UI_2 \rightarrow UI_1$ between universal models of these KBs is not enough to conclude that $Mod(\langle \mathcal{T}_1, \mathcal{A}_1 \rangle) \subseteq Mod(\langle \mathcal{T}_2, \mathcal{A}_2 \rangle)$. To illustrate this phenomena, let us consider a counterexample. Let $\mathcal{T}_1 = \emptyset$, $\mathcal{A}_1 = \{A(a)\}$ and $\mathcal{T}_2 = \{A_1 \sqsubseteq A_2\}$, $\mathcal{A}_2 = \{A(a)\}$. Let I be $I(\{A(a)\})$. The interpretation I coincides with $I(A_1)$ and $I(A_2)$ and obviously is a universal model for both KBs. Consider an interpretation $I_1 = I(\{A(a), A_1(a)\})$. It is easy to see that I_1 is a model for the first KB but not for the second one. Hence, $Mod(\langle \mathcal{T}_1, \mathcal{A}_1 \rangle) \not\subseteq Mod(\langle \mathcal{T}_2, \mathcal{A}_2 \rangle)$ despite the fact that there is a homomorphism between universal solutions.

3.3.2 Chase: Canonical Generation of Universal Model

Checking the conditions for a given model to be a universal one (see Definition 8) requires implicitly the ability to check the (infinite) space of all (probably infinite) models. Thus, it is not clear, at first hand, to what extent the notion of universal model is a computable one. This section addresses the question of how to check the existence of a universal model and how to compute one (if one exists). In particular, we show that the classical chase can be adopted for *DL-Lite \mathcal{F}* setting and that every chase, if a given KB is satisfiable, constructs a universal model for the KB. We show that satisfiability can be checked in LOGSPACE in data complexity (in the size of the KB ABox). In general, for arbitrary *DL-Lite \mathcal{F}* KB, there may not exist a finite chase. Hence, in Section ?? we refer to the the class of weakly acyclic positive inclusion assertions, for which, under certain conditions on negative inclusion assertions, the chase is guaranteed to terminate in polynomial time. The class of weakly acyclic positive inclusion assertions is adopted from the [FKMP05], where the assertions are introduced under the name weakly acyclic *tuple-generating-dependencies*. For such class of assertions a universal model (if it exists) can be produced in polynomial time.

Intuitively, for a satisfiable KB we apply the following procedure to produce a universal model: start with a minimal model for the ABox of the KB; then chase the model by applying positive inclusion assertions in the TBox of the KB in some arbitrary order and for as long as they are applicable. This process may terminate (this corresponds to a case when a finite universal model for the KB exists) or it may never terminate (this corresponds to a case when no finite chase exists). In any way the procedure generates a universal model.

Let us first define which *FOL* formulae correspond to *DL-Lite \mathcal{F}* assertions. Any positive inclusion assertion by the definition of its semantics is logically equivalent to a *FOL* formula of the form:

$$\forall \vec{x} (P_1(\vec{x}) \rightarrow \exists \vec{y} P_2(\vec{x}, \vec{y}))$$

where $P_1(\vec{x})$ and $P_2(\vec{x}, \vec{y})$ are atoms, such that the only variables that can occur in them are listed in the vectors \vec{x} and \vec{x}, \vec{y} , respectively; P_1 and P_2 are in $AC \cup AR$; and \vec{x} contains all free variables

occurring in $P_1(\vec{x})$ and $\exists \vec{y} P_2(\vec{x}, \vec{y})$. We say that the shortest implication of this form that is logically equivalent to a given positive inclusion assertion *corresponds* to the assertion. Note that formulae of the form $\forall \vec{x} (P_1(\vec{x}) \rightarrow \exists \vec{y} P_2(\vec{x}, \vec{y}))$ subsume formulae of the form $\forall \vec{x} (\exists \vec{x}' P'_1(\vec{x}, \vec{x}') \rightarrow \exists \vec{y} P_2(\vec{x}, \vec{y}))$, where $P_1(\vec{x})$, $P'_1(\vec{x}, \vec{x}')$ and $P_2(\vec{x}, \vec{y})$ are atoms, because the later implication is logically equivalent to the formula $\forall \vec{x} \forall \vec{x}' (P'_1(\vec{x}, \vec{x}') \rightarrow \exists \vec{y} P_2(\vec{x}, \vec{y}))$. Analogously we define corresponding formulae for negative inclusion and functional assertions. In the following table we present assertions and corresponding formulae.

	Assertion	Formula
(1)	$B \sqsubseteq B'$	$\forall x (B(x) \rightarrow B'(x))$
(2)	$B \sqsubseteq \neg B'$	$\forall x, y (B(x) \wedge B'(y) \rightarrow x \neq y)$
(2')	$B \sqsubseteq \neg B'$	$\forall x (\neg B(x) \vee \neg B'(x))$
(3)	(func R)	$\forall x, y, z (R(x, y) \wedge R(x, z) \rightarrow y = z)$
(4)	(func R^-)	$\forall x, y, z (R(y, x) \wedge R(z, x) \rightarrow y = z)$

where B and B' are basic concepts, that is, they are of the form A , or $\exists R$, or $\exists R^-$. If $B = A$, then $B(x) = A(x)$. If $B = \exists R$ ($B = \exists R^-$), then $B(x) = \exists y R(x, y)$ ($B(x) = \exists y R(y, x)$). In the cases (2) and (2') we present two different possible formulae (i.e. with and without using inequality) that correspond to negative inclusion assertions. As an example consider an inclusion assertion $\exists R \sqsubseteq A$. The corresponding formula to it is $\forall x, y (R(x, y) \rightarrow A(x))$.

Similar to homomorphisms between interpretations, we define a *homomorphism* from a conjunction of atomic formulae to an interpretation in the following way.

Definition 10. Let $\varphi = P_1(\vec{x}_1) \wedge \dots \wedge P_n(\vec{x}_n)$ be a conjunction of atomic formulae, $P_i \in AC \cup AR$. Let I be an interpretation. A homomorphism h from φ to I is a mapping from the variables \vec{x} to $Const$ such that $h(P_i(x_1^i, \dots, x_{n_i}^i)) = P_i(h(x_1^i), \dots, h(x_{n_i}^i))$ is true in I for each conjunct i , where $(x_1^i, \dots, x_{n_i}^i) = \vec{x}_i$.

We next define chase steps. The chase we define is a simplification of the notation proposed in [FKMP05], which is in turn a variation of the classical notion of the chase proposed in [BV84], where that Fagin et al. chase with instances rather than symbolic tableaux. Our simplification is that we eliminate a part of chase step corresponds to the case of so called equality-generating-dependencies. Simplification is possible because positive inclusion assertions are not *key-conflicting*, speaking in terms of [CCDGL02]. We will discuss this phenomena in details later on, after the proof of the Separation Theorem (see ??).

Definition 11. (Chase step) Let \mathcal{K} be a KB and I be an interpretation. Let $\forall \vec{x} (P_1(\vec{x}) \rightarrow \exists \vec{y} P_2(\vec{x}, \vec{y}))$ be a formula corresponding to a positive inclusion assertion e in \mathcal{K} .

Let h be a homomorphism from $P_1(\vec{x})$ to I such that there is no extension of h to a homomorphism h' from $P_1(\vec{x}) \wedge P_2(\vec{x}, \vec{y})$ to I . We say that e can be applied to I with the homomorphism h .

We extend h to h' by assigning for each variable in \vec{y} a fresh labelled null for I (which is not in $ln(\mathcal{K}, I)$). We extend I to I' by assigning the image of $P_2(\vec{x}, \vec{y})$ under h' to true. We say that I' is the result of applying e to I with h over \mathcal{K} , and denote $I \xrightarrow{e, h, \mathcal{K}} I'$.

A chase step is exactly $I \xrightarrow{e, h, \mathcal{K}} I'$ as it is defined above. We will omit an indication of the

KB over which a chase step is applied if the KB is clear from the context. Observe that the result of application of an inclusion assertion to an interpretation with a homomorphism is defined non-deterministically, in the sense, that we do not fix a labelled null to be chosen. Observe also that if I is a model for a KB \mathcal{K} , then non of the inclusion assertion of \mathcal{K} (if there are) can be applied to I .

Consider an inclusion assertion $e = B \sqsubseteq B'$ and an interpretation I . The fact that e can be applied to I with some homomorphism h (over some KB \mathcal{K}) essentially means that there is a constant $a \in \text{Const} \setminus \text{In}(\mathcal{K}, I)$ such that $I \models B(a)$, but $I \not\models B'(a)$. The result of applying e to I with h is just an extension of I to I' by assigning $I' \models B'(a)$. Due to this fact we say that the result applying e to I with h is a *propagation* of the constant a from B to B' for the interpretation I . We also say that the *propagation extends* I to I' . If B' has the corresponding formula of the form $A(x)$, then the propagation of a is just an assignment of $A(a)$ to true in I' . If B' has the corresponding formula of the form $\exists y R(x, y)$ or $\exists y R(y, x)$, then the propagation, i.e. the assignment $I' \models B'(a)$, is the assignment $I' \models \exists y R(a, y)$ or $I' \models \exists y R(y, a)$, respectively. Hence to satisfy $I' \models B'(a)$ we should invent a constant (fresh labelled null) for the existentially quantified variable y , say n , such that that $I' \models R(a, n)$ or $I' \models R(n, a)$. The invented labelled null is essentially a *Skolem constant*, namely $n = f(a)$, where f is a function that reflects a dependency between a and n .

Example 4. Let us consider an example to illustrate the introduced notions. Let \mathcal{K} be a KB with the following TBox and ABox:

$$\mathcal{T} = \{\exists R^- \sqsubseteq \exists R\}; \quad \mathcal{A} = \{R(a, b)\}$$

Let $e = \exists R^- \sqsubseteq \exists R$. The formula corresponding to e is $\forall x, y (R(x, y) \rightarrow \exists z R(y, z))$. The formula intuitively states that for any object of the domain if there is an incoming R -edge in it, then there is an outgoing R -edge from it. Let an interpretation I be $I = I(\{R(a, b)\})$. By the definition of I , the entailment $I \models R(a, b)$ holds. Hence the inclusion assertion e can be applied to I with a homomorphism $h = \{x \mapsto a, y \mapsto b\}$. Indeed, h maps $R(x, y)$ to a true atom $R(a, b)$ for I , and there is obviously no extension of h that maps the conjunction $R(x, y) \wedge R(y, z)$ to I . A result of the application of e to I with h (over \mathcal{K}) is I' , such that I' extends I by assigning $R(b, c)$ to true, where c is a fresh labelled null for I .

Let us now consider the application of e to I with h in terms of propagation. To do this let B and B' denote the concepts $\exists R^-$ and $\exists R$, respectively. In this denotation the assertion e is equal to $B \sqsubseteq B'$. By the definition of I , the entailments $I \models B(a)$ and $I \not\models B'(a)$ hold. Consequently, we can propagate a constant a from B to B' for I . The propagation of a from B to B' extends I to I'

Now we are ready to define the chase.

Definition 12. Let \mathcal{K} be a KB. A chase sequence of I over \mathcal{K} is a (finite or infinite) sequence of chase steps $I_i \xrightarrow{e_i h_i \mathcal{K}} I_{i+1}$ for $i = 0, 1, \dots$, where e_i is a positive inclusion assertion of \mathcal{K} , h_i is a homomorphism and $I = I_0$.

We say that the chase sequence starts with I_0 . If the chase sequence is finite, that is, if $i \leq k$, then we say that the sequence ends with the interpretation I_k and that I_k is obtained from I in k steps by chasing I with \mathcal{K} . We also say that the chase sequence is of depth k .

We call *finite chase* (of I with \mathcal{K}) a finite chase sequence $I_i \xrightarrow{e_i h_i \mathcal{K}} I_{i+1}$ with $0 \leq i < m$ and $I_0 = I$, for which there is no positive inclusion assertion e in \mathcal{K} and homomorphism h , such that e can be applied to I_m with h . We denote I_m as $\text{chase}^m(I, \mathcal{K})$. We call *infinite chase* a chase sequence which is not finite. Note that to ensure that all relevant positive inclusion assertions have a chance to influence a chasing sequence, we focus on chasing sequences that satisfy the following conditions:

- No positive inclusion assertion is “missed” (i.e., each assertion that can be applied is applied).
- No positive inclusion assertion is “starved” (i.e., each assertion that is applicable infinitely often is applied infinitely often).

We say that infinite chase is of infinite depth. We denote an infinite chase of I with \mathcal{K} as $\text{chase}^\infty(I, \mathcal{K})$. Alternatively, for an infinite chase sequence $I_i \xrightarrow{e_i h_i \mathcal{K}} I_{i+1}$, where $i \geq 0$ and $I_0 = I$ we define an infinite chase as a minimal model for the set of facts:

$$\{P(\vec{c}) \mid \exists m I_m \models P(\vec{c})\}.$$

This set of facts (by its definition) is the same as:

$$\{P(\vec{c}) \mid \exists n \forall m > n I_m \models P(\vec{c})\}.$$

If there is no need to underline (infinite) depth of the chase, then we use a term *chase* for finite or infinite chase and denotation $\text{chase}(I, \mathcal{K})$ for $\text{chase}^m(I, \mathcal{K})$ or $\text{chase}^\infty(I, \mathcal{K})$. For a given KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ we distinguish a specific case of the chased interpretation I , namely, when $I = I(\mathcal{A})$. If there exists a finite chase of $I(\mathcal{A})$ with \mathcal{K} of depth m , then we denote $\text{chase}^m(I(\mathcal{A}), \mathcal{K})$ as $\text{chase}^m(\mathcal{K})$, otherwise we denote an infinite chase of $I(\mathcal{A})$ with \mathcal{K} as $\text{chase}^\infty(\mathcal{K})$; if there is no need to underline the (infinite) depth of the chase we denote $\text{chase}^m(\mathcal{K})$ or $\text{chase}^\infty(\mathcal{K})$ as $\text{chase}(\mathcal{K})$.

3.3.3 Properties of Chase Sequences

Let us consider interesting properties of the chase sequences that are obtained, for a given KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, by chasing $I(\mathcal{A})$ with the positive inclusion assertions of \mathcal{T} . The first lemma shows that if all functional assertions of \mathcal{T} are satisfied by $I(\mathcal{A})$, then non of the chase sequences that starts with $I(\mathcal{A})$ ends with an interpretation that violates the functional assertions.

Lemma 1. *Given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. If $I(\mathcal{A})$ satisfies all functional assertions of \mathcal{T} , then for any chase sequence of $I(\mathcal{A})$ over \mathcal{K} of depth m that ends with an interpretation I' , it holds that I' satisfies all functional assertions in \mathcal{K} , where the constant m satisfies the following conditions. If there exists a finite chase(\mathcal{K}) of depth t , then $m \leq t$, otherwise $m \geq 0$.*

Proof. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a KB; $I(\mathcal{A})$ satisfies all functional assertions of \mathcal{T} . Let $I_i \xrightarrow{e_i h_i \mathcal{K}} I_{i+1}$ be a chase sequence of $I(\mathcal{A})$ (i.e. $I_0 = I(\mathcal{A})$) over \mathcal{K} of depth m , where $m \leq t$ if there is a finite chase(\mathcal{K}) of depth t , and $i \geq 0$ otherwise. Let us denote $I_{m-1} \xrightarrow{e_{m-1} h_{m-1} \mathcal{K}} I_m$, the last chase step in the chase sequence, as $I \xrightarrow{e h \mathcal{K}} I'$. The interpretation I' in the chase step is the result of applying the inclusion assertion e to I with the homomorphism h over \mathcal{K} . Let $\forall \vec{x} (P(\vec{x}) \rightarrow \exists \vec{y} P'(\vec{x}, \vec{y}))$ be the

formula corresponding to e . A homomorphism h maps \vec{x} to \vec{a} in such a way that $P(h(\vec{x})) = P(\vec{a})$ and $I \models P(\vec{a})$. An extension h' of h is a homomorphism that maps \vec{y} to \vec{b} in such a way that $P'(h(\vec{x}, \vec{y})) = P'(\vec{a}, \vec{b})$ and $I' \models P'(\vec{a}, \vec{b})$. We are to show that I' does not violate non of the functional assertions of \mathcal{K} . We will prove it by induction on m , that is, on the depth of the chase sequence. We will consider tree induction proofs, each for one possible form of the atom $P'(\vec{a}, \vec{b})$.

By the definition of the basic concepts the atom $P'(\vec{a}, \vec{b})$ has one of the following forms:

- (1) $R(l, n)$ (3) $A(l)$
 (2) $R(n, l)$

where $l \in \text{ln}(\mathcal{K}, I) \setminus \text{adom}(\mathcal{K})$ and n is a fresh labelled null for I , that is, $n \in \text{ln}(\mathcal{K}, I)$.

As it was discussed before, any functional assertion has a corresponding formula. We recall the correspondence in the following table:

	Assertion	Formula
(a)	$(\text{func } R)$	$\forall x, y, z (R(x, y) \wedge R(x, z) \rightarrow y = z)$
(b)	$(\text{func } R^-)$	$\forall x, y, z (R(y, x) \wedge R(z, x) \rightarrow y = z)$

The inductive proofs are the following.

- (1) Assume that $P'(\vec{a}, \vec{b})$ is of the form $R(l, n)$.

Induction base. Let $m = 1$, hence, $I_{m-1} = I_0 = I(A) = I$, a propagated constant $l \in \text{adom}(\mathcal{K})$ and $I_m = I_1 = I'$. The interpretation I' , by the definition of the chase step, is different from $I(A)$ only in the interpretation of $R(l, n)$.

- (a). Assume that the functional assertion of the type (a) is violated by I' , that is, there exists a fact $R(l, n')$, where n' is a constant different from n , such that $I' \models R(l, n')$ and $I' \models R(l, n)$. By the definition of the chase step, $I' \models R(l, n')$ implies $I \models R(l, n')$, hence, $n' \in \text{adom}(\mathcal{K})$. The fact $I \models R(l, n')$ contradicts with applicability of e to I with h . The reason is that from $I \models R(l, n')$ we conclude existence of an extension of h to $h' \supseteq \{u \mapsto l, v \mapsto n'\}$ that maps $P(\vec{x}) \wedge P'(\vec{x}, \vec{y}) = P(\vec{x}) \wedge R(u, v)$ to I . Existence of h' contradicts with the definition of applicability of e to I with h . Hence, the assertion of the type (a) is not violated in I' .
- (b). Assume that the functional assertion of the type (b) is violated by I' , that is, there exists a fact $R(l', n)$, where l' is a constant different from l , such that $I' \models R(l', n)$ and $I' \models R(l, n)$. By the definition of the chase step, $I' \models R(l', n)$ implies $I \models R(l', n)$. Hence, n is a labelled null that is not fresh for I , that is $n \notin \text{ln}(\mathcal{K}, I)$. This contradicts with the definition of the chase step. Hence, the assertion of the type (b) is not violated in I' .

Induction step. Assume that for all $m < k$ the current Lemma holds. Let $m = k$, hence, $I_{m-1} = I_{k-1} = I$ and, by the assumption, I does not violate any of the functional assertions of \mathcal{K} . We are to show that $I' = I_m = I_k$ does not violate the functional assertions of \mathcal{K} neither of the type (a) nor of the type (b).

- (a). To show that I' does not violate functional assertions of the type (a) one can change $n' \in \text{adom}(\mathcal{K})$ to $n' \in \text{adom}(\mathcal{K}) \cup \text{In}(\mathcal{K}, I)$ in the proof (a) of the *base case* and then reuse the proof.
- (b). To show that I' does not violate functional assertions of the type (b) one can simply reuse the proof (b) of the *base case*.

We conclude that I' does not violate any functional assertion neither of the type (a) nor of the type (b).

- (2) Assume that $P'(\vec{a}, \vec{b})$ is of the form $R(n, l)$. The proof that I' satisfies all inclusion assertions of \mathcal{K} is similar to the case (1).
- (3) Assume that $P'(\vec{a}, \vec{b})$ is of the form $A(l)$. The following inductive proof covers functional assertions of both types (a) and (b) at once.

Induction base. Let $m = 1$, hence, $I_{m-1} = I_0 = I(A) = I$ and $I_m = I_1 = I'$. The interpretation I' , by definition of the chase step, is different from $I(\mathcal{A})$ only in the interpretation of $A(l)$. Hence, if I' violates a functional assertion of \mathcal{K} of the type either (a) or (b), then the assertion is already violated in $I(\mathcal{A})$. This contradicts with the fact that $I(\mathcal{A})$ satisfies all functional assertions of \mathcal{K} .

Induction step. Assume that for all $m < k$ the lemma holds. Let $m = k$, hence, $I_{m-1} = I_{k-1} = I$ and, by the assumption, I does not violate any of the functional assertions of \mathcal{K} . Assume I' violates a functional assertion of the type either (a) or (b). The interpretation I' is different from I only in the interpretation of $A(l)$. Hence, then the functional assertion is already violated in I . We obtain a contradiction with the assumption of the induction step.

□

Unfortunately, similar result does not hold for negative inclusion assertions, that is, for a given KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ it is not enough to check that all negative inclusion assertions of \mathcal{K} are satisfied on $I(\mathcal{A})$ to be sure that non of the chase sequences that starts with $I(\mathcal{A})$ ends with an interpretation that violates the negative inclusion assertions. Consider an example that illustrates this phenomena.

Example 5. Consider the following KB:

$$\mathcal{T} = \left\{ \begin{array}{ll} \text{(i)} A \sqsubseteq B & \text{(ii)} B \sqsubseteq B' \\ \text{(iii)} B \sqsubseteq B'' & \text{(iv)} B' \sqsubseteq \neg B'' \end{array} \right\} \quad \mathcal{A} = \{A(a)\}$$

Observe that $I(\mathcal{A}) = I(\{A(a)\})$ does not violate $B' \sqsubseteq \neg B''$, the only negative inclusion assertion of \mathcal{T} . Let us consider the following chase sequence of depth three. We start the sequens with $I_0 = I(\{A(a)\})$. The result of the first chase step (with the assertion (i)) is a propagation of a from A to B and an interpretation $I_1 = I(\{A(a), B(a)\})$. The result of the second chase step (with the assertion (ii)) is a propagation of a from B to B' and an interpretation $I_2 = I(\{A(a), B(a), B'(a)\})$. The result of the third chase step (with the assertion (iii)) is a propagation of a from B to B'' and the interpretation $I_2 = I(\{A(a), B(a), B'(a), B''(a)\})$. We obtain that $I_2 \models B'(a)$, $I_2 \models B''(a)$, consequently, $I_2 \not\models B' \sqsubseteq \neg B''$. This means that we start with $I(\mathcal{A})$, an interpretation that does not

violate the negative inclusion assertion, and in three chase steps extend it to an interpretation that does violate. At the same time, if one look carefully on the TBox \mathcal{T} , one can see that $\mathcal{T} \models A \sqsubseteq \neg A$, and the entailed negative inclusion assertion is not satisfied by $I(\mathcal{A})$. Actually, it turns out (by the following second lemma) that checking that the minimal model of the KB's ABox satisfies all negative inclusion assertions entailed by a KB, is a sufficient condition that guaranties that stepwise extension of the minimal model by chasing it with the KB's TBox preserves satisfiability of all negative inclusion assertions of the KB (actually, even entailed by the KB).

Lemma 2. *Given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. If $I(\mathcal{A})$ satisfies all negative inclusion assertions that are entailed by \mathcal{T} , then for any chase sequence of $I(\mathcal{A})$ over \mathcal{K} of depth m that ends with an interpretation I' , it holds that I' satisfies all negative inclusion assertions entailed by \mathcal{K} , where the constant m satisfies the following conditions. If there exists a finite chase(\mathcal{K}) of depth t , then $m \leq t$, otherwise $m \geq 0$.*

Proof. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a KB; $I(\mathcal{A})$ satisfies all negative inclusion assertions entailed by \mathcal{T} . Let $I_i \xrightarrow{e_i \ h_i \ \mathcal{K}} I_{i+1}$ be a chase sequence of $I(\mathcal{A})$ (i.e. $I_0 = I(\mathcal{A})$) over \mathcal{K} of depth m , where $m \leq t$ if there is a finite chase(\mathcal{K}) of depth t , and $i \geq 0$ otherwise. Let us denote $I_{m-1} \xrightarrow{e_{m-1} \ h_{m-1} \ \mathcal{K}} I_m$, the last chase step in the chase sequence, as $I \xrightarrow{e \ h \ \mathcal{K}} I'$. The interpretation I' in the chase step is the result of application of the inclusion assertion $e = \tilde{B} \sqsubseteq B$ to I with h over \mathcal{K} . Let us denote as $B(l)$ an atomic fact which, by assigning it to true, extends the interpretation I to I' . As one can see from the proof of Lemma 1, $B(l)$ is of the form $R(l, n)$, or $R(n, l)$, or $A(l)$, where $l \in \text{ln}(\mathcal{K}, I) \setminus \text{adom}(\mathcal{K})$ and n is a fresh labelled null for I , that is, $n \in \text{ln}(\mathcal{K}, I)$. We do not explicitly mention the fresh labelled null n in the denotation $B(l)$ because the name of the null is irrelevant for the proof of the current Lemma. We are to show that I' does not violate non of the negative inclusion dependencies entailed by \mathcal{K} . We will prove it by induction on m , that is, on the depth of the chase sequence.

As it was discussed before, any negative inclusion assertion has a corresponding formula. We recall the correspondence in the following table:

	Assertion	Formula
(a)	$B \sqsubseteq \neg B'$	$\forall x (\neg B(x) \vee \neg B'(x))$
(b)	$B' \sqsubseteq \neg B$	$\forall x (\neg B'(x) \vee \neg B(x))$

Using this table we make an inductive proof of the lemma as follows:

Induction base. Let $m = 1$, hence, $I_{m-1} = I_0 = I(\mathcal{A}) = I$ and $I_m = I_1 = I'$. By the condition of the theorem I satisfies all negative inclusion assertions entailed by \mathcal{K} . By the definition of a chase step, $I \models \tilde{B}(l)$ and $I' \models B(l)$

- (a). Assume that the negative inclusion assertion of the type (a) entailed by \mathcal{K} , i.e. $\mathcal{K} \models B \sqsubseteq \neg B'$, is violated by I' , that is, $I' \models B'(l)$ and $I' \models B(l)$. The interpretation I' , by definition of the chase step, is different from I only in the interpretation of $B(l)$, hence, $I \models B'(l)$. Using the definition of the assertions one can show a trivial entailment:

$$(\tilde{B} \sqsubseteq B) \wedge (B \sqsubseteq \neg B') \models (\tilde{B} \sqsubseteq \neg B'), \quad (3.1)$$

where the conjunction of the \mathcal{FOL} formulae (corresponding to the assertions occurring on the left hand side of the entailment) entails the formula (corresponding to the assertion on the right hand side). Using the entailment (3.1) and the facts that \mathcal{K} entails both formulae in the conjunction, i.e. $\mathcal{K} \models \tilde{B} \sqsubseteq B$ and $\mathcal{K} \models B \sqsubseteq \neg B'$, we obtain the following entailment:

$$\mathcal{K} \models (\tilde{B} \sqsubseteq \neg B')$$

From this entailment and the facts that $I \models \tilde{B}(l)$ and $I \models B'(l)$ we conclude that I does not satisfy all the negative inclusion assertions entailed by \mathcal{K} . This contradicts with the fact that I does satisfy them. Hence, the assertion of the type (a) is not violated in I' .

- (b). The proof of the fact that non of negative inclusion assertions of type (b) entailed by \mathcal{K} , i.e. $\mathcal{K} \models B' \sqsubseteq \neg B$, can be violated by I' is similar to the one in the case (a). The only difference is that one need to use an entailment:

$$(\tilde{B} \sqsubseteq B) \wedge (B' \sqsubseteq \neg B) \models (\tilde{B} \sqsubseteq \neg B')$$

instead of:

$$(\tilde{B} \sqsubseteq B) \wedge (B \sqsubseteq \neg B') \models (\tilde{B} \sqsubseteq \neg B')$$

Induction step. Assume that for all $m < k$ the lemma holds. Let $m = k$, hence, $I_{m-1} = I_{k-1} = I$ and, by the assumption, I does not violate all the negative inclusion assertions entailed by \mathcal{K} . We are to show that $I' = I_m = I_k$ does not violate them as well. One again need to consider two types of negative inclusion assertions, namely (a) and (b). For each cases the proof is exactly the same as in the corresponding (a) and (b) cases in the *induction base*.

□

3.3.4 Separation Theorem

In the previous section we investigated that, under certain conditions on the minimal model $I(\mathcal{A})$ of the ABox of a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, chasing of $I(\mathcal{A})$ with \mathcal{T} does not lead to models that violate any negative inclusion or functional assertion of \mathcal{K} . We recall that these conditions on $I(\mathcal{A})$ are that $I(\mathcal{A})$ should satisfy all functional assertions of \mathcal{T} and negative inclusion assertions entailed by \mathcal{K} . In this section we show that this conditions are necessary and sufficient to guaranty satisfiability of a KB. First we show the necessity and then the sufficiency of the conditions.

Theorem 1. . *Given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. If $I(\mathcal{A})$ satisfies all functional assertions of \mathcal{T} and all negative inclusion assertions that are entailed by \mathcal{T} , then \mathcal{K} is satisfiable.*

Proof. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and $I(\mathcal{A})$ satisfies the conditions of the theorem. We will show that $chase(\mathcal{K})$ is a model of \mathcal{K} . There are two possibilities to be considered: first when $chase(\mathcal{K})$ is finite and second when $chase(\mathcal{K})$ is infinite.

Assume $chase(\mathcal{K})$ is finite of depth m and ends with I . Using Lemma 1 we obtain that I satisfies all functional assertions of \mathcal{K} . Using Lemma 2 we obtain that I satisfies all negative inclusion assertions of \mathcal{K} (in fact I satisfies not only negative inclusions that are in \mathcal{K} , but also

negative inclusions entailed by \mathcal{K}). It remains to show that \mathcal{K} satisfies all positive inclusion assertions of \mathcal{K} . Let e be a positive inclusion assertion in \mathcal{K} and $\forall \vec{x}(P(\vec{x}) \rightarrow \exists \vec{y}P'(\vec{x}, \vec{y}))$ be the formula corresponding to e . Assume I does not satisfy e , or, equivalently, $I \not\models \forall \vec{x}(P(\vec{x}) \rightarrow \exists \vec{y}P'(\vec{x}, \vec{y}))$. From this fact we conclude that for any homomorphism $h = \{\vec{x} \mapsto \vec{a}\}$, such that $I \models P(\vec{a})$ there is no extension of h to h' , such that $I \models h'(P'(\vec{x}, \vec{y}))$. Hence, e is applicable to I . The applicability contradicts with the definition of the finite chase. More precisely, it contradicts with the fact that the finite $\text{chase}(\mathcal{K})$ ends with I . Hence, I is a model of \mathcal{K} .

Assume $\text{chase}(\mathcal{K})$ is infinite and $I = \text{chase}^\infty(\mathcal{K})$. Using Lemma 1 and Lemma 2 we obtain that I satisfies all functional and negative inclusion assertions of \mathcal{K} , respectively. It remains to show that \mathcal{K} satisfies all positive inclusion assertions of \mathcal{K} . Let e be a positive inclusion assertion in \mathcal{K} and $\forall \vec{x}(P(\vec{x}) \rightarrow \exists \vec{y}P'(\vec{x}, \vec{y}))$ be the formula corresponding to e . Assume I does not satisfy e , or, equivalently, $I \not\models \forall \vec{x}(P(\vec{x}) \rightarrow \exists \vec{y}P'(\vec{x}, \vec{y}))$. From this fact we conclude that for any homomorphism $h = \{\vec{x} \mapsto \vec{a}\}$, such that $I \models P(\vec{a})$, there is no extension of h to h' , such that $I \models h'(P'(\vec{x}, \vec{y}))$. Hence, e is applicable to I . By the definition of the infinite chase, non of the positive inclusion assertions is neither “missed” nor “starved”, hence, e was (again) applied at some step, say at the step number n , during the construction of the infinite chase $\text{chase}(\mathcal{K})$. By the definition of the chase step, the result of the step n is an interpretation I_n , such that $I_n \models h'(P'(\vec{x}, \vec{y}))$, where h' is an extension of h . By the definition of the infinite chase:

$$I = I(\{P(\vec{c}) \mid \exists m I_m \models P(\vec{c})\}).$$

Hence, from $I_n \models h'(P'(\vec{x}, \vec{y}))$ we conclude $I \models h'(P'(\vec{x}, \vec{y}))$. It contradicts with the assumption that I does not satisfy e . Hence, I is a model of \mathcal{K} . □

There is an important corollary from Theorem 1.

Corollary 1. *Given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. If $I(\mathcal{A})$ satisfies all functional assertions of \mathcal{T} and all negative inclusion assertions that are entailed by \mathcal{T} , then $\text{chase}(\mathcal{K})$ is a model of \mathcal{K} .*

The proof of the Corollary is exactly the one of Theorem 1. The result of Theorem 1 can be easily extended in the following way.

Corollary 2. *Given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. If an interpretation I satisfies \mathcal{A} , all functional assertions of \mathcal{T} and all negative inclusion assertions that are entailed by \mathcal{T} , then $\text{chase}(I, \mathcal{K})$ is a model of \mathcal{K} .*

Proof. The proof is similar to the one for Theorem 1. □

Now we will show that satisfiability on a KB implies that the minimal model of it's ABox satisfies all functional assertions of the KB and negative inclusion assertions entailed by the KB. Thus, we will show the conditions on the minimal model are sufficient to guaranty satisfiability of the KB.

Theorem 2. *If a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, then $I(\mathcal{A})$ satisfies all functional assertions of \mathcal{T} and all negative inclusion assertions that are entailed by \mathcal{T} .*

Proof. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a satisfiable KB and I be its model. Since I satisfies \mathcal{A} we have that there is a homomorphism $h : I(\mathcal{A}) \rightarrow I$.

We recall correspondence between functional, negative inclusion assertions and \mathcal{FOC} formulae in the following table:

	Assertion	Formula
(a)	$B \sqsubseteq \neg B'$	$\forall x (\neg B(x) \vee \neg B'(x))$
(b)	$(\text{func } R)$	$\forall x, y, z (R(x, y) \wedge R(x, z) \rightarrow y = z)$
(c)	$(\text{func } R^-)$	$\forall x, y, z (R(y, x) \wedge R(z, x) \rightarrow y = z)$

We prove by contradiction that non of the assertions of the type (a)-(c) is violated by $I(\mathcal{A})$.

- (a). Let $B \sqsubseteq \neg B'$ be a negative inclusion assertion entailed by \mathcal{K} . Since $I \models \mathcal{K}$, we obtain $I \models B \sqsubseteq \neg B'$. Assume the assertion is violated by $I(\mathcal{A})$, i.e. $I(\mathcal{A}) \not\models \forall x (\neg B(x) \vee \neg B'(x))$. Thus, there is a constant $a \in \text{adom}(\mathcal{K})$, such that $I(\mathcal{A}) \models B(a)$ and $I(\mathcal{A}) \models B'(a)$. By the definition of h , entailments $I \models B(h(a))$ and $I \models B'(h(a))$ hold. Since h is a homomorphism, the equality $h(a) = a$ holds. We conclude that $I \models B(a)$ and $I \models B'(a)$ hold. This contradicts with the fact that $I \models B \sqsubseteq \neg B'$.
- (b). Let $(\text{func } R)$ be a functional assertion, such that $I \models (\text{func } R)$, but $I(\mathcal{A}) \not\models (\text{func } R)$. From the latter entailment we conclude that there exist two ground atoms $R(a, b)$ and $R(a, c)$, where $a, b, c \in \text{adom}(\mathcal{K})$, such that $I(\mathcal{A}) \models R(a, b)$ and $I(\mathcal{A}) \models R(a, c)$. By the definition of h , entailments $I \models R(h(a), h(b))$ and $I \models R(h(a), h(c))$ hold. Since $h(a) = a$, $h(b) = b$ and $h(c) = c$ hold, we conclude that entailments $I \models R(a, b)$ and $I \models R(a, c)$ hold. This contradicts with the fact that $I \models (\text{func } R)$.
- (c). The proof of the case when a functional assertion of the type $(\text{func } R^-)$ is violated by $I(\mathcal{A})$ is similar to the case (b).

□

Theorems 1 and 2 gives us a very important property of $DL\text{-Lite}_{\mathcal{F}}$ KBs, namely, a so-called *Separation Property*.

Theorem 3. (*Separation Theorem*)

A KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable if and only if $I(\mathcal{A})$ satisfies all functional assertions of \mathcal{T} and all negative inclusion assertions that are entailed by \mathcal{T} .

Theorem 3 and the property are called “separation” because, from the *query answering* point of view, dealing with a given KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ can be *separated* in two stages. First, one check that $I(\mathcal{A})$ satisfies negative inclusion assertions entailed by \mathcal{K} and functional assertions of \mathcal{K} . Second, if $I(\mathcal{A})$ does so, one can forget about both negative inclusion and functional assertion and use for (conjunctive) query answering positive inclusion assertions only. We will discuss an impact of the Separation Theorem on query answering in details in the Section 4.3.6.

3.3.5 Chase is a Universal Model

Next theorem is the main result in the current section. The theorem relates two basic notions of the section: universal model and chase.

Theorem 4. *If a KB \mathcal{K} is satisfiable, then $\text{chase}(\mathcal{K})$ is a universal model for \mathcal{K} .*

To proof this theorem we will first show the following proposition.

Proposition 2. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a knowledge base and $I_1 \xrightarrow{e, h, \mathcal{K}} I_2$ be a chase step, where I_1 and I_2 are models of \mathcal{A} . Let I be an interpretation such that (a) $I \models \mathcal{K}$ and (b) there exists a homomorphism $h_1 : I_1 \rightarrow I$. Then there exists a homomorphism $h_2 : I_2 \rightarrow I$.*

Proof. Let $\forall \vec{x} (P(\vec{x}) \rightarrow \exists \vec{y} P'(\vec{x}, \vec{y}))$ be the corresponding to e formula. Let for each variable y in \vec{y} denote as $n_y \in \text{ln}(\mathcal{K}, I_1)$ a labelled null that replaces y after the application of an extension of h in the chase step. A vector \vec{n}_y denotes the image of the vector \vec{y} under the extension of h . From two facts:

- (1) I_2 is different from I_1 in the interpretation of $P'(h(\vec{x}), \vec{n}_y)$ only and
- (2) there is a homomorphism $h_1 : I_1 \rightarrow I$,

we conclude that, in order to obtain a homomorphism $h_2 : I_2 \rightarrow I$, we need to extend h_1 to h_2 by defining it on constants in $h(\vec{x})$ (which are in $\text{Const} \setminus \text{ln}(\mathcal{K}, I_1)$) and \vec{n}_y (which are in $\text{ln}(\mathcal{K}, I_1)$) only.

By the definition of the chase step, the mapping $h : P(\vec{x}) \rightarrow I_1$ is a homomorphism. Composition of homomorphisms is a homomorphism, hence, the following composition:

$$h_1 \circ h : P(\vec{x}) \rightarrow I$$

is a homomorphism. Since $I \models \mathcal{K}$, there exists a homomorphism:

$$h' : P(\vec{x}) \wedge P'(\vec{x}, \vec{y}) \rightarrow I,$$

such that h' extends $h_1 \circ h$ on the variables in \vec{y} , i.e. $h'(\vec{x}) = (h_1 \circ h)(\vec{x})$.

Let us define a mapping h_2 from Const to Const as follows:

$$\begin{aligned} h_2(c) &= h_1(c) \quad \text{where } c \in \text{Const} \setminus \text{ln}(\mathcal{K}, I_1), \\ h_2(n_y) &= h'(y) \quad \text{where } n_y \in \text{ln}(\mathcal{K}, I_1) \text{ is a constant defined above.} \end{aligned}$$

Now we need to show that h_2 is a homomorphism from I_2 to I , i.e. that h_2 maps each ground atom which is true in I_2 to a corresponding (true) ground atom in I . For ground atoms which are true in I_2 and I_1 at once the corresponding atoms are true in I , because h_1 is a homomorphism. We only need to show that the corresponding atom for $P'(h(\vec{x}), \vec{n}_y)$ is true in I , namely $I \models P'(h_2(h(\vec{x})), h_2(\vec{n}_y))$. By the definition of h_2 and h' , we have $h_2(h(\vec{x})) = h_1(h(\vec{x})) = h'(\vec{x})$ and $h_2(\vec{n}_y) = h'(\vec{y})$. Hence, $P'(h_2(h(\vec{x})), h_2(\vec{n}_y)) = P'(h'(\vec{x}), h'(\vec{y}))$ and, by the definition of h' , the entailment $I \models P'(h'(\vec{x}), h'(\vec{y}))$ holds. We conclude that $I \models P'(h_2(h(\vec{x})), h_2(\vec{n}_y))$ and h_2 is a homomorphism from I_2 to I . □

The proof of Theorem 4 is based on Proposition 2 and the fact that for any KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ the identity mapping is a homomorphism from $I(\mathcal{A})$ to any model of \mathcal{K} . The details of the proof are the following.

Proof. (of Theorem 4)

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a satisfiable KB. As it follows from the Corollaries ?? and 1, $chase(\mathcal{K})$ is a model for \mathcal{K} . It remains to show, that $chase(\mathcal{K})$ can be homomorphically mapped in any model of \mathcal{K} , say I' . We distinguish two cases, namely when chase is finite and when it is infinite. In both cases we use the fact that the identity function on the constants from $adom(\mathcal{K})$, is a homomorphism $id : I(\mathcal{A}) \rightarrow I'$ from $I(\mathcal{A})$ to I' .

Let $chase(\mathcal{K})$ be a finite chase that ends with I . Then by applying Proposition 2 to each chase step of the finite chase we conclude that there is a homomorphism $h : I \rightarrow I'$.

Let $chase(\mathcal{K})$ be an infinite chase and $I = chase^\infty(\mathcal{K})$. We define a mapping h from $Const$ to $Const$ as the following function:

- (i) $h(a) = id(a)$, where $a \in adom(\mathcal{K})$;
- (ii) $h(b) = h_n(b)$, where
 - n is the depth of a finite initial fragment (chase subsequence) of $chase(\mathcal{K})$ (that starts with $I(\mathcal{A})$ and ends with I_n);
 - $b \in Const \setminus ln(\mathcal{K}, I_n)$;
 - $h_n : I_n \rightarrow I'$ is a homomorphism.

We are to show that h is a homomorphism from I to I' . By the definition of the infinite chase,

$$I = I(\{P(\vec{c}) \mid \exists n I_n \models P(\vec{c})\}).$$

Thus, if there is a ground atom $P(\vec{c})$, such that $I \models P(\vec{c})$, then there is a natural number n and a finite initial fragment of $chase(\mathcal{K})$ of depth n that ends with I_n , such that $I_n \models P(\vec{c})$ and, by definition of $ln(\mathcal{K}, I_n)$, all constants from \vec{c} are in $Const \setminus ln(\mathcal{K}, I_n)$. Hence, applying Proposition 2 to each chase step of the initial fragment, we conclude that there is a homomorphism $h_n : I_n \rightarrow I'$, such that $I' \models P(h_n(\vec{c}))$. By definition of h we have $h(\vec{c}) = h_n(\vec{c})$, thus, $I' \models P(h(\vec{c}))$. We conclude that h is a homomorphism from I to I' . □

Theorem 4 shows that any satisfiable *DL-Lite*_ℱ KB meets a universal model property, that is, it has a universal model. The next theorem shows that the universal model is unique (up to isomorphism).

Theorem 5. *For any satisfiable \mathcal{K} $chase(\mathcal{K})$ is unique up to homomorphism equivalence.*

Proof. The statement of the theorem follows from the fact that any $chase(\mathcal{K})$ is a universal model (Theorem 4) and the fact that all universal models are homomorphically equivalent (Proposition 1). □

The uniqueness of chase shows that applications of positive inclusions assertions are confluent, in the sense, that it does not matter from which assertion one starts chasing the minimal model of

the KB's ABox and in which order the assertions are applied, in any way, one will always ends with the chase that is homomorphically equivalent to any other chase.

Chapter 4

Conjunctive Query Answering for DL-Lite \mathcal{F}

In this chapter we focus on a specific kind of query answering for DL-Lite \mathcal{F} logic, namely, on *conjunctive query* (CQ) answering. We first define which queries are conjunctive, then we investigate two orthogonal approaches to answer such queries. The first approach is to perform *model checking* (model theoretical approach) and the second one is to perform *theorem proving* (proof theoretical approach). For the model theoretical approach we isolate classes of KBs for which query answering can be performed efficiently. For the proof theoretical approach we present a calculus for query answering which is valid for any satisfiable KB and any CQ. We also present a control strategy for the calculus which allows one to obtain a LOGSPACE algorithm for answering conjunctive queries

4.1 Conjunction Queries (CQs)

We define a (*simple*) *conjunctive query* q as a query of the following form:

$$q(\vec{x}) \leftarrow \exists \vec{y} A_1(z_1) \wedge \dots \wedge A_k(z_k) \wedge R_1(u_1, v_1) \wedge \dots \wedge R_m(u_m, v_m),$$

where k and m are natural numbers; \vec{x} is a vector of distinguished variables for q ; \vec{y} is a vector of non-distinguished variables for q ; elements u_i, v_i, z_i are either constants from *Cons* or variables occurring in \vec{x} or \vec{y} ; B_i s and R_j s are atomic concepts and atomic roles respectively. If a variable from the body of a query appears there at least twice or it is a distinguished one, then we call it *not unique variable*, otherwise we call it *unique*.

We called CQs defined above *simple* because they allow further extensions by adding either negation, or equality, or inequality, or arithmetical comparisons, or different combinations of these elements in the bodies of the queries. As the matter of convenience we will use shorter notations for CQs. For the CQ q defined above the shorter notation is the following:

$$q(\vec{x}) \leftarrow \exists \vec{y} \text{conj}(\vec{x}; \vec{y}),$$

where $\text{conj}(\vec{x}; \vec{y})$ denotes the conjunction in the body of q and “;” separates distinguished variables

from non-distinguished ones. We call the number of atoms in the body of a CQ the *length* of the CQ. We call a CQ q *safe* if any distinguished variable of q occurs at least once in its body. In the following sections we consider safe simple CQs only.

In terms of Datalog (see, for instance, [AHV95]) a CQ q of the length k is just a rule of the form:

$$q(\vec{x}) \leftarrow D_1(t_1), \dots, D_k(t_k),$$

where each D_i is an atom from the body of q (different D_i s correspond to different atoms in the body); each t_i is a free tuple, i.e. is a vector with components either from $\text{var}(q)$ or from Const . Note that in Datalog notation existential quantification is implicit.

We call query answering for CQs as *conjunctive query answering*. The answer set for $q(\vec{x}) \leftarrow \exists \vec{y} \text{conj}(\vec{x}; \vec{y})$ over \mathcal{K} is a set of tuples of constants:

$$q(\mathcal{K}) = \{\vec{c} \mid \mathcal{K} \models \exists \vec{y} \text{conj}(\vec{c}; \vec{y})\}.$$

The definition tells us that, in order to say that \vec{c} is in the answer set for q , one needs to perform one of the following alternative operations:

- check that the formula $\exists \vec{y} \text{conj}(\vec{c}; \vec{y})$ is true in all models of \mathcal{K} ;
- present a proof of the entailment $\mathcal{K} \models \exists \vec{y} \text{conj}(\vec{c}; \vec{y})$ in some calculus.

These two operations correspond to two alternative approaches to query answering, namely:

- model checking (of a formula over a possibly infinite number of possibly infinite models);
- theorem proving.

Model checking as a way to present query answering corresponds to a branch of logic called *model theory*. Theorem proving in turn corresponds to *proof theory*. In the following sections we consider both approaches separately. First we consider the model theoretical approach and then we consider the proof theoretical one.

4.2 Model Theoretical Approach to Answering CQs

In this section we investigate tractable cases of model checking for conjunctive query answering. By definition of query answering in order to say that \vec{c} is in the answer set for $q(\vec{x}) \leftarrow \exists \vec{y} \text{conj}(\vec{x}; \vec{y})$, one needs to check that the formula $\exists \vec{y} \text{conj}(\vec{c}; \vec{y})$ is true in (possibly infinite number of) all (possibly infinite) models of \mathcal{K} . The following theorem shows that it is enough to check it in a universal model for \mathcal{K} only.

Theorem 6. *Let \mathcal{K} be a satisfiable KB, an interpretation UI be its universal model and $q(\vec{x}) \leftarrow \exists \vec{y} \text{conj}(\vec{x}; \vec{y})$ be a CQ. Then $\vec{c} \in q(\mathcal{K})$ if and only if $UI \models \exists \vec{y} \text{conj}(\vec{c}; \vec{y})$.*

Proof. The *only if* part of the theorem holds by definition of $q(\mathcal{K})$.

To prove the *if* part of the theorem assume that the entailment $UI \models \exists \vec{y} \text{conj}(\vec{c}; \vec{y})$ holds. We are to show that the entailment $I \models \exists \vec{y} \text{conj}(\vec{c}; \vec{y})$ holds for all models $I \models \mathcal{K}$. From the entailment

$UI \models \exists \vec{y} \text{conj}(\vec{c}; \vec{y})$ we conclude that there exists a vector \vec{m} of constants from $Const$, such that $|\vec{y}| = |\vec{m}|$ and the entailment $UI \models \text{conj}(\vec{c}; \vec{m})$ holds.

Let the body of the query be the following conjunction:

$$\text{conj}(\vec{x}, \vec{y}) = B_1(z_1) \wedge \dots \wedge B_k(z_k) \wedge R_1(u_1, v_1) \wedge \dots \wedge R_m(u_m, v_m).$$

Grounding of the conjunction $\text{conj}(\vec{x}; \vec{y})$ with the substitution $\theta = \{\vec{x}/\vec{c}, \vec{y}/\vec{m}\}$ gives the following sentence:

$$\text{conj}(\vec{c}, \vec{m}) = B_1(z_1^0) \wedge \dots \wedge B_k(z_k^0) \wedge R_1(u_1^0, v_1^0) \wedge \dots \wedge R_m(u_m^0, v_m^0).$$

where z_i^0, u_j^0, v_l^0 denote constants obtained from the variables z_i, u_j, v_l (respectively) after the grounding. From the entailment $UI \models \text{conj}(\vec{c}; \vec{m})$ we conclude that the entailments:

$$UI \models B_i(z_i^0) \text{ and } UI \models R_j(u_j^0, v_j^0)$$

hold for all B_i s and R_j s from the body of the query.

Let us consider a model $I \models \mathcal{K}$. Since UI is a universal model for \mathcal{K} we have that there exists a homomorphism $h : UI \rightarrow I$, such that the entailments:

$$I \models B_i(h(z_i^0)) \text{ and } I \models R_j(h(u_j^0), h(v_j^0))$$

hold for all B_i s and R_j s from the body of the query. Hence, the following entailment holds:

$$I \models B_1(h(z_1^0)) \wedge \dots \wedge B_k(h(z_k^0)) \wedge R_1(h(u_1^0), h(v_1^0)) \wedge \dots \wedge R_m(h(u_m^0), h(v_m^0)).$$

This entailment is the same as $I \models \text{conj}(h(\vec{c}); h(\vec{m}))$. Since, any homomorphism is the identity function on the constants from $\text{adom}(\mathcal{K})$, we conclude that $h(\vec{c}) = \vec{c}$ and $I \models \text{conj}(\vec{c}; h(\vec{m}))$ hold. The vector $h(\vec{m})$, by its definition, is just a vector of constants from $Const$, hence, the entailment $I \models \exists \vec{y} \text{conj}(\vec{c}; \vec{y})$ holds. We showed that the entailment $I \models \exists \vec{y} \text{conj}(\vec{c}; \vec{y})$ holds for an arbitrary model $I \models \mathcal{K}$ and, consequently, it holds for all models of \mathcal{K} . This is exactly what we are to prove in the *if* part of the theorem. \square

Taking into account that $\text{chase}(\mathcal{K})$ is a universal model for a satisfiable KB \mathcal{K} (Theorem 4) we obtain the following corollary from Theorem 6.

Corollary 3. *Let \mathcal{K} be a satisfiable KB and $q(\vec{x}) \leftarrow \exists \vec{y} \text{conj}(\vec{x}; \vec{y})$ be a CQ. Then $\vec{c} \in q(\mathcal{K})$ if and only if $\text{chase}(\mathcal{K}) \models \exists \vec{y} \text{conj}(\vec{c}; \vec{y})$.*

Hence, one can perform query answering over any satisfiable KB \mathcal{K} as query evaluation over its chase. As we already saw in Section 3.1, $\text{chase}(\mathcal{K})$ can be infinite in principle. Since, model checking in general and for infinite models in particular is undecidable one need to find a class of KBs such that their chases are finite and model checking over the chases can be performed efficiently.

4.2.1 Chase of Polynomial Depth

In this section we present so-called sets of *weakly-acyclic positive inclusion assertions*, which are a DL version of so-called sets of *weakly-acyclic tgds* presented in [FKMP05]. These assertions are interesting because if the set of all positive inclusion assertions of a KB is weakly-acyclic, then the depth of any chase of the KB is polynomial in the size of the KB.

Let \mathcal{K} be a KB. We denote as $B_{\mathcal{K}}$ a set of all basic concepts that occur in \mathcal{K} . By its definition, $B_{\mathcal{K}}$ consists of three disjoint sets, i.e. atomic concepts and concepts of the form either $\exists R$ or $\exists R^-$.

We define a *dependency graph* for a KB as follows.

Definition 13. *Let \mathcal{K} be a KB. A dependency graph for \mathcal{K} , denoted as $G_{\mathcal{K}}$, is a directed edge-labelled graph, such that:*

1. *the set of nodes of $G_{\mathcal{K}}$ is $B_{\mathcal{K}}$;*
2. *the set of non-labelled edges of $B_{\mathcal{K}}$ is defined as follows: if for two nodes B and B' from $B_{\mathcal{K}}$ there is a positive inclusion assertion $B \sqsubseteq B'$ in \mathcal{K} , then there is an (none-labelled) edge $B \longrightarrow B'$ between them;*
3. *the set of *-labelled edges of $B_{\mathcal{K}}$ is defined as follows: if for two nodes B and B' (i) there is an edge $B \longrightarrow B'$ between them, (ii) $B' = \exists R/B' = \exists R^-$, and (iii) $\exists R^- \in B_{\mathcal{K}}/\exists R \in B_{\mathcal{K}}$, then there is a (*-)labelled edge $B \longrightarrow^* \exists R^-/B \longrightarrow^* \exists R$ in $G_{\mathcal{K}}$.*

Note that in a dependency graph there may be two edges between two nodes with the same direction. That can happen if exactly one of the two edges is labelled.

We say that a set of positive inclusion assertions is *weakly-acyclic* if its dependency graph has no cycles that contain a labelled edge. A KB is weakly-acyclic if the set of all its inclusion assertions is weakly-acyclic.

Intuition behind the definition is the following. Each edge $B \longrightarrow B'$ in dependency graphs keep track of the fact that a constant may be propagated from a basic concept B to B' in the chase sequence. Labelled edges keep track of the fact that, if B' is of the form $\exists R/\exists R^-$, then propagation of a constant to B' also creates a labelled null on the second/first coordinate of R . If a cycle goes through a labelled edge, then the created labelled null cause the creation of another labelled null, in the same position, at a later chase step. This process thus continues forever and leads to an infinite chase.

Theorem 7. *For a satisfiable weakly-acyclic KB its chase has depth which is polynomial in the size of the KB.*

Proof. The proof of this theorem is a simplified version of the one for Theorem 3.9 in [FKMP05]. \square

Theorem 7 and Corollary 3 tells us that if a KB is weakly-acyclic, then one can (i) compute a chase of the KB, it will be finite and have the size, which is polynomial in the size of the KB (ii) store it as a relation DB and (iii) evaluate any query, which is posed to the KB, over the chase.

4.3 Proof Theoretical Approach to Answering CQs

In this section we present a calculus which is to prove (syntactically) that a given vector \vec{c} of constants is in the answer set $q(\mathcal{K})$ of a query q over a satisfiable KB \mathcal{K} . The calculus has a variation of *resolution* in its core. In order to present the calculus we define *extended Horn clauses* and *Extended Horn logic* and then a translation from *DL-Lite_F* and CQs to Extended Horn logic.

4.3.1 Extended Horn Clause (EHC) Logic

A \mathcal{FOL} formula is called an *extended clause*, EC in short, (see [Llo87] as a reference to Horn logic and Logic programming) if it is of the form:

$$\forall \vec{y} \exists \vec{z} (L_1(\vec{x}, \vec{y}, \vec{z}) \vee \dots \vee L_m(\vec{x}, \vec{y}, \vec{z}))$$

where each L_i is a literal; a vector \vec{x} contains all free variables and vectors \vec{y} and \vec{z} contain all quantified variables occurring in $L_1 \vee \dots \vee L_m$. Consider examples of EC:

1. $\forall u_1 \forall v \forall v_1 \exists u (P(x, y, u) \vee \neg Q(x, z, u_1) \vee \neg R(a, v, c, v_1))$;
2. $\forall u_1 \forall v \forall v_1 (\neg Q(x, z, u_1) \vee \neg R(a, v, c, v_1))$;
3. $G(a, b)$

where P, Q, R and G are predicate symbols; a, c are constants; and x, y, z, u_1, v, v_1, u are variables.

Later on we will eliminate quantifications over variables in the prefix of ECs assuming that quantifications are implicit. In order to distinguish between universally, existentially quantified and free variables we will mark variables in the following way:

variable	free occurrence	existentially quantified occurrence	universally quantified occurrence
x	x	x^e	x^u

Thus, we will not mark free variables at all, but put the labels “ e ” and “ u ” on existentially and universally quantified variables, respectively. For instance, an occurrence of a variable u^e in an EC means that a variable u is existentially quantified in the EC. Later on in this section we will use letters (meta-variables) x, y, z (possibly with subscripts) to denote free variables; letters u, v, w (possibly with subscripts) to denote quantified ones and letters p, r, s, \dots (possibly with subscripts) to denote variables without underlining either they are quantified or not. This should not lead to ambiguity.

Extended Horn clause logic (EHC logic) is a fragment of \mathcal{FOL} with equalities such that the only admissible formulae are *extended Horn clauses* (EHC), namely, extended clauses with at most one positive literal. In this work we focuss on functional free EHC logic only, i.e. no functional symbols can appear in EHCs. Consider examples of EHCs, which are the same EHCs that presented above, but written in a quantifiers free notation:

1. $P(x, y, u^e) \vee \neg Q(x, z, u_1^u) \vee \neg R(a, v^u, c, v_1^u)$;

2. $\neg Q(x, z, u_1^u) \vee \neg R(a, v^u, c, v_1^u)$;
3. $G(a, b)$

where P, Q, R and G are predicate symbols; a, c are constants; x, y, z are free variables; u_1^u, v^u, v_1^u are universally quantified variables; u^e is existentially quantified variable. Clauses with both positive and negative literals (like in the formula 1.) are called *rules*, with negative literals only (like in the formula 2.) are called *goals* and with positive grounded literal only (like in the formula 3.) are called *facts*.

4.3.2 Translation from DL-Lite _{\mathcal{F}} Assertions and CQs to EHCs

Let us define a translation (a function) *horn* from DL-Lite _{\mathcal{F}} assertions to EHCs and then extend it to CQs.

On order to construct the translation we first (inductively) define a function *lit* from concepts to literals as follows:

	basic concept: $B =$	literal: $lit(B, x, u^e) =$
(1)	A	$A(x)$
(2)	$\neg A$	$\neg A(x)$
(3)	$\exists R$	$R(x, u^e)$
(4)	$\exists R^-$	$R(u^e, x)$

where A and R are an atomic concept and atomic role, respectively; x, u^e are variables. The literal $lit(B, x, u^e)$ in fact an EHC, which has one atom and at most two variables; all its variables are in the list x, u^e . For example, as it follows from the table, $lit(A, x, u^e) = A(x)$ and $lit(\exists R, x, u^e) = R(x, u^e)$.

The function *horn* is defined as follows:

	assertion: $g =$	extended Horn clause: $horn(g) =$
(1)	$B \sqsubseteq B'$	$lit(B', v^u, u^e) \vee \neg lit(B, v^u, w^u)$
(2)	$B \sqsubseteq \neg B'$	$\neg lit(B', v^u, u^e) \vee \neg lit(B, v^u, w^u)$
(3)	$(func R)$	$\neg R(v^u, v_1^u) \vee \neg R(v^u, v_2^u) \vee (v_1^u = v_2^u)$
(4)	$(func R^-)$	$\neg R(v_1^u, v^u) \vee \neg R(v_2^u, v^u) \vee (v_1^u = v_2^u)$
(5)	$A(a)$	$A(a)$
(6)	$R(a, b)$	$R(a, b)$

where R is an atomic role; B, B' are basic concepts; $v^u, v_1^u, v_2^u, u^e, w^u$ are variable; and a, b are constants. We use a notation $horn(\mathcal{K})$ to denote a set of horn clauses obtained after application of *horn* to all assertion in \mathcal{K} .

Let us consider several examples of application of *horn* to assertions:

- 1) $horn(A \sqsubseteq \exists R) = R(v^u, u^e) \vee \neg A(v^u)$,
- 2) $horn(\exists R^- \sqsubseteq \exists R') = R'(v^u, u^e) \vee \neg R(w^u, v^u)$,

$$3) \text{horn}(\exists R \sqsubseteq \neg A) = \neg A(v^u) \vee \neg R(v^u, w^u).$$

Note, that *horn* maps any assertion to an EHC which is a sentence (has no free variables). Note also, that the function *horn* embed *DL-Lite_F* positive inclusion assertions into an interesting fragment of EHC logic. More precisely, in terms of logic programming, for each positive inclusion assertion g the EHC $\text{horn}(g)$ can be viewed as a logical rule with existentially quantified variables in the head. For example, for an assertion

$$A \sqsubseteq \exists R'$$

the corresponding logical rule (written in Prolog notation) is

$$\exists y R'(x, y) : \neg A(x).$$

If in a set of positive inclusion assertions P each assertion is of the form $B \sqsubseteq A$, where A is an atomic concept, then $\text{horn}(P)$ is just a standard functional free logic program.

Let us extend the function *horn* to CQs. To do it we first define a function *term* that maps terms (i.e. constants and variables) to terms and then, using this function, we extend *horn* to CQs. The function *term* is defined as follows:

term:	image:	
$t =$	$\text{term}(t) =$	
(1) c	c	if c is a constant
(1) x	x	if x is a distinguished variable
(2) v	v^u	if v is an undistinguished variable

The function *term* can be naturally extended to tuples. Essentially, *term* just marks undistinguished (existentially quantified) variables with “ u ”. As one can see, we marked existential variable with the mark of universal variables. The reason is that we will translate CQs (using *horn*) to EHCs which are equivalent to the negation of the body of the CQs. Formally the definition of the function *horn* on CQs is the following. Consider a CQ q :

$$q = q(\vec{x}) \leftarrow D_1(t_1), \dots, D_k(t_k),$$

the image $\text{horn}(q)$ is equal to:

$$\text{horn}(q) = \neg D_1(\text{term}(t_1)) \vee \dots \vee \neg D_k(\text{term}(t_k)),$$

where each t_i is a free tuple, each $D_i(t_i)$ is an atom, and each $\text{term}(t_i)$ is the application of the extended function *term* to t_i . As one can see, the image of a CQ is a goal, where all variables are either free or universally quantified. We call these goals as *CQ goals* to underline that they are obtained from CQs. We say that a CQ goal is *empty* and denote it with \perp if it is a disjunction of the length zero. Consider some examples of application of *horn* to CQs:

1. $\text{horn}(q) = \text{horn}(q(x_1, x_2) \leftarrow A_1(x_1), A_2(v), R_1(a, x_2)) =$

$$\neg A_1(x_1) \vee \neg A_2(v^u) \vee \neg R_1(a, x_2),$$

$$\begin{aligned} 2. \text{horn}(q) = \text{horn}(q(x_1) \leftarrow R(x_1, x_2), R(x_2, x_3), R(x_3, v)) = \\ \neg R(x_1, x_2) \vee \neg R(x_2, x_3) \vee \neg R(x_3, v^u). \end{aligned}$$

The query 2. in the example asks for all constants c_1 such that there is a path (labelled with R) starting in c_1 , ending in c_4 and going through c_1, c_2, c_3, c_4 , for some constants c_2, c_3, c_4 .

4.3.3 Calculus for EHC Logic

Now we will present a calculus for EHC logic that consists of five groups of deductive rules and is called *EHC calculus*. Note, that all the rules should be read bottom-up. Using the rules, a CQ goal corresponding to a CQ and clauses corresponding to a KB, one can deduce answers for the CQ over the KB. A way how to do it and an example of a deduction will be presented after the deductive rules.

Before we introduce calculus let us define a notion of a *query homomorphism*. We say that a mapping h from terms to terms, such that $h(c) = c$ for any $c \in Const$, is a query homomorphism between a CQ query q and a CQ query q' if it is a homomorphism from $T(q)$, the tableau of q , to $T(q')$, the tableau of q' (see the definition of tableaux in [AHV95]). We denote as $h(q)$ a CQ obtained after the application of h (extended from terms to atoms) to q , which is in fact coincides with q' . If Γ is a CQ goal such that $\Gamma = \text{horn}(q)$, then $h(\Gamma)$ denotes a CQ goal, such that $h(\Gamma) = \text{horn}(h(q))$, i.e. it denotes a CQ goal corresponding to a homomorphic image of q under h .

Let s, r be variables (either free or universally quantified); a, b be constants; t, t_1, t_2 be terms, i.e. either free or universally quantified variables or constants; A, R be an atomic concept and an atomic role respectively; Γ be CQ goal (probably of the empty length). Note that in the following deductive rules we assume that the order of literals in the goals is irrelevant, i.e. goals $\Gamma \vee \neg L$ and $\Gamma \vee \neg L \vee \neg L'$, where L and L' are literals should be seen as multisets of their disjuncts.

1. Duplication rules:

1.1 (atomic) concept duplication (cd): 1.2 (atomic) role duplication (rd):

$$\begin{aligned} cd: \frac{\Gamma \vee \neg A(t_1) \vee \neg A(t_1)}{\Gamma \vee \neg A(t_1)} \qquad \qquad rd: \frac{\Gamma \vee \neg R(t_1, t_2) \vee \neg R(t_1, t_2)}{\Gamma \vee \neg R(t_1, t_2)} \end{aligned}$$

2. (New) variable introduction rules:

2.1 variable introduction to a (atomic) concept: (cvi):

$$cvi: \frac{\Gamma \vee \neg A(t) \vee \neg A(v^u)}{\Gamma \vee \neg A(t)}$$

2.2 variable introduction to a (atomic) role: (rvi):

$$\begin{aligned} rvi(1): \frac{\Gamma \vee \neg R(t_1, t_2) \vee \neg R(v^u, t_2)}{\Gamma \vee \neg R(t_1, t_2)} \qquad \qquad rvi(2): \frac{\Gamma \vee \neg R(t_1, t_2) \vee \neg R(t_1, v^u)}{\Gamma \vee \neg R(t_1, t_2)} \end{aligned}$$

where v^u is a **unique variable** for the goals located above the lines of the deductive rules cvi , $rvi(1)$ and $rvi(2)$. Uniqueness of the variable is a condition to apply the cvi and rvi s

rules. Numbers (1) and (2) in $rvi(1)$ and $rvi(2)$ denote that the variable \hat{u} is introduced on the first and the second coordinates of the role R , respectively. We call the literals $\neg A(t)$ and $\neg R(t_1, t_2)$ *active*.

3. Atom introduction rules:

3.1 (atomic) concept introduction (ci): 3.2 (atomic) role introduction (ri):

$$ci : \frac{\Gamma \vee \neg A(t) \quad A(a)}{\Gamma[t/a]} [\theta = \{t/a\}] \quad ri : \frac{\Gamma \vee \neg R(t_1, t_2) \quad R(a, b)}{\Gamma[t_1/a, t_2/b]} [\theta = \{t_1/a, t_2/b\}]$$

where θ is the identity function on constants. The terms t, t_1, t_2 satisfies the following conditions: if t or t_1 is in *Const* then $t = a$ or $t_1 = a$ respectively; if t_2 is in *Const* then $t_2 = b$. Let us consider some cases of the introduction rules. In the case when all the tree terms are constants the introduction rules look as follows:

$$ci : \frac{\Gamma \vee \neg A(a) \quad A(a)}{\Gamma} \quad ri : \frac{\Gamma \vee \neg R(a, b) \quad R(a, b)}{\Gamma}$$

The cases when $t_1 = s, t_2 = b$ and $t_1 = a, t_2 = s$ are respectively the following:

$$ri : \frac{\Gamma \vee \neg R(s, b) \quad R(a, b)}{\Gamma[s/a]} [\theta = \{s/a\}] \quad ri : \frac{\Gamma \vee \neg R(a, s) \quad R(a, b)}{\Gamma[s/b]} [\theta = \{s/b\}]$$

4. Propagation rule:

$$pr : \frac{\Gamma \vee \neg lit(B, t, v^u) \quad lit(B, w^u, w_1^e) \vee \neg lit(B', w^u, w_2^u)}{\Gamma \vee \neg lit(B', t, v^u)}$$

where v^u is a **unique variable** for the goals located above and below the line of the deductive rule. Uniqueness of the variable is a condition to apply the pr rule, an intuition behind the condition is discussed right after the Example 6 and in the Example 7. The clause $lit(B, w^u, w_1^e) \vee \neg lit(B', w^u, w_2^u)$ corresponds to a positive inclusion assertion $B' \sqsubseteq B$. We say that after an application of the deductive rule pr the term t is propagated from the basic concept B' to the basic concept B using the clause $lit(B, w^u, w_1^e) \vee \neg lit(B', w^u, w_2^u)$. As one can see, the pr rule has the clauses $lit(B, w^u, w_1^e)$ and $lit(B', w^u, w_2^u)$ as its parameters. Substitution of different basic concepts for the parameters leads to different variations of the rule. In the Table 4.1 we consider all (nine) possible variations of the pr rule.

5. Generalization rule:

$$gen : \frac{\Gamma}{h(\Gamma)}[h]$$

where, if $\Gamma = horn(q)$ for some CQ q , then $h(\Gamma)$ is a CQ goal with corresponding query, say q' , which is a homomorphism image of q under h . The rule is called “generalization”, for q' is a more specific query then q , i.e. for the queries the inclusion $q' \subseteq q$ holds. This fact can be easily proved using Homomorphism theorem for CQ (see the theorem in [AHV95]).

Note, that the clauses that can appear in the deductive rules of the EHC calculus are CQ goals (they appear below and above the line of each rule); facts (they appear above the line of the atom

Table 4.1: Cases of Propagation Rule

$B' =$	$B = A$	$B' =$	$B = \exists R$
A'	$\frac{\Gamma \vee \neg A(t) \quad A(w^u) \vee \neg A'(w^u)}{\Gamma \vee \neg A'(t)}$	A'	$\frac{\Gamma \vee \neg R(t, v^u) \quad R(w^u, w_1^e) \vee \neg A(w^u)}{\Gamma \vee \neg A(t)}$
$\exists R'$	$\frac{\Gamma \vee \neg A(t) \quad A(w^u) \vee \neg R'(w_2^u, w_2^u)}{\Gamma \vee \neg R'(t, v^u)}$	$\exists R'$	$\frac{\Gamma \vee \neg R(t, v^u) \quad R(w^u, w_1^e) \vee \neg R'(w_2^u, w_2^u)}{\Gamma \vee \neg R'(t, v^u)}$
$\exists R'^-$	$\frac{\Gamma \vee \neg A(t) \quad A(w^u) \vee \neg R'(w_2^u, w^u)}{\Gamma \vee \neg R'(v^u, t)}$	$\exists R'^-$	$\frac{\Gamma \vee \neg R(t, v^u) \quad R(w^u, w_1^e) \vee \neg R'(w_2^u, w^u)}{\Gamma \vee \neg R'(v^u, t)}$
$B' =$	$B = \exists R^-$		
A'	$\frac{\Gamma \vee \neg R(v^u, t) \quad R(w_1^e, w^u) \vee \neg A(w^u)}{\Gamma \vee \neg A(v^u)}$		
$\exists R'$	$\frac{\Gamma \vee \neg R(v^u, t) \quad R(w_1^e, w^u) \vee \neg R'(w_2^u, w_2^u)}{\Gamma \vee \neg R'(t, v^u)}$		
$\exists R'^-$	$\frac{\Gamma \vee \neg R(v^u, t) \quad R(w_1^e, w^u) \vee \neg R'(w_2^u, w^u)}{\Gamma \vee \neg R'(v^u, t)}$		

introduction rules); and rules (that appear above the line of the propagation rule and correspond to positive inclusion assertions). Hence, none of EHCs that correspond to negative inclusion or functional assertions appear in the deductive rules.

Note also, that all variable introduction rules are in fact just instances of sequential applications of two rules, i.e. first the duplication rules and then the generalization rule with specific homomorphisms. More precisely, the *cvi* rule is the *cd* rule combined with the *gen* rule with $h = \{v^u/t\}$ and *rvi*(1), *rvi*(2) rules are *rd* combined with *gen* with $h = \{v^u/t_1\}$, $h = \{v^u/t_2\}$, respectively.

If one reads deductive rules in the top-down fashion, one can notice, that the propagation and atom introduction rules work similarly to the resolution rule. Let us illustrate this similarity. Consider a *ci* rule:

$$\frac{\Gamma \vee \neg A(s) \quad A(a)}{\Gamma[s/a]} [\theta = \{s/a\}].$$

It works exactly as the resolution does, namely: (i) it takes $\Gamma \vee \neg A(s)$ and $A(a)$ as the first and the second input clauses (ii) it unifies two atoms $A(s)$ and $A(a)$ in the input clauses, the unification returns an *mgu* $\theta = \{s/a\}$; (iii) it returns a *resolvent* of the two input clauses, i.e. it substitutes \perp to $A(s)$ in the first input clause, applies θ to the result of this substitution and returns the clause $\Gamma[s/a]$. As for the propagation rule, consider a *pr* rule:

$$\frac{\Gamma \vee \neg \text{lit}(B, t, v^u) \quad \text{lit}(B, w^u, w_1^e) \vee \neg \text{lit}(B', w^u, w_2^u)}{\Gamma \vee \neg \text{lit}(B', t, v^u)}$$

This deductive rule works a bit differently from the resolution, namely it first renames w_1^e to w_1^u (it changes the existential quantification of w_1 to the universal one) and then applies the resolution, i.e. the steps (i)-(iii) presented for the *ci* rule. Intuitively, this changing in the quantification of w_1 makes sense (from the query answering point of view) only because w_1 is to be unified with the

variable v^u , which is unique (and consequently irrelevant for the query answering). Formally we will prove it in the next section (see Lemma 3).

Let Γ and Γ' be CQ goals; g either a positive inclusion or a membership assertion; θ be a substitution; and h be a homomorphism from Γ' to Γ . We say that

- (1) Γ' is directly derived from Γ and denote it as $\Gamma \vdash \Gamma'$; or
- (2) Γ' is directly derived from Γ with h and denote it as $\Gamma \vdash_h \Gamma'$; or
- (3) Γ' is directly derived from Γ and g and denote it as $\Gamma \wedge \text{horn}(g) \vdash \Gamma'$; or
- (4) Γ' is directly derived from Γ and g with θ and denote it as $\Gamma \wedge \text{horn}(g) \vdash_\theta \Gamma'$,

if there exists an EHC calculus rule

$$(1) \frac{\Gamma'}{\Gamma} \quad \text{or} \quad (2) \frac{\Gamma'}{\Gamma}[h] \quad \text{or} \quad (3) \frac{\Gamma' \quad \text{horn}(g)}{\Gamma} \quad \text{or} \quad (4) \frac{\Gamma' \quad \text{horn}(g)}{\Gamma}[\theta],$$

respectively. We use a term *derivation step* and a notation \mathcal{D} to denote a rule of the EHC calculus. Obviously, any derivation step \mathcal{D} has one of the forms (1)–(4) as it is presented above. We use terms *premise of the derivation step* \mathcal{D} and *conclusion of the derivation step* \mathcal{D} to denote the goals which are below and above the line of some derivation step \mathcal{D} , respectively. If Γ is the premise and Γ' is the conclusion of a derivation step \mathcal{D} , then we say that \mathcal{D} is a derivation step from Γ to Γ' . We define a relation *derived from* between two CQ goals, say Γ and Γ' , as the transitive closure of the relation “directly derived from” or, equivalently, as a sequence of derivation steps $\mathcal{D}_1 \dots \mathcal{D}_n$, such that Γ is the premise of \mathcal{D}_1 , Γ' is the conclusion of \mathcal{D}_n and for each pair of derivation steps $\mathcal{D}_i, \mathcal{D}_{i+1}$ the conclusion of the former one coincides with the premise of the latter one. This sequences of derivation steps can be depicted as a *derivation tree*. The definition of the derivation tree is the same as it is for the Gentzen systems (you can see detailed definitions in [TS03]), hence, will not give it explicitly. One can see an example of the tree in the Examples 6 and 7. Formally the definition of the “derived from” relation is the following. Let Γ and Γ' be CQ goals again; G be a set of positive inclusion or membership assertions; θ be a mapping from terms to terms. We say that Γ' is derived from Γ and G with θ and denote it as:

$$\Gamma \wedge \text{horn}(G) \vdash_\theta^* \Gamma', \quad \text{if}$$

- there exists a sequence of CQ goals $\Gamma_1, \dots, \Gamma_n$, for some natural n ,
- there exists a set of substitutions $\Theta = \{\theta_1, \dots, \theta_m\}$, and a sequence of homomorphisms $H = h_1, \dots, h_k$ where $k + m < n$,

such, that the following conditions hold:

1. $\theta = \bigcup_{j=1}^m \theta_j \circ (h_k \circ \dots \circ h_1)$;
2. $\Gamma_1 = \Gamma, \Gamma_n = \Gamma'$;
3. for any pair Γ_{i-1}, Γ_i in the sequence there is a derivation step from Γ_{i-1} to Γ_i , which can involve an assertion from G only;

4. for each substitution $\theta' \in \Theta$ or homomorphism $h' \in H$ there is a pair of goals Γ_{i-1}, Γ_i in the sequence, such that θ' or h' participates in a derivation step from Γ_{i-1} to Γ_i . Moreover, if h_i participates in a derivation step from Γ_{j-1} to Γ_j and h_{i+1} from Γ_{l-1} to Γ_l , then $j < l$.

Note, that in the definition some assertions from G may not participate in any derivation step from a goal Γ_{i-1} to a goal Γ_i . We say that *there is a derivation from Γ to Γ'* if there is a set of positive inclusion and membership assertions G and a substitution θ , such that Γ' is derived from Γ and G with θ . As the matter of convenience we will use $\mathcal{D}er$ to denote a derivation.

We say that a CQ goal Γ is *derived from a KB \mathcal{K} with θ* and denote it as $\mathcal{K} \vdash_{\theta}^* \Gamma$, if it is derived from \perp and the set of assertions from \mathcal{K} with a substitution θ .

Example 6. Let us consider an example of a derivation. Let a query q be the following:

$$q(x_1) \leftarrow A'(a), A(x_1), A(x_1), R(x_1, x_2), R(x_2, x_3);$$

the corresponding CQ goal to the query is:

$$\text{horn}(q) = \neg A'(a) \vee \neg A(x_1) \vee \neg A(x_1) \vee \neg R(x_1, x_2) \vee \neg R(x_2, v^u).$$

This CQ goal can be derived from a KB:

$$\mathcal{K} = \{\exists R^- \sqsubseteq \exists R, \quad R(a, b), A'(a), A(a)\}.$$

EHCs corresponding to the \mathcal{K} 's assertions are the following:

$$R(w^u, w_1^e) \vee \neg R(w_2^u, w^u), \quad R(a, b), A(a)', A(a).$$

Consider a derivation $\mathcal{K} \vdash_{\{x_1/a, x_2/b\}}^* \text{horn}(q)$:

$$\begin{array}{c} ci \frac{\neg A'(a) \vee \neg A(x_1) \vee \neg A(x_1) \vee \neg R(x_1, x_2) \vee \neg R(x_2, v^u) \quad A'(a)}{\neg A(x_1) \vee \neg A(x_1) \vee \neg R(x_1, x_2) \vee \neg R(x_2, v^u)} \\ cd \frac{\neg A(x_1) \vee \neg A(x_1) \vee \neg R(x_1, x_2) \vee \neg R(x_2, v^u)}{pr \frac{\neg A(x_1) \vee \neg R(x_1, x_2) \vee \neg R(x_2, v^u) \quad R(w^u, w_1^e) \vee \neg R(w_2^u, w^u)}{rvi(1) \frac{\neg A(x_1) \vee \neg R(x_1, x_2) \vee \neg R(v^u, x_2)}{\neg A(x_1) \vee \neg R(x_1, x_2)}}} \\ [x_1/a, x_2/b] \frac{\neg A(x_1) \vee \neg R(x_1, x_2) \vee \neg R(x_2, v^u) \quad R(a, b)}{ci \frac{\neg A(a)}{\perp}} ri \quad A(a) \end{array}$$

Let us understand how the derivation works. In order to do it, let us read the derivation in a top-down fashion. In the first step of the derivation the literal $\neg A'(a)$ is eliminated by the *ci* rule. In the second step of the derivation the literal $\neg A(x_1)$ is eliminated by the *cd* rule. In the third step, using the clause $R(w^u, w_1^e) \vee \neg R(w_2^u, w^u)$, the variable x_2 is propagated from the second coordinate of R to the first one, i.e the variables x_2 and v^u swap around in the atom $R(x_2, v^u)$ by the *pr* rule. In the fourth step v^u is merged with x_2 and $\neg R(x_2, v^u)$ is eliminated by the *rvi(1)* rule. In the fifth step, using the atom $R(a, b)$ and the substitution $\theta = \{x_1/a, x_2/b\}$, the literal $\neg R(x_1, x_2)$ is eliminated by the *ri* rule. In the last sixth step, using the atom $A(a)$, the literal $\neg A(a)$ is eliminated by the *ci* rule.

The result of the last step is the empty goal \perp . If one read the derivation in a bottom-up fashion one will see how the CQ goal $\text{horn}(q)$ is derived (generated) from \perp using clauses corresponding to assertion in \mathcal{K} and θ . If one applies the substitution θ obtained in the derivation to the distinguished variables of q , namely to x_1 , one will get an answer $\theta(x_1) = a \in q(\mathcal{K})$ for q over \mathcal{K} . In the next sections we will show that all the answers $q(\mathcal{K})$ of a CQ q over a KB \mathcal{K} can be derived in EHC calculus from \mathcal{K} ; moreover, nothing else can be derived. This result shows *correctness* of the EHC calculus wrt CQ answering.

Now we will discuss an importance of the conditions in the propagation rule. Recall the propagation rule:

$$\text{pr} : \frac{\Gamma \vee \neg \text{lit}(B, t, v^u) \quad \text{lit}(B, w^u, w_1^e) \vee \neg \text{lit}(B', w^u, w_2^u)}{\Gamma \vee \neg \text{lit}(B', t, v^u)}$$

where v^u is a unique variable for the goals located above and below the line of the deductive rule. In order to illustrate an importance of the condition on the variable v^u , we introduce a new deductive rule, say pr' , which is the same as the pr rule except the fact that the variable v^u is an arbitrary variable in the goals located above and below the line of the rule. We denote a calculus obtained from EHC calculus by adding the pr' rule as EHC' calculus. The following example shows that EHC' calculus are unsound wrt to CQ answering, namely one can provide a derivation that returns a tuple of constants that is not in the answer set.

Example 7. Let a query q be the following:

$$q(x_1) \leftarrow R(x_1, x_2) \wedge R(x_2, x_3).$$

The query asks for all constants c , such that there is an R -path of length two (with two R -arcs) that starts in c and connects c to some constant c' and c' to some constant c'' . A CQ goal corresponding to the query is:

$$\text{horn}(q) = \neg R(x_1, x_2) \vee \neg R(x_2, v^u).$$

Consider a KB:

$$\mathcal{K} = \{A \sqsubseteq \exists R, \quad A(a)\}.$$

It is obvious that an interpretation $I = I(\{A(a), R(a, n)\})$ is a universal model for \mathcal{K} . Hence, by Theorem 6, one can evaluate the query q over I . There are obviously no R -paths of the length two in I , thus, the evaluation $q(I)$ returns the empty set as the answer set. Consider EHCs corresponding to the \mathcal{K} 's assertions:

$$R(w^u, w_1^e) \vee \neg A(w^u), \quad A(a).$$

Consider a derivation $\mathcal{K} \vdash_{\{x_1/a, x_2/a\}}^* \text{horn}(q)$ in the EHC' calculus:

$$\begin{array}{c} \text{pr}' \frac{\neg R(x_1, x_2) \vee \neg R(x_2, v^u) \quad R(w^u, w_1^e) \vee \neg A(w^u)}{\text{pr} \frac{\neg A(x_1) \vee \neg R(x_2, v^u) \quad R(w^u, w_1^e) \vee \neg A(w^u)}{[x_1/a] \frac{\neg A(x_1) \vee \neg A(x_2) \quad A(a)}{[x_2/a] \frac{\neg A(x_2)}{\perp} \text{ci} } A(a)} \text{ci} } \end{array}$$

If one applies the substitution $\theta = \{x_1/a\} \cup \{x_2/a\} = \{x_1/a, x_2/a\}$ obtained in the derivation to the distinguished variables of q , namely to x_1 , one will get a constant $\theta(x_1) = a$. The constant is derived from \mathcal{K} but is not in the answer set $q(\mathcal{K}) = \emptyset$. Hence, the EHC' calculus are not sound wrt to CQ answering.

Since, as it shown in Theorem 8, EHC calculus are sound wrt CQ answering, the unsoundness in the EHC' calculus comes from the pr' rule. Let us understand the reasons of this phenomena. Consider the very last derivation step, pr' , in the derivation above. From the query $A(x_1) \wedge R(x_2, v^e)$ corresponding to the premises and the clause $R(w^u, w_1^e) \vee \neg A(w^u)$, the pr' rule derives a query $R(x_1, x_2) \wedge R(x_2, v^e)$ corresponding to the conclusion. So, the rule substitutes the literal $\neg R(x_1, x_2)$ which has two variables for the literal $\neg A(x_1)$ which has only one variable. This difference in the number of variables a crucial one. The reason is that there is no way to find a connection between the variable x_2 on the second coordinate in the literal $\neg R(x_1, x_2)$ and the variable in $\neg A(x_1)$. So, one need to “invent” this variable. In the example above the “invented” variable is x_2 and it coincides with the one on the first position in the literal $\neg R(x_2, v^u)$. This unjustified binding between two literals in the CQ goal $\neg R(x_1, x_2) \vee \neg R(x_2, v^u)$, which appeared due to the fact that the “invented” variable is x_2 , is actually the reason why we deduced a constant a which is not in the answer set $q(\mathcal{K})$. In order to avoid this unexpected binding of variables in the conclusion of the derivation step pr we put a uniqueness condition on the variable v^u in the propagation rule.

4.3.4 Soundness of EHC Calculus.

In this section we show that the EHC calculus are sound wrt to CQ answering, i.e. if there is a derivation of a CQ q from a KB \mathcal{K} , which gives a substitution that (after its application to a vector of distinguish variables of q) returns a tuple of constants, then the tuple is in the answer set $q(\mathcal{K})$. In order to show this, we first prove that each derivation rule of the EHC calculus returns a goal in the conclusion that has the same truth value as the goal in the premise does. This result is similar to the one for the resolution derivation step (see [Llo87]).

Lemma 3. *Let Γ and Γ' be CQ goals; an assertion g be either a positive inclusion assertion or a membership assertion; and μ be a grounding substitution for Γ (it maps all free variables of Γ to constants). The following holds:*

- (i) *if $\Gamma \vdash \Gamma'$ then $\neg\mu\Gamma \models \neg\mu\Gamma'$;*
- (ii) *if $\Gamma \vdash_h \Gamma'$ then $\neg\mu\Gamma \models \neg(\mu \circ h)\Gamma'$;*
- (iii) *if $\Gamma \wedge \text{horn}(g) \vdash \Gamma'$ then $\neg\mu\Gamma \wedge \text{horn}(g) \models \neg\mu\Gamma'$;*
- (iv) *if $\Gamma \wedge \text{horn}(g) \vdash_\theta \Gamma'$, then $\neg\mu\Gamma \wedge \text{horn}(g) \models \neg(\mu \cup \theta)\Gamma'$.*

Proof. Assume there are given two CQ goals Γ and Γ' , a positive inclusion assertion or a membership assertion g and a substitution μ which is a grounding one for Γ . We will first show that the (i) statement holds, then that the (ii) one holds and at the end that the (iii) one holds.

(i) statement. We are to show that a derivation step $\Gamma \vdash \Gamma'$ implies an entailment $\neg\mu\Gamma \models \neg\mu\Gamma'$. The derivation step corresponds to several deductive rules, namely to the duplication (*cd* and *rd*) rules and the variable introduction (*cvi*, *rvi*(1) and *rvi*(2)) rules. The (i) statement trivially holds for both *cd* and *rd* because of the following \mathcal{FOL} equivalence: $\phi \vee \psi \equiv \phi \vee \psi \vee \psi$.

Let us show the (i) statement for the *cvi* rule, i.e. let $\Gamma \vdash \Gamma' = \frac{\Gamma_1 \vee \neg A(s) \vee \neg A(v^u)}{\Gamma_1 \vee \neg A(s)}$, where a CQ goal $\Gamma = \Gamma_1 \vee \neg A(s)$; a CQ goal $\Gamma' = \Gamma_1 \vee \neg A(s) \vee \neg A(v^u)$; and s is a variable which is not existentially quantified; and v^u is a unique variable for Γ' . We are to consider two cases, namely, when s is a free and universally quantified variable. If s is universally quantified, then the *cvi* rule transforms to the *cd* rule, hence, the (i) statement holds. Assume s is a free variable. In order to show that $\neg\mu\Gamma \models \neg\mu\Gamma'$, we must prove that for any interpretation I such that $I \models \neg\mu\Gamma$, the entailment $I \models \neg\mu\Gamma'$ holds. Assume that I satisfies the entailment $I \models \neg\mu(\Gamma_1 \vee \neg A(s))$, we conclude that $I \models \mu A(s)$, or, equivalently, $I \models A(\mu(s))$. Using the last entailment and the fact that for any \mathcal{FOL} formula $\phi(x)$, where x is a free variable in ϕ , the implication:

$$\phi(x) \rightarrow \exists x \phi(x) \quad (4.1)$$

is true in any interpretation, we conclude that the entailment $I \models \exists v A(\mu(v))$ holds for some variable v . Consequently, the entailment $I \models \neg\forall v A(\mu(v))$ holds. Since, $\forall v A(\mu(v))$ is exactly $\mu A(v^u)$ we obtain that $I \models \neg\mu(\neg A(v^u))$. Two entailments $I \models \neg\mu(\Gamma_1 \vee \neg A(s))$ and $I \models \neg\mu(\neg A(v^u))$ implies that the entailment $I \models \neg\mu(\Gamma_1 \vee \neg A(s) \vee \neg A(v^u))$ and consequently the entailment $I \models \neg\mu\Gamma'$ holds.

One can prove the (i) statement for the *rvi*(1) and *rvi*(2) rules in a similar way as it is done for *cvi*.

(ii) statement. This part of the lemma trivially follows from the fact that $\Gamma = h(\Gamma')$, which implies that the entailment $\neg\mu\Gamma \models \neg(\mu \circ h)\Gamma'$ transforms to the one $\neg\mu(h(\Gamma')) \models \neg(\mu(h(\Gamma')))$ that obviously holds.

(iii) statement. We are to show that a derivation step $\Gamma \wedge \text{horn}(g) \vdash \Gamma'$ implies an entailment $\neg\mu\Gamma \wedge \text{horn}(g) \models \neg\mu\Gamma'$. The derivation step corresponds to the propagation rule (*pr*) and some of the atom introduction rules (*ci* and *ri* where no substitution is involved). Let us first consider the *pr* rule:

$$pr = \frac{\Gamma_1 \vee \neg \text{lit}(B, t, v^u) \quad \text{lit}(B, w^u, w_1^e) \vee \neg \text{lit}(B', w^u, w_2^u)}{\Gamma_1 \vee \neg \text{lit}(B', t, v^u)},$$

where a CQ goal $\Gamma = \Gamma_1 \vee \neg \text{lit}(B', t, v^u)$; a CQ goal $\Gamma' = \Gamma_1 \vee \neg \text{lit}(B, t, v^u)$; a clause $\text{horn}(g) = \text{lit}(B, w^u, w_1^e) \vee \neg \text{lit}(B', w^u, w_2^u)$; t is a term; and v^u, w^u, w_1^e, w_2^u are variables. We will consider three cases, namely when t is a free variable, when t is a universally quantified one and when t is a constant.

Assume that t is a universally quantified variable, that is $t = u^u$ and $\mu(u^u) = u^u$. In order to show the (iii) statement we must prove that for any interpretation I such that $I \models \neg\mu\Gamma$ and $I \models \text{horn}(g)$ the entailment $I \models \neg\mu\Gamma'$ holds. Assume that I satisfies the entailment $I \models \neg\mu(\Gamma_1 \vee \neg \text{lit}(B', u^u, v^u))$, we conclude that $I \models \neg\mu(\Gamma_1)$ and $I \not\models \neg\mu(\text{lit}(B', u^u, v^u))$, or, equivalently, $I \not\models \neg \text{lit}(B', u^u, v^u)$. After a renaming of variables $\{u/w, v/w_2\}$ in the last entailment we get:

$$I \not\models \neg \text{lit}(B', w^u, w_2^u). \quad (4.2)$$

Since, the clause $horn(g)$ is true in I , i.e. $I \models lit(B, w^u, w_1^e) \vee \neg lit(B', w^u, w_2^u)$, we obtain $I \models lit(B, w^u, w_1^e)$. Using the last entailment and the fact that for any \mathcal{FOL} formula ϕ the implication:

$$\forall x \phi \rightarrow \exists x \phi \quad (4.3)$$

holds in an interpretation, we conclude that $I \models lit(B, w^e, w_1^e)$, or equivalently $I \not\models \neg lit(B, w^u, w_1^u)$. Using the later entailment and the fact that $I \models \neg \mu(\Gamma_1)$, we conclude that $I \not\models \mu(\Gamma_1) \vee \neg lit(B, w^u, w_1^u)$. After renaming of variables $\{w/t, w_1/v\}$ in the last entailment we get $I \not\models \mu(\Gamma_1 \vee \neg lit(B, t, v^u))$, which is the same as $I \models \neg \mu(\Gamma_1 \vee \neg lit(B, t, v^u))$. By definition of Γ' we obtain that the entailment $I \models \neg \mu \Gamma'$ holds.

Assume that t is a free variable. In order to show the (iii) statement we must prove that for any interpretation I such that $I \models \neg \mu \Gamma$ and $I \models horn(g)$ the entailment $I \models \neg \mu \Gamma'$ holds. Assume that I satisfies the entailment $I \models \neg \mu(\Gamma_1 \vee \neg lit(B', t, v^u))$, we conclude $I \models \neg \mu(\Gamma_1)$ and $I \not\models \neg \mu(lit(B', t, v^u))$, or, equivalently, $I \models lit(B', \mu(t), v^e)$. Last entailment can be rewritten as $I, \mu \models lit(B', t, v^e)$. Using the Formula 4.1 we obtain that the entailment $I, \mu \models \exists w lit(B', w, v^e)$ holds for some w or, equivalently, $I \models \exists w lit(B', w, v^e)$. Since, $\exists w lit(B', w, v^e)$ is exactly $lit(B', w^e, v^e)$ we obtain $I \models lit(B', w^e, v^e)$. Hence the entailment $I \not\models \neg lit(B', w^u, v^u)$ holds. After a renaming of variables $\{v/w_2\}$ in the last entailment we get $I \not\models \neg lit(B', w^u, w_2^u)$. Observe, that obtained entailment is exactly the same as in the Formula 4.2. Hence, in order to accomplish the current proof (for the case when t is free variable) one can reuse the part the proof (for the case when t is a universally quantified variable), that follows the Formula 4.2.

Assume that t is a constant, say c . In order to show the (iii) statement we must prove that for any interpretation I such that $I \models \neg \mu \Gamma$ and $I \models horn(g)$ the entailment $I \models \neg \mu \Gamma'$ holds. Assume that I satisfies the entailment $I \models \neg \mu(\Gamma_1 \vee \neg lit(B', c, v^u))$, we conclude $I \models \neg \mu(\Gamma_1)$ and $I \not\models \neg \mu(lit(B', c, v^u))$, or, equivalently, $I \not\models \neg lit(B', \mu(c), v^u)$ and, since $\mu(c) = c$, we obtain $I \not\models \neg lit(B', c, v^u)$. Taking into account that $horn(g) = lit(B, w^u, w_1^e) \vee \neg lit(B', w^u, w_2^u)$ and $I \models horn(g)$, we conclude $I \models lit(B, c, w_1^e)$ and $I \not\models \neg(\mu(lit(B, c, w_1^u)))$. After a renaming of variables $\{w_1/v\}$ in the last entailment we obtain $I \not\models \neg(\mu(lit(B, c, v^u)))$ and, consequently, $I \models \neg \mu \Gamma'$ holds.

Let us show the (iii) statement for the ci rule (for the ri rule it can be done similarly), i.e. let: $\Gamma \vdash \Gamma' = \frac{\Gamma_1 \vee \neg A(a) \quad A(a)}{\Gamma_1}$, where a CQ goal $\Gamma = \Gamma_1$; a CQ goal $\Gamma' = \Gamma_1 \vee \neg A(a)$; and $horn(g) = A(a)$. We must prove that for any interpretation I such that $I \models \neg \mu \Gamma$ and $I \models horn(g)$ the entailment $I \models \neg \mu \Gamma'$ holds. Assume that I satisfies the entailment $I \models \neg \mu \Gamma_1$, and $I \models A(a)$. We obviously conclude that $I \models \neg \mu(\Gamma_1) \wedge A(a)$, or, equivalently, $I \models \neg \mu(\Gamma_1 \vee \neg A(a))$, i.e. the entailment $I \models \neg \mu \Gamma'$ holds.

(iv) statement. We are to show that a derivation step $\Gamma \vdash_{\theta} \Gamma'$ implies an entailment $\neg \mu \Gamma \models \neg(\mu \cup \theta) \Gamma'$. The derivation step corresponds those atom introduction rules ci and ri which involves substitutions. We will prove the statement for ci , a proof for ri can be done similarly. Recall the ci derivation step:

$$ci : \frac{\Gamma_1 \vee \neg A(s) \quad A(a)}{\Gamma_1[s/a]} [\theta = \{s/a\}]$$

where a CQ goal $\Gamma = \Gamma_1[s/a]$; a CQ goal $\Gamma' = \Gamma_1 \vee \neg A(s)$; s is variables; a is a constant; and $\theta = \{s/a\}$ is a substitution. In order to show the (iii) statement we must prove that for any interpretation

I such that $I \models \neg\mu(\Gamma_1[s/a])$ and $I \models A(a)$ the entailment $I \models \neg(\mu \cup \theta)(\Gamma_1 \vee \neg A(s))$ holds. This can be equivalently rewritten as the following: for any interpretation I if $I \models (\mu \cup \theta)(\neg\Gamma_1)$ and $I \models A(a)$, then $I \models (\mu \cup \theta)(\neg\Gamma_1) \wedge (\mu \cup \theta)A(s)$. Hence, it is enough to show that $I \models A(a)$ implies $I \models (\mu \cup \theta)A(s)$. If s is a free variable, then the implication trivially holds. If s is existentially quantified (note, that by definition of Γ' , s can not be universally quantified), then $I \models (\mu \cup \theta)A(s)$ is equivalent to $I \models A(u^e)$, for some variable u . Hence, the implication trivially holds by definition of truth valuation of formulae with existentially quantified variables. \square

In the following theorem we show soundness of the EHC calculus, applying the previous lemma to each derivation step; and using *transitivity of logical implication*, i.e. the fact that for any \mathcal{FOC} formulae ϕ, ψ and any ground goals $\Gamma, \Gamma', \Gamma''$ the following holds. If $\neg\Gamma \wedge \phi \models \neg\Gamma'$, and $\neg\Gamma' \wedge \psi \models \neg\Gamma''$, then $\neg\Gamma \wedge \phi \wedge \psi \models \neg\Gamma''$.

Theorem 8. (*Soundness Theorem*)

Let \mathcal{K} be a KB; $q(\vec{x}) \leftarrow \exists \vec{y} \text{conj}(\vec{x}; \vec{y})$ be a CQ over \mathcal{K} ; and θ be a substitution. If the following derivation exists:

$$\mathcal{K} \vdash_{\theta}^* \text{horn}(q)$$

then the following entailment holds:

$$\mathcal{K} \models \exists \vec{y} \text{conj}(\vec{c}; \vec{y}), \text{ where } \vec{c} = \theta(\vec{x})$$

Proof. Assume $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a satisfiable KB; $q(\vec{x}) \leftarrow \exists \vec{y} \text{conj}(\vec{x}; \vec{y})$ is a CQ over \mathcal{K} ; and θ is a substitution.

Assume the derivation $\mathcal{D}er = \mathcal{K} \vdash_{\theta}^* \text{horn}(q)$ exists. Thus, by definition of derivation, there exists a sequence of derivation steps $\mathcal{D}_1, \dots, \mathcal{D}_n$, such that:

- the premise of \mathcal{D}_1 is \perp , the conclusion of \mathcal{D}_n is $\text{horn}(q)$,
- and for each pair of steps $\mathcal{D}_{i-1}, \mathcal{D}_i$ the conclusion of \mathcal{D}_{i-1} coincides with the premise of \mathcal{D}_i ,
- if $\{\theta_1, \dots, \theta_m\}$ is a set of all substitutions and h_1, \dots, h_k is a sequence of all homomorphisms, where $m + k \leq n$, that participate in $\mathcal{D}er$ then $\theta = \bigcup_{i=1}^m \theta_i \circ h_k \circ \dots \circ h_1$,
- if $\{\text{horn}(p_1), \dots, \text{horn}(p_k)\}$, where $k \leq n$ is a set of all positive inclusion and membership assertions that participate in $\mathcal{D}er$ then each $p_i \in \mathcal{K}$.

Using (i) transitivity of logical implication, (ii) the fact that the empty substitution ϵ is a grounding for the goal \perp , (iii) homomorphisms are closed under composition and applying Lemma 3 to each derivation step in $\mathcal{D}er$, we obtain that $\text{horn}(p_1) \wedge \dots \wedge \text{horn}(p_k) \models \neg((\epsilon \cup \bigcup_{i=1}^m \theta_i) \circ h_k \circ \dots \circ h_1) \text{horn}(q)$. By definition of the function horn , the formula on the right hand side of the logical implication is equal to $\theta(\exists \vec{y} \text{conj}(\vec{x}; \vec{y}))$ and, consequently, $\mathcal{K} \models \exists \vec{y} \text{conj}(\vec{c}; \vec{y})$, where $\vec{c} = \theta(\vec{x})$, holds. \square

4.3.5 Completeness of EHC Calculus

In this section we show that the EHC calculus are complete wrt to CQ answering, i.e. if a tuple of constants is in the answer set of a CQ over a KB then there is a derivation of the query from the KB, which gives a substitution that (after its application to a vector of distinguish variables of the query) returns this tuple. This result together with the one that EHC calculus are sound implies correctness of EHC calculus wrt to CQ answering. In order to show completeness, we first prove two supporting lemmas.

In the first lemma we show how one can derive a query $q(\vec{x})$ from the ABox of a KB if there is such a grounding γ of the distinguished variables of the query, that $q(\gamma(\vec{x}))$ is true in the minimal model of the ABox.

Lemma 4. *Let $q(\vec{x}) \leftarrow \exists \vec{y} conj(\vec{x}; \vec{y})$ be a CQ; \mathcal{K} be a satisfiable KB; and $\vec{c} \in q(\mathcal{K})$. If $I(\mathcal{A}) \models \exists \vec{y} conj(\vec{c}; \vec{y})$, then there is an derivation $\mathcal{A} \vdash_{\theta}^* horn(q)$, where $\{\vec{x}/\vec{c}\} \subseteq \theta$.*

Proof. Assume $q(\vec{x}) \leftarrow \exists \vec{y} conj(\vec{x}; \vec{y})$ is a CQ; \mathcal{K} is a satisfiable KB; $\vec{c} \in q(\mathcal{K})$; and $I(\mathcal{A}) \models \exists \vec{y} conj(\vec{c}; \vec{y})$. We will show that there is a derivation $\mathcal{A} \vdash_{\theta}^* horn(q)$, where $\{\vec{x}/\vec{c}\} \subseteq \theta$.

Let $conj(\vec{x}; \vec{y})$ be $P_1(\vec{w}_1) \wedge \dots \wedge P_n(\vec{w}_n)$. Since $I(\mathcal{A}) \models \exists \vec{y} conj(\vec{c}; \vec{y})$ holds, there is a vector of constants \vec{c}_1 such that $I(\mathcal{A}) \models conj(\vec{c}; \vec{c}_1)$ and for each atom $P_j(\vec{w}_j)$ in $conj(\vec{x}; \vec{y})$, the inclusion $P_j(\vec{a}_j) \in \mathcal{A}$ holds, where \vec{a}_j is a projection of \vec{c} and \vec{c}_1 on the vector of variables \vec{w}_j . Using this fact one can construct the following derivation tree for q that consists of n derivation steps and derives q from \mathcal{A} :

$$\begin{array}{c}
 ri \frac{-P_1(\vec{w}_1) \vee \dots \vee \neg P_n(\vec{w}_n) \quad P_n(\vec{a}_n)}{ri \frac{(-P_1(\vec{w}_1) \vee \dots \vee \neg P_{n-1}(\vec{w}_{n-1}))[\vec{w}_n/\vec{a}_n] \quad \{\vec{w}_n/\vec{a}_n\}}{(-P_1(\vec{w}_1) \vee \dots \vee \neg P_{n-2}(\vec{w}_{n-2}))[\vec{w}_n/\vec{a}_n, \vec{w}_{n-1}/\vec{a}_{n-1}] \quad P_{n-1}(\vec{a}_{n-1}) \quad \{\vec{w}_{n-1}/\vec{a}_{n-1}\}}}} \\
 \dots \\
 ri \frac{(-P_1(\vec{w}_1) \vee \neg P_2(\vec{w}_2))[\vec{w}_n/\vec{a}_n, \dots, \vec{w}_3/\vec{a}_3] \quad P_2(\vec{a}_2) \quad \{\vec{w}_2/\vec{a}_2\}}{ri \frac{(-P_1(\vec{w}_1))[\vec{w}_n/\vec{a}_n, \dots, \vec{w}_2/\vec{a}_2] \quad P_1(\vec{a}_1) \quad \{\vec{w}_1/\vec{a}_1\}}{(\perp)[\vec{w}_n/\vec{a}_n, \dots, \vec{w}_1/\vec{a}_1]}}}}
 \end{array}$$

Note, that all the rules that are used in the derivation are atom introduction rules. □

Let us consider a shorn-hand notation $\mathcal{D}up(Goal', Literal, n, Goal)$ for a derivation that takes a goal $Goal'$ as an input and then makes n duplications of the literal $Literal$ in $Goal'$. The output of the derivation is the goal $Goal$.

Let \mathcal{K} be a KB and q a CQ. Let I_{m+1} be an interpretation obtained on the m -th step of a chase sequence $chase(\mathcal{K})$ and I_m be the one obtained on the $m - 1$ -th step. Assume I_{m+1} is obtained from I_m by a propagation of some constant from the basic concept B' to B . In the second lemma we show that if q

1. can be homomorphically mapped to I_{m+1} ,
2. can not be homomorphically mapped I_m ,

then the query q can be *normalized wrt the level $m + 1$* of the $chase(\mathcal{K})$. Where q can be *normalized wrt the level $m + 1$* of $chase(\mathcal{K})$ if one can provide a query q' (a *normalization* of q wrt the level $m + 1$), such that it satisfies the following conditions:

- (a) q can be derived from q' by applying atom duplication, and/or variable introduction, and/or generalization rules,
- (b) q' can be mapped to I_{m+1} but not to I_m ,
- (c) there is a query q'' such that q' can be directly derived from q'' by applying a propagation rule with $\text{horn}(B' \sqsubseteq B)$ and q'' can be homomorphically mapped to I_m .

We call the query q'' as a *direct descendent of q'* , i.e. of a normalization of q , and denote it as $\text{desc}(q')$. Note, that the condition (b) means that the tableau $T(q')$ is a homomorphic image of the tableau $T(q)$ after “cleaning” of the image from duplications of tuples. The condition (c) means that there is only one atom in q' that can not be mapped to I_m , moreover, the extension of I_m to I_{m+1} provides such an interpretation of this atom, that it allows to map q to I_{m+1} . Since there is only one such an atom, the query q' is the shortest one that satisfies (a) and (b) and this is actually the reason why we called q' a normalization of q .

Lemma 5. *Let $q(\vec{x})$ be a CQ; $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a satisfiable KB; \vec{c} be a vector of constants. Let $\text{chase}^n(\mathcal{K})$, where $n \leq \infty$, be a chase of $I(\mathcal{A})$ with \mathcal{K} of the depth n . Let $I_m \xrightarrow{e_m h_m \mathcal{K}} I_{m+1}$ be the m -th step of the chase sequence that ends with $\text{chase}^n(\mathcal{K})$ (where $m \leq n$; the assertion e is a positive inclusion assertion equal to $B' \sqsubseteq B$; the mapping h_m is a homomorphism; and I_m, I_{m+1} are interpretations).*

If the following holds:

$$I_{m+1}(\mathcal{K}) \models q(\vec{c}), \text{ but } I_m(\mathcal{K}) \not\models q(\vec{c}), \quad (4.4)$$

then

1. q has the corresponding EHC of the form: $\text{horn}(q) = \Gamma \vee \neg \text{lit}(B, t, s)$, where s is a variable; t is a term; and Γ is a CQ goal;
2. there exists a query q' which is a normalization of q .

Proof. Assume the conditions of the lemma holds. We will first show that $\text{horn}(q) = \Gamma \vee \neg \text{lit}(B, t, s)$ and then we construct the required q' , which is a normalization of q , i.e. q' satisfies the following conditions:

1. the body of q' satisfies the conditions (4.4),
2. $\text{horn}(q') = \Gamma' \vee \neg \text{lit}(B, t, s)$, where Γ' is a CQ goal,
3. there is a derivation \mathcal{D} of the form:

$$q' \vdash_h^* q$$

where all derivation steps are duplication, and/or variable introduction, and/or generalization rules.

4. there is a query q'' such that q' is directly derived from q'' by application of a propagation rule with $\text{horn}(B' \sqsubseteq B)$ and $I_m \models q''(\vec{c})$.

Proof of the part 1. Let us show that the EHC that corresponds to the query q is *not* of the form $horn(q) = \Gamma \vee \neg lit(B, t, s)$. Since the condition (4.4) holds, there is at least one atom in the body of q , which is not true in I_m , but true in I_{m+1} . The difference between I_m and I_{m+1} is in the interpretation of one atom $lit(B, r, w^e)$ only, where r and w^e are variables. Hence, this atom should occur in the body of q at least once. Hence, $horn(q(\vec{c})) = \Gamma \vee \neg lit(B, t, s)$ holds. Assume that $lit(B, t, s) = P(t, s)$.

Proof of the part 2. Consider a multiset D of atoms that occur in $q(\vec{c})$, which is defined as follows: (i) all atoms from D are of the form $P(t, s)$; (ii) if one delete all atoms of D from $q(\vec{c})$ then the resulting conjunction, say $conj$, is true in I_m and (iii) $conj \wedge P(t, s)$ for any $P(t, s)$ from D is false in I_m . Since (4.4) holds such a set D exists. This set essentially contains all atoms from $q(\vec{c})$ that does not allow to map $q(\vec{c})$ to I_m . The set D consists of $k \in [1, \dots, |D|]$ pairwise disjoint multisets of atoms $D = D_1 \cup \dots \cup D_l$, where each multiset D_k consists of n_k atoms of the form $P(t_{i,k}, s_k)$, $i \in [1, \dots, n_k]$ and for different D_i, D_j the variables s_i and s_j are different.

Observe that if an atom $P(t, s)$ is in D , then the variable s can not appear in atoms of $conj$. Indeed, assume that s occurs in an atom $P_1(r, s)$, where r is a term, and this atom occurs in $conj$. By the definition of $P(t, s)$ and I_{m+1} , $I_{m+1} \models P(a, b)$, for some constants a and $b \in ln(I_m, \mathcal{K})$. By the definition of $conj$, $I_m \models P_1(a', n)$ holds, for some variable a' . This entailment contradicts with the fact that $n \in ln(I_m, \mathcal{K})$. As a consequence, we note that s can not appear in atoms with predicate symbols different from P .

There are three possibilities for the shape of the basic concept B and, consequently, for $P(t, s)$. They are the following:

1. $B = A$ and $lit(B, t, s) = P(t, s) = A(t)$,
2. $B = \exists R$ and $lit(B, t, s) = P(t, s) = R(t, s)$,
3. $B = \exists R^-$ and $lit(B, t, s) = P(t, s) = R(s, t)$.

let us consider all the cases separately.

Case 1. Assume $B = A$ and $lit(B, t, s) = P(t, s) = A(t)$. In this case each D_i consists of n_i atoms of the form $A(t)$ and some of them can coincide. Hence, D consists of $|D|$ atoms of the form $A(t)$ and some of them can coincide. Assume in D there are k_1 atoms $A(t_1)$, \dots , k_p atoms $A(t_p)$, where $k_1 + \dots + k_p = |D|$, i.e.

$$D = \{A(t_1), \dots, A(t_1), A(t_2), \dots, A(t_2), \dots, A(t_{k_p}), \dots, A(t_{k_p})\}.$$

Hence, the goal $horn(q)$ is of the form:

$$horn(q) = \Gamma' \vee \neg A(t_1) \vee \dots \vee \neg A(t_1) \vee \neg A(t_2) \vee \dots \vee \neg A(t_2) \vee \dots \vee \neg A(t_{k_p}) \vee \dots \vee \neg A(t_{k_p}),$$

where Γ' is a CQ goal. Now we can easily find a normalization of q , namely q' . It is the goal with the corresponding EHC equal to $horn(q') = \Gamma' \vee \neg A(t)$. The derivation tree corresponding to $q' \vdash_h^* q$ is the following:

$$\begin{array}{c}
\frac{\mathcal{D}up(\Gamma'_{|D|}, \neg A(t_{k_p}), n_{k_p}, \text{horn}(q))}{\dots} \\
\frac{\mathcal{D}up(\Gamma'_2, \neg A(t_2), k_2, \Gamma'_3)}{\mathcal{D}up(\Gamma'_1, \neg A(t_1), k_1, \Gamma'_2)} \\
\text{cd} \frac{\Gamma' \vee \neg A(t_1) \vee \neg A(t_2) \vee \dots \vee \neg A(t_{k_p})}{\Gamma' \vee \neg A(t)} [t_1/t, \dots, t_{k_p}/t] \\
\text{gen}
\end{array}$$

where $\Gamma'_1 = \Gamma' \vee \neg A(t_1) \vee \neg A(t_2) \vee \neg A(t_{k_p})$. In this derivation we first generate the goal $\neg A(t_1) \vee \neg A(t_2) \vee \neg A(t_{k_p})$ from $A(t)$ using the homomorphism $h = \{t_1/t, \dots, t_{k_p}/t\}$. Then we just apply the concept duplication rule to each literal in the the generated goal as many times as it is required to obtain $\text{horn}(q)$.

Case 2. Assume $B = \exists R$ and $\text{lit}(B, t, s) = P(t, s) = R(t, s)$. In this case q' is the goal with the corresponding EHC equal to $\text{horn}(q') = \text{horn}(\text{conj}) \vee \neg R(t, s)$. We will construct a derivation $q' \vdash_h^* q$ in a similar to *Case 1.* manner. Let us consider a multiset D_k in details. It consists of n_k atoms of the form $P(t_{i,k}, s_k)$, $i \in [1, \dots, n_k]$. Let $D_{k,1} \cup \dots \cup D_{k,m_k}$ be a partition of D_k into m_k multisets such that each $D_{k,j}$ consists of $\alpha_{k,j}$ repetition of an atom $P(t_{j,k}, s_k)$. Now we will construct a derivation $q' \vdash_h^* q$, we denote it as Der , which is a sequence of consecutive sequences of derivations $\text{Der}_1(\Gamma_0, \Gamma_1), \dots, \text{Der}_l(\Gamma_{l-1}, \Gamma_l)$, where $\text{horn}(q') = \Gamma_0$ and $\text{horn}(q) = \Gamma_l$. Each derivation $\text{Der}_j(\Gamma_{j-1}, \Gamma_j)$

- corresponds to the multiset D_j ;
- takes the CQ goal Γ_{j-1} as an input and returns the CQ goal Γ_j as an output;
- works as follows: it extends Γ_{j-1} to Γ_{j+1} by adding negations of all atoms from D_j . The extension is done in two steps:
 1. using a homomorphism it adds m_j literals of the form $\neg P(t_{i,j}, s_j)$ to Γ_{j-1} , i.e. one literal from each multiset $D_{j,i}$.
 2. using concept duplication rule it duplicates $\alpha_{j,i}$ times each added literal $\neg P(t_{i,j}, s_j)$.

Consider the derivation tree corresponding to $\text{Der}_j(\Gamma_{j-1}, \Gamma_j)$

$$\begin{array}{c}
\frac{\mathcal{D}up(\Gamma'_{m_j}, \neg R(t_{m_j,j}, s_j), \alpha_{j,m_j}, \Gamma_j)}{\dots} \\
\frac{\mathcal{D}up(\Gamma'_2, \neg R(t_{2,j}, s_j), \alpha_{j,2}, \Gamma'_3)}{\mathcal{D}up(\Gamma'_1, \neg R(t_{1,j}, s_j), \alpha_{j,1}, \Gamma'_2)} \\
\text{rd} \frac{\Gamma_{j-1} \vee \neg R(t_{1,j}, s_j) \vee \dots \vee \neg R(t_{m_j,j}, s_j)}{\Gamma_{j-1}} [t_{j,1}/t_{j-1,1}, \dots, t_{j,m_j}/t_{j-1,1}] \\
\text{gen}
\end{array}$$

where $\Gamma'_1 = \Gamma_{j-1} \vee \neg R(t_{j,1}, s_j) \vee \dots \vee \neg R(t_{j,m_j}, s_j)$. The very first derivation $\text{Der}_1(\Gamma_0, \Gamma_1)$, i.e. when $j = 1$, is a bit different from the rest derivations. The different is only in the homomorphism that is in the first step of the derivation, more precisely, the homomorphism is the following: $\{t_{1,1}/t, \dots, t_{1,m_1}/t\}$. The homomorphism h in the derivation $q' \vdash_h^* q$ is a composition of homomorphisms $h_l \circ \dots \circ h_1$, where each h_i obtained in derivation $\text{Der}_i(\Gamma_{i-1}, \Gamma_i)$.

Case 3. Assume $B = \exists R^-$. In this case q' is the goal with the corresponding EHC equal to $\text{horn}(q') = \text{horn}(\text{conj}) \vee \neg R(s, t)$. This can be proved similarly to the *Case 2*.

The only part of the lemma which is not proved is that there is a $\text{desc}(q') = q''$ such that $I_m \models q''(\vec{c})$. Recall that q' is of the form either $\Gamma \vee \neg A(t)$ or $\Gamma \vee \neg R(t, s)$ or $\Gamma \vee \neg R(s, t)$, depending on the form of the basic concept B . Let \tilde{q} be the CQ that corresponds to the CQ goal Γ . As it was shown above, \tilde{q} satisfies the condition $I_m \models \tilde{q}(\vec{c})$. Consider the following derivations:

$$\begin{aligned} \text{pr} : & \frac{\Gamma \vee A(s) \quad A(w^u) \vee \neg \text{lit}(B', w^u, w_2^u)}{\Gamma \vee \neg \text{lit}(B', s, t)}; \\ \text{pr} : & \frac{\Gamma \vee R(t, s) \quad R(w^u, w_1^e) \vee \neg \text{lit}(B', w^u, w_2^u)}{\Gamma \vee \neg \text{lit}(B', s, t)}; \quad \frac{\Gamma \vee R(s, t) \quad R(w_1^e, w^u) \vee \neg \text{lit}(B', w^u, w_2^u)}{\Gamma \vee \neg \text{lit}(B', s, t)}. \end{aligned}$$

The goals in the premisses of the derivations are all instances of q'' depending on the form of B . By the definition of the chase step $I_m \xrightarrow{e_m h_m \mathcal{K}} I_{m+1}$, the entailment $I_m \models \text{lit}(B', s, t)$ holds, consequently, $I_m \models q''(\vec{c})$ holds. □

Now we are ready to show completeness of the EHC calculus. For a given query q and a KB \mathcal{K} the proof consists of applying (i) of Lemma 5, first to q , then to $\text{desc}(q')$ and so on recursively until we reach a query that can be homomorphically mapped to the ABox of \mathcal{K} and then (ii) of Lemma 4. Detailed proof is the following.

Theorem 9. (*Completeness Theorem*)

Let \mathcal{K} be a KB; $q(\vec{x})$ be a CQ over \mathcal{K} ; and θ be a substitution. If the following entailment holds:

$$\mathcal{K} \models q(\vec{c}), \text{ where } \vec{c} = \theta(\vec{x})$$

then the following derivation exists:

$$\mathcal{K} \vdash_{\gamma}^* \text{horn}(q)$$

where $\gamma = \theta \cup h$, for some homomorphism h .

Proof. Assume $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a satisfiable KB; $q(\vec{x})$ is a CQ over \mathcal{K} ; and θ is a substitution.

Assume the entailment $\mathcal{K} \models q(\vec{c})$, where $\vec{c} = \theta(\vec{x})$, holds. We are to show that there exists a derivation $\text{Der} = \mathcal{K} \vdash_{\theta}^* \text{horn}(q)$. Since, for any satisfiable KB conjunctive query answering can be performed over its universal model (see Theorem 6) and chase is a universal model (see Theorem 4) we conclude that the entailment $\text{chase}(\mathcal{K}) \models q(\vec{c})$ holds. Recall that $\text{chase}(\mathcal{K})$ is defined as an interpretation obtained after several (possibly infinitely many) extensions of $I(\mathcal{A})$ using positive inclusion assertions of \mathcal{K} . Assume that $\text{chase}(\mathcal{K}) = \text{chase}^k(\mathcal{K})$, where $k \leq \infty$. Since the formula $\text{body}(q)$ consists of finitely many conjuncts, there is a natural number m which is not more than n (or strictly less than n in a case of infinite chase) such that $\text{chase}^m(\mathcal{K}) \models q(\vec{c})$.

If $m = 0$, i.e. if $I(\mathcal{A}) \models q(\vec{c})$ holds, then by Lemma 4 there is a derivation $\mathcal{A} \vdash_{\theta}^* \text{horn}(q)$ and, consequently, $\mathcal{K} \vdash_{\theta}^* \text{horn}(q)$.

If $m \neq 0$, i.e. if $I(\mathcal{A}) \not\models q(\vec{c})$ holds, then let us choose the minimal m such that it satisfies the condition (4.4), i.e.:

$$\text{chase}^m(\mathcal{K}) \models q(\vec{c}), \text{ but } \text{chase}^{m-1}(\mathcal{K}) \not\models q(\vec{c}).$$

Assume that $\text{chase}^m(\mathcal{K})$ is obtained (on the m -th chase step) after applying of a positive inclusion assertion $B' \sqsubseteq B$ to $\text{chase}^{m-1}(\mathcal{K})$ with some homomorphism. By Lemma 5, there is a normalization of q , say q' , such that $q' \vdash_{h_1}^* q$ and there is $\text{desc}(q') = q''$ such that the following derivation holds:

$$\text{pr} : \frac{\text{horn}(q') \quad \text{horn}(B' \sqsubseteq B)}{\text{horn}(q'')}.$$

Hence, there is a derivation $q'' \vdash_{h_1}^* q$. Let us denote it as $\mathcal{D}er_1$. So, in our reasoning we took q as an input and, since $I(\mathcal{A}) \not\models q(\vec{c})$ holds, we returned the query q'' and the derivation $q'' \vdash_{h_1}^* q$ as an output.

Now, one should apply the same reasoning using q'' as an input and then using the obtained outputs as inputs until the output query can not be mapped to $I(\mathcal{A})$. Assume there should be done k steps of the reasoning all in all in order to obtain a query \tilde{q} , such that $I(\mathcal{A}) \models \tilde{q}(\vec{c})$. Each of these k steps returns a derivation. Let the i -th step returns a derivation $\mathcal{D}er_i$. Consider a derivation which is the sequence of consecutive derivations $\mathcal{D}er_i, \dots, \mathcal{D}er_1$. This derivation is in fact a derivation of q from \tilde{q} of the form $\tilde{q} \vdash_{h_k \circ \dots \circ h_1}^* q$. Let us denote as h the composition $h_k \circ \dots \circ h_1$. Since a composition of homomorphisms is a homomorphism, h is a homomorphism. Note that, by its construction, the derivation $\tilde{q} \vdash_h^* q$ involves k positive inclusion assertions from \mathcal{T} and none membership assertions from \mathcal{A} .

Using Lemma 4 and the fact $I(\mathcal{A}) \models \tilde{q}(\vec{c})$ we conclude that there is a derivation $\mathcal{A} \vdash_{\theta}^* \tilde{q}$. Let us denote it as $\mathcal{D}er_{i+1}$. Using $\mathcal{D}er_{i+1}$ and $\mathcal{D}er_i, \dots, \mathcal{D}er_1$ we obtain the derivation we were looking for, i.e. $\mathcal{K} \vdash_{\theta \cup h}^* q$. The derivation is the sequence of consecutive derivations: $\mathcal{D}er_{i+1}, \mathcal{D}er_i, \dots, \mathcal{D}er_1$. \square

4.3.6 Algorithm to Implement EHC Calculus

Here we show how by imposing a specific control strategy on the EHC calculus one can obtain an algorithm for implementing the calculus proposed by Calvanese et al. in [CDGL⁺05b].

Roughly speaking, the strategy is to derive, starting from a given query, the empty EHC by using a given KB and by applying the derivation rules in a top-down manner. This strategy involves two main stages and can be described as follows: for a given query q and a given KB \mathcal{K} one must

- first, take q and construct all possible derivations in a top-down manner, such that they
 - start with $\text{horn}(q)$
 - use the positive inclusion assertions from \mathcal{K} and all derivation rules except atomic introduction ones.

While constructing the derivations, one must detect loops that can appear in the derivations, i.e. detect the cases when after application of a rule, say r , in a derivation, say $\mathcal{D}er$, the obtained query is exactly the same as a one already obtained before in $\mathcal{D}er$. If such a case detected, then one should either to apply r using another values of the parameters of the rule (e.g. chose another atom to delete a duplication, if r is a duplication rule) or not to apply r at all (if there is no way to find a value that will not lead to a loop).

All the queries obtained in every step of every constructed derivation must be collected in a so-called *set of reformulations* for the given query.

- Second, for each query from the set of reformulations construct all possible derivations in a top-down manner by applying atomic introduction rules and using the ABox of the KB. Each derivation that ends with the empty EHC returns a certain answer for the original query over the KB.

In fact, when Calvanese et al. implemented the first stage, they used a simplified version of the generalization rule. They used only a particular case of homomorphisms, namely unification of two atoms. Another comment is that they completely delegate the derivations involved in the second stage to an SQL-engine. The delegation is possible because the rules that are involved in the second stage are atom introduction ones only. Applications of these rules in a top-down manner to a query is just grounding of variables in the query with constants from $atom(\mathcal{K})$. Thus, if a derivation of this kind for a query q' reaches the empty EHC, then all variables in q are grounded with constants from $atom(\mathcal{K})$. This total grounding of q' is in fact a “pattern matching” of the CQ q' to the ABox of \mathcal{K} and it can be easily done by an SQL-engine.

This very strategy is shown to be complete wrt query answering and implemented in QuOnto system [ACDG⁺05].

Chapter 5

Reduction of Reasoning in DL-Lite \mathcal{F} to CQ Answering

In this chapter we discuss how to reduce the reasoning tasks for *DL-Lite \mathcal{F}* presented in Section 2.3 to CQ answering. In order to do it we will first present a reduction of KB satisfiability to CQ answering and then a reduction of disjointness/subsumption/equivalence checking to KB satisfiability. Details and proofs for all the reductions one can find in [CGL⁺06].

5.1 KB Satisfiability

Satisfiability of a given KB \mathcal{K} can be checked in three steps.

1. Collect all functional assertions of \mathcal{K} in a set S . Only the TBox of \mathcal{K} is involved in this step.
2. Compute all negative inclusion assertions that are entailed by \mathcal{K} . Collect all the obtained negative inclusion assertions in a set S . Only the TBox of \mathcal{K} is involved in this step.
3. Transform each assertion from S to a conjunctive query in the following way:

Assertion	Query
$funcR$	$\exists x, y, z. R(x, y) \wedge R(x, z) \wedge y \neq z$
$funcR^-$	$\exists x, y, z. R(y, x) \wedge R(z, x) \wedge y \neq z$
$B \sqsubseteq \neg B'$	$\exists x, y, z. lit(B, x, y) \wedge lit(B', x, z)$

Take the union of the obtained conjunctive queries. Evaluate the union over the ABox of \mathcal{K} stored as a relation DB, by delegating of the union to a RDBMS. If the evaluation returns “false”, then the KB is satisfiable; otherwise the KB is unsatisfiable. Only the ABox of \mathcal{K} is involved in this step.

The computation of all negative inclusion assertions entailed by \mathcal{K} in Step 2. can be done in polynomial time in the size of the TBox of \mathcal{K} . The following recursive algorithm describes a way to compute it.

1. Collect all negative inclusion assertion in a set N .
2. Take an arbitrary assertion, say $B \sqsubseteq B'$, from the TBox of \mathcal{K} . If there is an assertion in N that says that B' is disjoint with some B'' , i.e. if either $B'' \sqsubseteq \neg B'$ or $B' \sqsubseteq \neg B''$ is in N , then check whether $B \sqsubseteq \neg B''$ is in N and if it not the case then add $B \sqsubseteq \neg B''$ to N .
3. If there is an assertion of the form $\exists R \sqsubseteq \neg \exists R$ in N , then check whether there is the assertion $\exists R^- \sqsubseteq \neg \exists R^-$ in N and, if it is not the case, then add $\exists R^- \sqsubseteq \neg \exists R^-$ to N . Analogously, if $\exists R^- \sqsubseteq \neg \exists R^-$ is in N , but $\exists R \sqsubseteq \neg \exists R$ is not there, then one should add the latter assertion in N .
4. Repeat executions of the steps 2. and 3. as long as they keep extending the set N . If a sequential execution of the steps 2. and 3. does not change N , then terminate and return N .

Note that in Step 3. the assertion $\exists R \sqsubseteq \neg \exists R$ reflects the fact that the relation R has no elements on its first coordinate in any model of \mathcal{K} , or, equivalently, R is empty in any model of \mathcal{K} . The emptiness of R , obtained because of the “emptiness” of R 's first coordinate, implies “emptiness” of the R 's second coordinate, i.e. it implies $\exists R^- \sqsubseteq \neg \exists R^-$. This is the reason why one need to make Step 3. in order to compute all negative inclusion assertions implied by \mathcal{K} .

5.2 Instance checking

For a given KB \mathcal{K} , a (general) concept C /role R and a constant a /a pair of constants a, b the instance checking problem is to verify whether $I \models C(a)/I \models R(a, b)$ holds for every model $I \models \mathcal{K}$.

This problem can be easily reduced to a KB satisfiability problem in the following way. Consider a new KB $\mathcal{K}_{C(a)} = \mathcal{K} \cup \{\{A \sqsubseteq \neg C\}, \{A(a)\}\}$, where A is a new atomic concept, i.e. it does not occur in \mathcal{K} . In [CGL⁺06] it is shown that $\mathcal{K} \models C(a)$ if and only if $\mathcal{K}_{C(a)}$ is unsatisfiable. Thus, we reduced the instance checking problem to the KB satisfiability problem which is turn reducible to the CQ answering. The composition of this two reductions gives us a reduction of the instance checking to the CQ answering.

Regarding the instance checking for roles, it can be reduced to the KB satisfiability similarly to the case of concepts, the only difference is that one need to use a new role P and a pair of constants a and b , instead of a new concept A and a constant a .

5.3 Subsumption Checking

For a given KB \mathcal{K} and two (general) concepts C, D the subsumption problem is to verify whether $I \models C \sqsubseteq D$ holds for every model $I \models \mathcal{K}$.

This problem can be easily reduced to a KB satisfiability problem in the following way. Consider a new KB $\mathcal{K}_{C \sqsubseteq D} = \mathcal{K} \cup \{\{A_1 \sqsubseteq C, A_2 \sqsubseteq \neg D\}, \{A_1(a), A_2(a)\}\}$, where a is a constant that does not appear in \mathcal{K} , A_1 and A_2 are atomic concepts that do not appear in \mathcal{K} . In [CGL⁺06] it is shown that $C \sqsubseteq D$ if and only if $\mathcal{K}_{C \sqsubseteq D}$ is unsatisfiable. Thus, we reduced the subsumption problem to the KB satisfiability problem which is turn reducible to the CQ answering. The composition of this two reductions gives us a reduction of the instance checking to the CQ answering.

5.4 Equivalence Checking

For a given KB \mathcal{K} and two (general) concepts C, D the equivalence problem is to verify whether $I \models C \sqsubseteq D$ and $I \models D \sqsubseteq C$ hold for every model $I \models \mathcal{K}$. Thus, this problem just consists of two subsumption problems, which are both reducible to the CQ answering.

Chapter 6

Conclusions

In this thesis we have studied several properties of query answering over ontologies expressed in $DL-Lite_{\mathcal{F}}$ ontology language first introduced in [CDGL⁺05b]. Specifically, we have introduced the notion of universal model for a KB, which has roots in universal solutions proposed by Fagin et al. in [FKMP05] for the data-exchange setting. We show that any satisfiable $DL-Lite_{\mathcal{F}}$ KB has a universal model. Universal models for a $DL-Lite_{\mathcal{F}}$ KB could be infinite in principle, thus, the results obtained in the thesis extend the ones in [FKMP05]. In particular, Fagin et al. showed that a finite chase constructed for the data-exchange problem is a universal solution, while we show that even an infinite chase constructed for a KB is a universal model. We adopt the notion of weakly-acyclic tgds for the $DL-Lite_{\mathcal{F}}$ setting and this gives us the class of so-called weakly-acyclic KBs which have finite chases of polynomially bounded sizes.

The most interesting feature of the universal models is that answering conjunctive queries can be based on them. If one wants to evaluate a CQ over a satisfiable KB, one can simply do it over a chase of the KB, which is in fact a universal model of the KB. This way of answering queries can be used for weakly-acyclic KBs but not for arbitrary KBs, because of potential infinity of chases. It turns out that even if there are no finite universal models for a KB, one can still perform answering CQs over the KB by means of the calculus that is proposed in the thesis for the so-called Extended Horn Logic.

We present Extended Horn Logic as an extension of the standard functional free \mathcal{FOL} Horn Logic by allowing free and existentially quantified variables in the clauses. The calculus proposed for the logic is sound and complete wrt to answering CQs, i.e. for a given KB \mathcal{K} and a CQ q the calculus allows one to derive all answers for q over \mathcal{K} . In order to use the calculus for query answering, one needs to translate \mathcal{K} and q into Extended Horn Logic (the translation is proposed in the thesis) and then to apply the calculus. Since there are many ways to derive all answers for q over \mathcal{K} , we present a specific control strategy on the calculus. The strategy gives an algorithm to implement query answering. This algorithm in fact resembles the one proposed by Calvanese et al. in [CDGL⁺05b] and implemented in the QuOnto system [ACDG⁺05].

So, $DL-Lite_{\mathcal{F}}$ can be used as an ontology language for data integration if the data sources are relational databases. Moreover, answering conjunctive queries, posed to a data integration systems that uses $DL-Lite_{\mathcal{F}}$ ontology, can be performed extremely efficiently.

The further work is to understand whether it is possible to use wider fragments of SQL for answering queries in the $DL-Lite_{\mathcal{F}}$ setting without loss of efficiency, i.e. still keeping delegation of the query evaluation to SQL engines. This thesis gives a better understanding of theoretical properties of $DL-Lite_{\mathcal{F}}$ logic and the obtained results will be used for further investigations on query languages for $DL-Lite_{\mathcal{F}}$.

Bibliography

- [ACDG⁺05] Andrea Acciari, Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Mattia Palmieri, and Riccardo Rosati. QUONTO: Querying ONTOlogies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 1670–1671, 2005.
- [ACHK93] Y. Arens, C. Y. Chee, C. Hsu, and C. A. Knoblock. Retrieving and integrating data from multiple information sources. *J. of Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.
- [AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., 1995.
- [AKS96] Y. Arens, C. A. Knoblock, and W. Shen. Query reformulation for dynamic information integration. *J. of Intelligent Information Systems*, 6:99–130, 1996.
- [BB03] Alexander Borgida and Ronald J. Brachman. Conceptual modeling with description logics. In Baader et al. [BCM⁺03], chapter 10, pages 349–372.
- [BBMR89] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. CLASSIC: A structural data model for objects. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 59–67, 1989.
- [BCDG01] Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo. Reasoning on UML class diagrams using description logic based systems. In *Proc. of the KI'2001 Workshop on Applications of Description Logics*. CEUR Electronic Workshop Proceedings, <http://ceur-ws.org/Vol-44/>, 2001.
- [BCM⁺03] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
- [BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.
- [BLN86] Carlo Batini, Maurizio Lenzerini, and Shamkant B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

- [BS85] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.
- [BV84] Catriel Beeri and Moshe Y. Vardi. A proof procedure for data dependencies. *J. of the ACM*, 31(4):718–741, 1984.
- [BvHH⁺04] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language reference. W3C Recommendation, February 2004. Available at <http://www.w3.org/TR/owl-ref/>.
- [CCDGL01] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Reasoning on UML class diagrams in description logics. In *Proc. of IJCAR Workshop on Precise Modelling and Deduction for Object-oriented Software Development (PMD 2001)*, 2001.
- [CCDGL02] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Data integration under integrity constraints. In *Proc. of the 14th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2002)*, volume 2348 of *Lecture Notes in Computer Science*, pages 262–279. Springer, 2002.
- [CDGL⁺98a] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Description logic framework for information integration. In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'98)*, pages 2–13, 1998.
- [CDGL⁺98b] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Information integration: Conceptual modeling and reasoning support. In *Proc. of the 6th Int. Conf. on Cooperative Information Systems (CoopIS'98)*, pages 280–291, 1998.
- [CDGL⁺05a] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. In *Proc. of the 2005 Description Logic Workshop (DL 2005)*. CEUR Electronic Workshop Proceedings, <http://ceur-ws.org/Vol-147/>, 2005.
- [CDGL⁺05b] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.
- [CDGL⁺05c] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tailoring OWL for data intensive ontologies. In *Proc. of the Workshop on OWL: Experiences and Directions (OWLED 2005)*, 2005.
- [CGL⁺06] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The dl-lite family. *Submitted to an international journal*, 2006.

- [CGMH⁺94] Sudarshan S. Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey D. Ullman, and Jennifer Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proc. of the 10th Meeting of the Information Processing Society of Japan (IPSJ'94)*, pages 7–18, 1994.
- [CHS91] Christine Collet, Michael N. Huhns, and Wei-Min Shen. Resource integration using a large knowledge base in Carnot. *IEEE Computer*, 24(12):55–62, 1991.
- [dB03] Jos de Bruijn. Using ontologies - enabling knowledge sharing and reuse on the semantic web. Technical Report DERI-2003-10-29, DERI, Austria, 2003.
- [FAD⁺00] Dieter Fensel, Jürgen Angele, Stefan Decker, Michael Erdmann, Hans-Peter Schnurr, Rudi Studer, and Andreas Witt. Lessons learned from applying AI to the web. *Int. J. Cooperative Inf. Syst.*, 9(4):361–382, 2000.
- [FDES98] Dieter Fensel, Stefan Decker, Michael Erdmann, and Rudi Studer. Ontobroker: The very high idea. In Diane J. Cook, editor, *FLAIRS Conference*, pages 131–135. AAAI Press, 1998.
- [Fel98] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
- [FKMP05] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. *Theoretical Computer Science*, 336:89–124, 2005.
- [GM95] A. Gupta and I. S. Mumick. Maintenance of materialized views: Problems, techniques, and applications. *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, 18(2):3–18, 1995.
- [Gru92] T. R. Gruber. Ontolingua: A mechanism to support portable ontologies. Technical report, Stanford University Knowledge Systems Laboratory, 1992.
- [Gru93] T. R. Gruber. A translation approach to portable ontology specification. *Knowledge Acquisition*, 5(2):199–220, 1993.
- [HBP94] A. R. Hurson, M. W. Bright, and S. Pakzad, editors. *Multidatabase Systems: An Advanced Solution for Global Information Sharing*. IEEE Computer Society Press, 1994.
- [HFB⁺00] I. Horrocks, D. Fensel, J. Broekstra, S. Decker, M. Erdmann, C. Goble, F. van Harmelen, M. Klein, S. Staab, R. Studer, and E. Motta. OIL: The Ontology Inference Layer. Technical Report IR-479, Vrije Universiteit Amsterdam, Faculty of Sciences, September 2000.
- [HGMW⁺95] Joachim Hammer, Hector Garcia-Molina, Jennifer Widom, Wilburt Labio, and Yue Zhuge. The Stanford data warehousing project. *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, 18(2):41–48, 1995.
- [HH00] Jeff Heflin and James Hendler. Dynamic ontologies on the web. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, Menlo Park, California, 2000. AAAI/MIT Press.

- [HJK⁺93] Michael N. Huhns, Nigel Jacobs, Tomasz Ksiezyk, Wei-Min Shen and Munindar P. Singh, and Philip E. Cannata. Integrating enterprise information models in Carnot. In *Proc. of the Int. Conf. on Cooperative Information Systems (CoopIS'93)*, pages 32–42, 1993.
- [Hor02] Ian Horrocks. DAML+OIL: a description logic for the Semantic Web. *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, 25(1):4–9, 2002.
- [Hul97] Richard Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'97)*, pages 51–61, 1997.
- [HZ96] Richard Hull and Gang Zhou. A framework for supporting data integration using the materialized and virtual approaches. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 481–492, 1996.
- [Inm96] W. H. Inmon. *Building the Data Warehouse*. John Wiley & Sons, second edition, 1996.
- [JLVV99] Matthias Jarke, Maurizio Lenzerini, Yannis Vassiliou, and Panos Vassiliadis, editors. *Fundamentals of Data Warehouses*. Springer, 1999.
- [KL95] Craig Knoblock and Alon Y. Levy, editors. *Proc. of the AAAI'95 Spring Symp. on Information Gathering from Heterogeneous, Distributed Environments*, number SS-95-08 in AAAI Spring Symposium Series. AAAI Press/The MIT Press, 1995.
- [KLSS95] Thomas Kirk, Alon Y. Levy, Yehoshua Sagiv, and Divesh Srivastava. The Information Manifold. In *Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Environments*, pages 85–91, 1995.
- [Len95] D. B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):32–38, 1995.
- [Len02] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.
- [LL01] Hector J. Levesque and Gerhard Lakemeyer. *The Logic of Knowledge Bases*. The MIT Press, 2001.
- [Llo87] John W. Lloyd. *Foundations of Logic Programming (Second, Extended Edition)*. Springer, Berlin, Heidelberg, 1987.
- [LRO96] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Query answering algorithms for information agents. In *Proc. of the 13th Nat. Conf. on Artificial Intelligence (AAAI'96)*, pages 40–47, 1996.
- [LSK95] Alon Y. Levy, Divesh Srivastava, and Thomas Kirk. Data model and query evaluation in global information systems. *J. of Intelligent Information Systems*, 5:121–143, 1995.

- [SL90] Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.
- [TS03] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Cambridge University Press, Cambridge, England, 2 edition, 2003.
- [Ull97] Jeffrey D. Ullman. Information integration using logical views. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 1997.
- [Var82] Moshe Y. Vardi. The complexity of relational query languages. In *Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC'82)*, pages 137–146, 1982.
- [vdM98] Ron van der Meyden. Logical approaches to incomplete information. In Jan Chomicki and Günter Saake, editors, *Logics for Databases and Information Systems*, pages 307–356. Kluwer Academic Publisher, 1998.
- [WGL⁺96] Janet L. Wiener, Himanshu Gupta, Wilburt J. Labio, Yue Zhuge, Hector Garcia-Molina, and Jennifer Widom. A system prototype for warehouse view maintenance. Technical report, Stanford University, 1996. Available at <http://www-db-stanford.edu/warehousing/warehouse.html>.
- [Wie92] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.
- [Wie96] Gio Wiederhold (ed.). Special issue: Intelligent integration of information. *J. of Intelligent Information Systems*, 6(2/3), 1996.
- [ZGMHW95] Yue Zhuge, Hector Garcia-Molina, Joachim Hammer, and Jennifer Widom. View maintenance in a warehousing environment. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 316–327, 1995.
- [ZHK96] Gang Zhou, Richard Hull, and Roger King. Generating data integration mediators that use materializations. *J. of Intelligent Information Systems*, 6:199–221, 1996.