unibz
Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

Reasoning Over Description Logic Ontologies under non-standard Assumptions

Dissertation

submitted in accordance with the requirements of
the Free University of Bozen-Bolzano
for the degree of Doctor in Philosophy by

Yazmín Angélica Ibáñez García

Free University of Bozen-Bolzano

Dissertation advisors:

Prof. Alessandro Artale
KRDB Research Centre
Free University of Bozen-Bolzano

Prof. Dr. Carsten Lutz
AG Theorie der künstlichen Intelligenz
Universität Bremen

Dissertation Reviewers:

Prof. Riccardo Rosati
Sapienza Università di Roma

Prof. Ulrike Sattler
University of Manchester

Examination Committee:

Prof. Michael Zakharyashev (chair)
Birbeck College, University of London

Prof. Sebastian Rudolph
Technische Univesität Dresden

Prof. Johann Gamper (secretary)
Free University of Bozen-Bolzano

Date of public defense: 24. 10. 2014

Abstract

The use of *description logics (DLs)* ontologies to provide a formal and general-purpose conceptual model of a domain of interest has become a popular choice due to the increasing need to add a semantic dimension to information processing. In that context, ontologies constitute an ideal tool to provide a conceptualization of the domain of interest in areas such as Enterprise Application Integration, Data Integration, and the Semantic Web. Nevertheless, *conceptual models* are still usually designed in some class-based language such as *UML diagrams* or *ER schemata* because of their intuitive and user-friendly interface. The downside of using graphical languages is, however, the lack of formal semantics. Formalizing the semantics of conceptual modeling languages in DLs has the advantage of keeping the graphic interface for modeling, while providing *reasoning capabilities* that aid to verify the quality of the conceptual models, as well as for performing *reasoning*, that is, to infer implicit knowledge from the explicitly represented one. Indeed, those reasoning capabilities can not only be used to infer subsumption relationships between *classes* in a conceptual model or to verify the consistency of the model itself, but also to provide a formalization for *incomplete databases*. In the latter context, DLs TBoxes can be understood as database schema languages, ABoxes as a representation of the (incomplete) data, and *query answering* as the main *reasoning service*. Unfortunately, the semantic desiderata in the mentioned applications are not fully matched on the DLs side. In particular, the common assumption in database applications is that the intended models (i.e., database instances) are finite. However, this is by no means the usual assumption in DLs mainly because traditional DL languages have the *finite model property* (FMP). Languages supporting expressive constraints, on the other hand, lack the FMP. This means that using description logics for reasoning in the latter applications amounts to perform *finite model reasoning*. This task, however, has been shown to be difficult from the algorithmic view point. One of the main objectives of this thesis is to investigate finite model reasoning in DLs. Our results concern the so-called Horn DLs, which are known for having good model theoretical properties, and although they are not able to express disjunction (covering constraints), they have still enough expressive power to capture interesting modeling constraints, such as isa relationships between classes and relations, and disjointness. Besides standard reasoning tasks (subsumption, satisfiability), we also investigate *ontological query answering* under the finite model assumption. The second objective of this thesis is to investigate the impact of extending positive existential queries with negation on the computational complexity of ontological query answering over Horn ontologies. The importance of considering negation stems from the need of many natural queries to express *difference* or *complementation* to retrieve the required information. A well-known fact from database theory is that answering queries with negation is harder than without it, we will then focus on queries allowing for the use of negative atoms in restricted forms: the so-called *safe* and *guarded negation* as well as *inequalities*.

To my parents

"Let me not pray to be sheltered from dangers,
but to be fearless in facing them.

Let me not beg for the stilling of my pain, but
for the heart to conquer it."

–Rabindranath Tagore,
Collected Poems and Plays of Rabindranath Tagore

Acknowledgements

"Gratitude is not only the greatest of virtues, but the parent of all others."

Marcus Tullius Cicero

More than four years of efforts have crystallized in the present dissertation. However, this was only possible thanks to the influence, inspiration and support from so many people. I am afraid I cannot name all of them by name, and for that I apologize.

First, I wish to express my gratitude to my supervisor, Alessandro Artale, for his support along all the way, as well as for his patience and for never giving up on me. I am particularly grateful to Alessandro for providing me with all the freedom and resources I needed to pursue my academic interests.

I am most grateful to Carsten Lutz, who has become my second supervisor and thanks to whom a significant part of the research reported in this dissertation came to a realization. During the time I had the privilege to work with Carsten, he was always inspiring, and encouraging. Thanks to Carsten's influence, I became a better researcher, although sometimes his stringent style really made me shake in my boots.

I also appreciate the support from Diego Calvanese, who provided resources to fund the fourth year of my PhD. Diego and his wife Paola were also kind enough to let me live in their house while they were away. That really made my life a lot easier in Bolzano.

Much of this work would not have been possible without the assistance of Thomas Schneider. I thank Thomas for his insightful comments and all the discussions, I have learned a great deal, academic-wise, while working together. I also acknowledge Thomas' willingness to help, which I consider one of his most valuable qualities as a person.

I would like to show my appreciation to my co-authors Roman Kontchakov and Egor Kostylev for the discussions and work we carried out together, which enriched the research reported in this dissertation (Chapter 6). I specially appreciate Roman's efforts in leading the way on that research, resulting in a Best Student Paper award for one of our papers.

Many thanks to Uli Sattler and Riccardo Rosati, who made an outstandingly dedicated work by reviewing this dissertation. Their insightful comments and suggestions were most appreciated and helped to significantly improve the presentation of the results in this dissertation.

I wish to thank to all the colleagues and friends from the KRDB group at the UNIBZ for the countless moments of entertainment that lightened my PhD journey. I will treasure all the parties, drinks, night outs, poker nights, hiking trips, jokes and meals we shared. My time in Bolzano would not have been the same without my friends: the best room-mate and friend one can ask for, Erika; my loquacious and charismatic mates, Mariano and Martin; and Luis and Naomi, the most adorable couple I had the pleasure to meet.

Thanks to the TDKIers at Uni Bremen: İnanç, Jean, Peter, Stefan, and Thomas. I will always remember the grilling evenings, so keenly promoted by İnanç, the amusing anecdotes vividly narrated by Stefan, and the laughter we all shared during those times. I appreciate the optimism of Jean, the enthusiasm of Peter, and the kindness of Thomas. Thank you guys for making my time in Bremen so memorable.

Finally, I wish to give a special mention to Víctor, who I cherish not only for being a brilliant co-author, but most importantly, for being a loving husband. During the time we have shared our lives so far, he has always managed to bring the best out of me (which is not an easy task, I must say). He has been a true friend, a shoulder to cry on and a driving force. It is thanks to Victor's unconditional love and support that I persevere.

Yazmín Ibáñez García
Bozen–Bolzano, October 2014

Contents

List of Figures

List of Tables

Chapter

1 Introduction

One of the prominent advances in *Knowledge Representation and Reasoning (KR)* is the development of technologies that use semantic means for handling information. In particular, logic-based approaches provide general-purpose knowledge representation frameworks in which inferring implicit knowledge amounts to verifying logical consequences.

Description logics (DLs) [12] are arguably among the most prominent logic-based formalisms for knowledge representation. In a nutshell, DLs can be understood as syntactic variants of modal logics, and thus decidable fragments of *first-order logic (FOL)* with well-defined formal semantics and well-understood model theoretical properties.

In practice, the use of DLs for knowledge representation has reaffirmed itself with the emerging paradigm of *ontology*. An ontology in computer science is intended to provide a formal and general-purpose conceptual model of a domain of interest, which can then be used to add a semantic dimension to information processing within that domain. The use of ontologies as a formal tool to provide a conceptualization of the domain of interest has been recognized and studied in several areas, such as Enterprise Application Integration, Data Integration [19], and the Semantic Web [13]. Nevertheless, the conceptualization of a domain of interest is still usually designed in some class-based language such as *ER schema* or *UML diagrams*. Mainly, because these languages offer a graphical, intuitive and user-friendly interface. For example, the UML class diagram in Figure 1.1 contains classes (e.g., *Food, Dish*) and associations (e.g., *hasIngredient*) between them conceptualizing notions on the domain of *food and nutrition*. The downside of using these kinds of languages is, however, the lack of formal semantics.

Notably, it has been extensively advocated that DLs formalize the semantics of conceptual modeling languages such as the *Entity-Relation Model (ER)* and UML class diagrams; and provide the logical underpinning for the *Web Ontology Language (OWL)*. This means that database schemata and ontologies specified on those conceptual modeling formalisms can be transformed into DL ontologies. And then, one can verify properties which ensure the correctness and optimality of such schemata and ontologies using reasoning in DLs. For instance, verifying

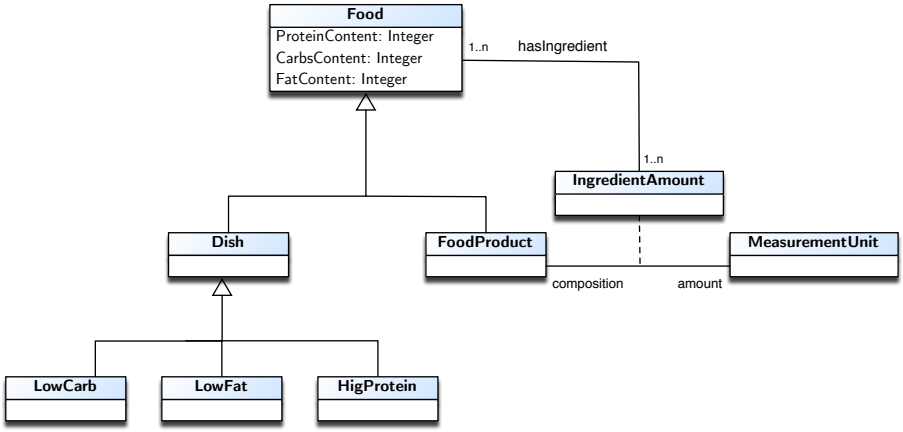


Figure 1.1: Example of an ontology as UML class diagram

that all the constraints specified in a given schema are satisfied for some database instance (*schema consistency*) can be rephrased into checking *satisfiability* of the corresponding DL ontology. Furthermore, by verifying logical entailment w.r.t. the corresponding DL ontology, one could verify whether some constraint in a schema is satisfied.

Unfortunately, the semantic desiderata in conceptual modeling are not fully matched on the DLs side. In particular, the common assumption in database applications is that the intended models (i.e., database instances) are finite. However, this is by no means the usual assumption in DLs. In fact, this assumption is irrelevant when considering traditional description logics such as $\mathcal{ALC}$ since they enjoy the *finite model property (FMP)*, which implies that whenever a DL ontology (specified in $\mathcal{ALC}$) is satisfiable, then it has a *finite* model.

Indeed, in the presence of certain constraints in the modeling formalism, reasoning with respect to finite structures differs from reasoning with respect to unrestricted ones, which corresponds to the fact that certain DLs lack the finite model property. For instance, this holds for the ER model, whose semantics can be formalized by the expressive description logic $\mathcal{ALCQI}$, which allows to express *cardinality constraints* and referring to the *inverse* of relations. In particular, in an ER schema, an inconsistency might arise due to the interaction of cardinality constraints along a *cycle* in the schema and the requirement of the legal databases for the schema to be finite.

Therefore, when using description logics for reasoning in the latter applications one actually should regard reasoning over finite models. Available results on *finite model reasoning* in DLs have shown that this is not a trivial task. In particular, although reasoning over finite models in $\mathcal{ALCQI}$ has the same computational complexity as reasoning over unrestricted (finite or infinite) models [94], the procedures developed for the finite model case are not suitable for providing an algorithmic tool to support some verification tasks in conceptual modeling and ontological reasoning.

There are other typical assumptions in data models that do not have a match on the DLs

side. For example, it is intended that every entity modeled in a schema has at least one instance in every possible instantiation of the schema. This notion has been called *strong consistency* or *full satisfiability* in the literature [78, 90], and needs also to be considered within the approach of reasoning in conceptual models using DLs.

Furthermore, the use of DLs for ontological representation has proved to be successful in data-intensive applications, especially in the context of the Semantic Web and in data integration. In these applications, the typical scenario is that of answering queries over DL ontologies formalizing incomplete databases under the open world semantics. In that case, DLs TBoxes formalize the schema (constraints) of the database, and the ABox the available (known) data. The latter usage of ontologies is referred as *ontology-based data access (OBDA)*. The main reasoning task in the OBDA setting is that of *query answering*, for which the same assumption of finite models applies. As in the case of satisfiability or subsumption, when the logic used to represent the ontology lacks the FMP, *finite query answering* is a different problem than evaluating queries over unrestricted models.

One of the main objective of this thesis is to investigate finite model reasoning in DLs. In particular, we focus on sub-Boolean fragments of expressive DLs. Since the main motivation for the latter investigation is the application of DLs in *data management systems*, we address as well in this thesis the application of our results in DLs to conceptual modeling and OBDA.

Due to their desirable model theoretical properties, positive existential queries, in particular, conjunctive queries (CQs) and union of conjunctive queries, have played a key role in classical database theory. Based on this, most of the research on query answering in the OBDA paradigm has mainly considered such queries. However, it has been noticed in database theory that many natural queries require complementation and difference; for example, to retrieve all 'database researchers that do `not` work in Europe'. Indeed, in the OBDA paradigm only few investigations have considered extensions of CQs with negation. This might be explained by the fact that answering queries with negation in expressive DLs easily becomes undecidable. Some recent investigations provide initial results on answering queries with negation in Horn DLs [116]. However, there are still important problems left open. Hence, the second objective of this thesis is to sharpen the panorama of answering queries with negation in Horn DLs.

In the next section, we provide a wider perspective on the applications of DLs in data management, and thus on the motivation of the work carried out in this thesis. The following two sections on this chapter survey known results and particulars on the use of DLs in conceptual modeling (Section 1.3) and ontology-based data access (Section 1.5). We discuss related work to full satisfiability and finite model reasoning in Sections 1.4 and 1.6, respectively.

1.1 Description Logics

The syntax of DLs allows to represent knowledge in a structured way in terms of *concepts* and *roles*, which correspond to unary predicates and binary predicates, respectively. Using the various constructors provided by a particular *description language* one can then describe *complex concepts*. For example, suppose we want to model knowledge in the domain of *food and*

nutrition. Using the description language $\mathcal{ALC}$ which contains constructors that correspond to the Boolean connectives, and constructors for quantifying (universally and existentially) over domain elements connected through a certain role, one could express, starting from *atomic concepts* such as Food and Ingredient, the $\mathcal{ALC}$ concepts

$$\mathsf{Food} \sqcap \mathsf{HighProtein} \quad \text{and} \quad \mathsf{Food} \sqcap \mathsf{LowSugar} \ .$$

These concepts intuitively describe '*food with a high content of protein*' and '*food with low sugar content*' that may be relevant, e.g., for the dietary plan of a patient with type-2 diabetes. If in addition containsNutrient is an atomic role and Sodium is another atomic concept, we can form the concept descriptions

$$\mathsf{Food} \sqcap \exists \mathsf{containsNutrient}.\mathsf{Sodium} \quad \text{and} \quad \mathsf{Food} \sqcap \forall \mathsf{containsNutrient}.\neg \mathsf{Sodium} \ .$$

denoting, respectively, those *food items that contain sodium among their nutrients*, and those *food items that do not contain sodium* (all their nutrients are not Sodium).

Knowledge representation systems based on DLs provide facilities to store and manipulate knowledge by means of an ontology. A *DL ontology* has two components: the *terminological* knowledge or *TBox* and the *assertional* knowledge or *ABox*. The TBox introduces the vocabulary of an application domain, while the ABox describes a specific state of affairs of an application domain in terms of that vocabulary. Continuing with our example of the food domain, we can state, e.g., that all *meat ingredients* contain *minerals*:

$$\mathsf{Meat} \sqsubseteq \exists \mathsf{containsNutrient}.\mathsf{Minerals}$$

ABoxes, on the other hand, contain assertions about named individuals. For instance, if BEEF and IRON are individuals then

$$\mathsf{Meat}(\mathsf{BEEF}) \quad \text{and} \quad \mathsf{Mineral}(\mathsf{IRON})$$

are concept assertions, and

$$\mathsf{containsNutrient}(\mathsf{BEEF}, \mathsf{IRON})$$

is a role assertion. Together all these ABox assertions denote that *beef* is an instance of *meat* that contains a *mineral* named *iron* as a nutrient.

DLs support inference patterns that occur in many applications of intelligent information processing systems, and which are also used by humans to structure and understand the world such as classification of concepts and individuals. Classification of concepts determines subconcept/superconcept relationships between the concepts of a given terminology, and thus allows one to structure hierarchically the terminology. This hierarchy provides useful information on the connection between different concepts and it is used advantageously to speed-up other inference services. Classification of individuals determines whether a given individual is always an instance of a certain concept (i.e., whether this instance relationship is implied by the description of the individual and the definition of the concept). It thus provides useful information on the properties of an individual. In Section 2.1, we will cover the formal notions related to DLs and reasoning upon them.

1.2 DLs for Data Management

An important aspect for developing a database application is to represent the relevant features of the domain of interest about which the database should be knowledgeable. These features are primordially about the structure of the data. Typically, the specification of such features is done in a high-level language, which makes it accessible to both users and designers. The activity of specifying the structure of the data managed in a database application is known as *conceptual (data) modeling*. In databases, the best known language for conceptual data modeling is the *Entity-Relationship (ER)* data model [44].

From the *conceptual schema* built during the conceptual modeling phase, the database designer develops a *logical schema*. The role of the logical schema is to describe the structure of the data stored in the database, which includes the data types, interconnections, and constraints that must be satisfied by the data. Different data models are used for this purpose, but the *relational data model* is usually the logical model of choice. The data structured and stored according to the logical schema is then managed by large software systems called *Database Management Systems (DBMS)*. The *database* then stores facts about the (current) state of the world. Usually the so-called *closed world assumption (CWA)* is made, this means, that a fact is regarded as false unless it has been explicitly stated as true. Note that this is an assumption that works well with the restriction that the database represents only a very limited form of partial information. In order to provide access to the data stored in databases, DBMS support a variety of *query languages*, specifying in a declarative way the data to be retrieved. For relational databases, *SQL (Structured Query Language)* is the practical query language of choice. However, based on the observation that tables can be viewed as predicates, first-order logic formulas with free variables offer a more suitable representation from the theoretical viewpoint (although the expressiveness of SQL goes beyond first-order logic [3, 91]). Figure 1.2 depicts the architecture of a *data management system*.

With the emerging of more complex kinds of databases, such as *distributed databases*, that keep information at various sources connected by networks, the problem of accessing information stored in databases has become more difficult. Notably, the technologies for accessing distributed databases need to be such that the user perceives a single database. In a more general setting, *heterogeneous* and *federated databases* are collections of independent databases which choose to share information but are maintained autonomously. Integrating heterogeneous data sources is a fundamental problem in databases, which has been studied extensively for about three decades both from a formal and from a practical point of view [20, 43, 70, 92, 123, 125].

Most of the research around this problem in the last two decades has been motivated by the need to integrate data sources on the Web. The research around combining data residing at different sources has been carried on under the name of *data integration*. Data integration, as discussed, aims to provide the user with a unified view of multiple (possibly heterogeneous) data sources. In the declarative approach to the data integration problem, it is assumed that the data integration system is characterized by giving explicitly to the client a global, virtual, reconciled and unified view of the data. In order to access the data, the user formulates queries

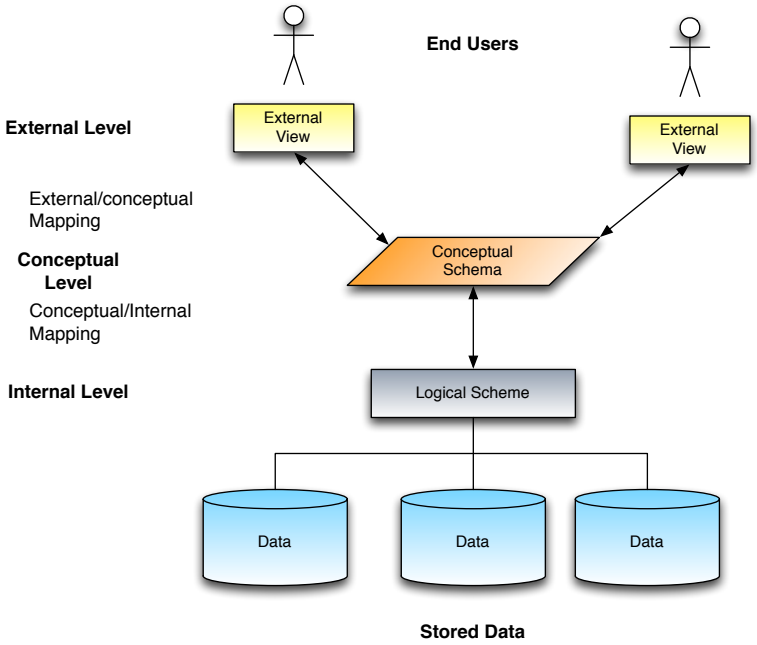


Figure 1.2: Architecture of a data management system

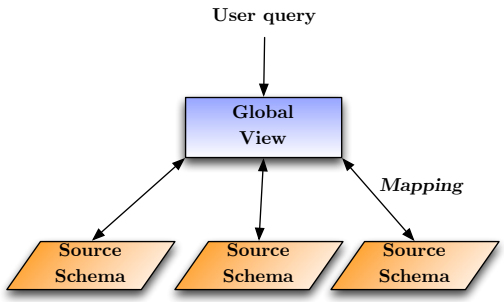


Figure 1.3: Abstract architecture of a data integration system

in terms of the *global view*, and the system then reformulates the user query in terms of the data sources using appropriate *schema mappings*. The abstract architecture to such an approach is depicted in Figure 1.3.

We will now discuss how DLs have been applied in the described setting of data management. Indeed, DLs are good candidates for describing the schema of databases since they are well suited to capture structural aspects of data. In particular, the description logic language $\mathcal{ALCQI}$ fully captures *class-based* representation formalisms such as ER schema and UML class diagrams [33]. For instance, consider the ER schema in Figure 1.4, specifying the structure of a food delivery system. The following $\mathcal{ALCQI}$ axioms exemplify how the constraints in that diagram can be

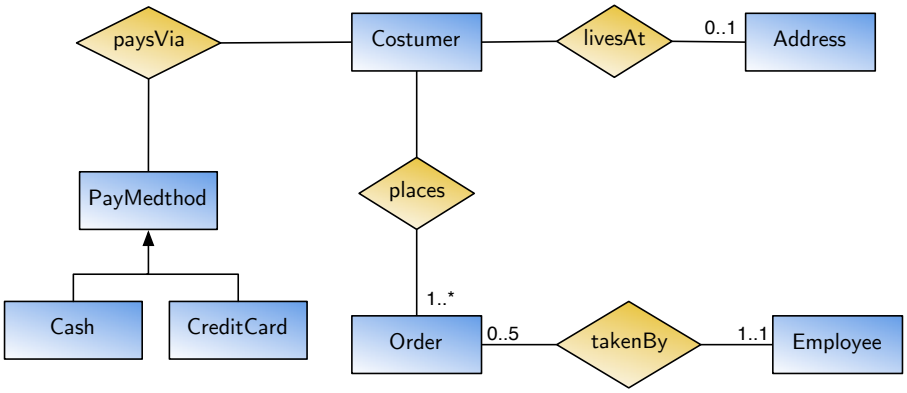


Figure 1.4: Example of an ER schema

captured:

$$\mathsf{Costumer} \sqsubseteq \forall \mathsf{livesAt}.\mathsf{Address} \qquad \mathsf{Costumer} \sqsubseteq (\leqslant 1\ \mathsf{livesAt})$$

$$\mathsf{Costumer} \sqsubseteq \exists \mathsf{places}.\mathsf{Order} \qquad \mathsf{Costumer} \sqsubseteq \forall \mathsf{paysVia}.\mathsf{PayMethod}$$

$$\mathsf{Cash} \sqsubseteq \mathsf{PayMethod} \qquad \mathsf{CreditCard} \sqsubseteq \mathsf{PayMethod}$$

$$\mathsf{Order} \sqsubseteq \forall \mathsf{takenBy}.\mathsf{Employee} \qquad \mathsf{Employee} \sqsubseteq (\leqslant 5\ \mathsf{takenBy}^{-})$$

Furthermore, in data integration ontologies are considered as the ideal formal tool to provide a shared conceptualization of the domain of interest. In the *ontology-based data access* (OBDA) paradigm, from a given set of data sources at the internal level of an information system, the aim is to build a service on top of that level presenting a conceptual view of the data to the end users of the information system. The role of the ontology is to express the conceptual level of the system and provide the unique access point for the interaction between clients and the system, while keeping the data sources independent from the ontology. That means, that the ontology describes the domain of interest at a high-level, abstracting away from how data sources are maintained in the data level of the system itself. Essentially, in the OBDA scenario the goal is to link to the ontology a collection of data that exists autonomously, and that has not been necessarily structured with the purpose of storing the ontology instances [109].

Besides providing a (global) conceptual view of the data, DL ontologies also endow the data in an information system with a semantic dimension. Indeed, with the advent of the Semantic Web, which aims to allow computers to intelligently search, combine and process Web content, representing the *semantics* of distributed information sources has become a necessity. To cover that necessity, the intended meaning (i.e., the semantics) of some Web sources can be explicitly specified in a format that is processable by computers in such a way that the data stored is provided with *formal semantics* that specifies in a precise way which conclusions should be drawn from the collected information. Formal modeling languages with automated deduction capabilities have turned out to be successful in the latter task, and have given rise to the paradigm of ontology. In that context, an ontology is understood as a formalism whose purpose

is to support human or machines to share common knowledge in a structured way. The *Web Ontology Language (OWL)* is intended for modeling complex schematic knowledge [62, 69, 130]; as the name suggests, OWL is most relevant for ontological modeling. Notably, the formal underpinning of significant part of the OWL standard is provided by description logics [69]. Certain sublanguages, or so-called profiles, of OWL are of particular interest for this thesis: OWL EL, OWL RL and OWL QL. The particularity of these profiles is that the description logics underlying them are tractable. Specifically, the *standard* reasoning tasks (subsumption and satisfiability) can be carried out in polynomial time.

1.3 Conceptual Data Modeling

In this section, we offer a general view of some results on applications of DLs in conceptual data modeling.

One interesting aspect of conceptual modeling is the identification of a number of mechanisms that support the development of large models by initially abstracting details, and then introducing them in a step-wise and systematic manner. These mechanisms include: thinking of objects as whole instead of just a collection of their attributes or components; abstracting away the detailed differences between individuals, so that a class can represent the commonalities; and abstracting the commonalities of several classes into a superclass. The benefits of abstraction in conceptual modeling is that it results in a structured information model, which is easy to build and maintain.

However, a major challenge of conceptual design is the generation of a schema that is consistent and does not contain redundancies. Traditional tools developed to aid conceptual schema design are mainly graphical, which made them appropriate for visualization purposes. But the decision on how to organize concepts in a taxonomy, and specially the systematic verification of the correctness of the schema is guided only by the designer's experience. Application domains are complex, and the construction of a large application schema must often be partitioned among various designers, each part designed under the particular designer's view of the domain. The resulting schemata have to be then integrated into a larger one without losing consistency of the information. Hence the need for automatic design tools that more effectively support the construction of large, complex conceptual schemata, and the verification of their correctness.

A great deal of research has been dedicated to investigate the integration of deductive capabilities of KR formalisms into conceptual data modeling. In [22] taxonomic reasoning techniques of *frame description languages* are proposed for supporting knowledge acquisition and conceptual schema design. The approach followed in this work is to extend conceptual models with defined concepts and giving them a logic-based semantics. This made it possible to infer so-called isa relationships between concepts on the basis of their descriptions. Further, Piza et al. [108] propose a generalization of both semantic networks and frame systems with type constructors, which are used in semantic data models. In particular, the similarity of abstraction mechanisms of semantic models and the proposed KR formalism is exploited to develop an integrated

conceptual modeling framework. The resulting framework is restricted to basic features, e.g., multi-valued roles and cardinality constraints are left out, as well as role hierarchies. Later on, Artale et al. [7] related the *object data model* to $\mathcal{ALC}$. The authors show how object database descriptions can be expressed using concept descriptions. One significant aspect of this work is the identification of reasoning problems in which one can take advantage of techniques for automated reasoning in description logics. In a more general view of the problem, Calvanese et al. [33] propose description logics as a unifying framework for class-based formalisms. That work establishes a relationship between semantic and object-oriented data models by rephrasing them in terms of description logics. More specifically, they show that the description logic $\mathcal{ALCQI}$ is expressive enough to formalize the semantics of ER and object oriented data models. In fact, they provide a translation from ER schema (and object-oriented data models) to an $\mathcal{ALCQI}$ TBox that is "information preserving". Notably, that formalization provided the basis for reasoning in conceptual models. We will provide details of the formalization approach in Chapter 2.

The seminal work in [21, 34, 35] was refined by Artale et al. [8], who investigate the computational complexity of reasoning over various fragments of the *Extended Entity-Relationship (EER)* language, which includes isa between entities and relationships, disjointness and covering of entities and relationships, cardinality constraints for entities in relationships and their refinements as well as multiplicity constraints for attributes. In particular, this work extends the known ExpTime-completeness result for UML class diagrams in [21]. Artale et al. [9] showed that reasoning over EER diagrams with isa relations between relationships is ExpTime-complete even without relationship covering. They also establish that reasoning becomes NP-complete when isa between relationships is dropped (while still allowing all types of constraints on entities); and that by further omitting disjointness and covering over entities reasoning becomes polynomial. The complexity upper bounds are established by a formalization of the EER diagrams using variants of the description logic *DL-Lite*. These correspondences also show the usefulness of *DL-Lite* as a language for reasoning over conceptual models and ontologies. We will revisit these results in Chapter 7.

1.4 Full Satisfiability of Conceptual Models

To simplify the presentation in the following, we will refer to either an UML class diagram or an ER schema simply as *class diagram (CD)*. Informally, a *legal instance* of a CD $\mathcal{C}$ is an instantiation of the classes (entities) and associations (relations) in $\mathcal{C}$ that precisely satisfy all constraints in $\mathcal{C}$. Hence, $\mathcal{C}$ is *satisfiable* if it has a legal instance. However, a more meaningful satisfiability property involves 'intended' instances. For example, to require that for each class C in $\mathcal{C}$, there is a legal instance of $\mathcal{C}$ such that the extension of C is non-empty and finite. This stronger notion of satisfiability was first introduced by Lenzerini and Nobili [90], and termed *strong satisfiability*. Indeed, under the assumption that class diagrams are intended to model real world domains, which are intrinsically finite, this notion seems appropriate. Lenzerini and Nobili, have also pointed out that the interaction of cardinality restrictions and cycles in the

diagram might cause that the only legal instance of the diagram is the empty instance. They also show that, for a simple class of diagrams, a strongly satisfiable concept diagram has an instance in which *all* class extensions are non-empty. In particular, these 'restricted' class of diagrams allow only relations with cardinality constraints among classes, and do not include isa relations.

Similar notions of satisfiability have been studied for more complex CDs. Kaneiwa and Satoh [78] investigated the problem of *full satisfiability* on restricted CDs that include classes with typed attributes and cardinality constraints on the attributes, unconstrained associations and constrained generalization sets. They describe three *triggers* for inconsistency in such diagrams and provide algorithms for deciding full satisfiability of the restricted CDs.

Verifying full satisfiability of CDs allows to detect unsatisfiable classes, which means either that the diagram contains unnecessary information that should be removed, or that there is some modeling error (e.g., over-constraining in the diagram) that leads to the loss of satisfiability. In some scenarios, like in *configuration management* [26], verifying full satisfiability is an useful reasoning task. Configuration management touches aspects of CDs that are not in the focus of traditional model engineering. Most notably, it requires numerical reasoning as well as the ability to handle instances independently from classes. Hence, it seems relevant to study a reasoning task such as full satisfiability on the side of DLs, to provide a formal approach for supporting verification of CDs. We will address full satisfiability on DLs that capture interesting modeling features for the scenario of configuration management and discuss how our results can be applied in that area in Chapter 7.

1.5 Ontology-Based Data Access

DL ontologies, as discussed in Section 1.2, can serve as high-level conceptualizations for (possible distributed and heterogenous) data sources with the advantage that a semantic account of the information is also provided. That setting allows users to access data without the need to know how the data is actually organized and where it is stored. The constraints imposed by the ontology aid to detect inconsistencies that might be present in the data. On the other hand, the background knowledge provided by the ontology can also be used to enrich the data schema with additional symbols to be used in a query. Figure 1.5 describes the abstract architecture of an *ontology-based data access* system.

From the scenario described above, one can draw the conclusion that accessing data through ontologies amounts to querying possibly incomplete and constrained databases. We will refer to the latter task as *ontological query answering*. Notably, query processing in OBDA usually involves also reformulating the query by using *mappings* that relate the data schema and the ontology. However, this will not be explicitly considered in this thesis as it is out of scope.

From the semantic view point, constraints imposed by the ontology are taken into account in ontological query answering using the notion of *certain answers* [1], which amounts to logical entailment. Intuitively, the incompleteness of the data is handled by considering *all possible*

extensions of the data satisfying the axioms in the ontology (i.e., all models of the ontology). As a result of the certain answer semantics, the choice of the query language needs to be more careful than for traditional databases. In particular, one cannot consider first-order logic queries since this leads to undecidability. However, in settings dealing with incomplete databases [98], a good trade-off regarding the query language choice has been found. In those settings, *conjunctive queries* (and unions thereof), which correspond to the *select-project-join* fragment of SQL, are the language of choice. Notably, these kind of queries are also the best supported by commercial database management systems.

A typical assumption in OBDA scenarios [37, 48, 56, 89, 109] is that the amount of data is large; think, for example, about large data sources on the Web. Hence, one of the challenges in OBDA is to provide efficient ontological query answering, where efficient in this case means that query answering has to be at least tractable (i.e., polynomial) w.r.t. to the size of the data [39, 71, 103]. The impact of the choice of the ontology language on the OBDA setting has been largely investigated in the literature [39–41, 53, 55, 81, 104]. The results of such investigations have shown that expressive ontology languages are not really suited for ontological query answering. In particular, in expressive fragments of OWL ontological query answering is co-NP complete in data complexity [55, 93]. Hence, to overcome costly query answering one needs to restrict the ontology language. Among the ontology languages guaranteeing at least tractable complexity we have the languages Horn-$\mathcal{SHIQ}$ [53, 71, 104], $\mathcal{EL}^{++}$ [13, 88] and *DL-Lite* [9].

From those restricted languages, the *DL-Lite* family of DLs was developed specifically for capturing meaningful modeling features of ontology and conceptual modeling formalisms and on the same time keeping the complexity of reasoning low [38, 40].

In particular, answering (unions of) conjunctive queries in *DL-Lite* is in AC^0 which allows to perform conjunctive query answering over *DL-Lite* ontologies by first reformulating the query using the knowledge of the terminological part of the ontology (and thus independently of the data) and then evaluating such a reformulation of the query over the data. Since the data is assumed to be managed by a RDBMS, the evaluation of the reformulated query can thus be carried out by an SQL-engine. This means that query answering in *DL-Lite* is as costly as query answering in traditional databases (in case the size of the query is ignored), and that we can thus take advantage of all the optimizations present in the SQL-engines. However, it is interesting to note that the size of the proposed reformulations [40] may become very large (exponential in the size of the original query). A number of alternative reformulating query mechanisms have been proposed to limit the size of the reformulation; see e.g., [81] and reference therein. Furthermore, for DLs for which query answering is polynomial, e.g., $\mathcal{EL}^{++}$, such a rewriting approach is not possible anymore; although the so-called *combined approach* [85] allows to delegate query answering to RDBMs by reformulating both the query and the data in terms of the constraints of the terminological part of the ontology.

Following the tradition of classical databases, most of the research on ontological query answering has concentrated on the study of *positive* queries, such as CQs and UCQs. Unfortu-

nately, some natural queries require to express either of these *difference* or *complementation*; for example, to retrieve *'all members of the staff that do* `not` *belong to a union'* or *'all students whose month of birth is* `(different from) not` *September'*. Indeed, a well-known fact from database theory is that answering CQs with negation is harder than answering CQs; this is the case, for example, for *open world* query-answering with integrity constraints [115], query-answering in the context of data exchange [54] or query-answering using materialized views [2]. In the case of ontological query answering, it has been shown that answering conjunctive queries extended with *inequalities* and *safe negation* over expressive ontologies is undecidable [41, 116]. On the other hand, the investigation of ontological query answering with these two forms of negation over lightweight ontologies has received only limited attention. A remarkable exception is the work by Rosati [116] in which it is shown that answering *unions* of conjunctive queries with inequalities and safe negation over ontologies formulated in (some variants) of *DL-Lite* is undecidable. Given the fact that the undecidability results were established for the case of *unions of* CQs with inequality or safe negation, the question of whether this is also the case for a single (instead of a union) negated CQ naturally arises. Moreover, even if this is the case, one would be interested in knowing whether these extensions of CQs with negation can be restricted in a way that decidability is recovered. Indeed, this is one of the research lines followed in this thesis. We will discuss in more detail this topic and answer some of these questions in Chapter 6.

Observe that some of the data schemata *integrated* by the ontology in the OBDA setting may contain constraints such as functionality or more generally, cardinality constraints. Hence, in order to faithfully capture these constraints, one should use DLs that lack the finite model property. This means that in such cases ontological query answering amounts to logical entailment w.r.t. finite models. Again, this assumption stems from the fact that the data sources considered are typically finite databases. We present a more detailed discussion on query answering under the finite model assumption in Section 1.7 below, and present some contributions on ontological query answering under this assumption in Chapter 5.

1.6 Finite Model Reasoning on DLs

In this section, we will discuss some of the techniques used in passed research on finite model reasoning in description logics. As we discussed at the beginning of this section, finite model reasoning differs from reasoning over arbitrary models in logics that lack the *finite model property (FMP)*. For description logics, this is the case e.g., for logics that allow the inverse constructor for roles and some form of functionality constraints. These constructors present, e.g., in $\mathcal{ALCQI}$, in combination with cycles in the ontology may interact in such a way that an ontology admits no finite model, although it admits one with infinite domain. That is, $\mathcal{ALCQI}$ lacks the FMP, and thus, reasoning under the finite and under the unrestricted models constitute two distinct problems.

Traditional techniques developed for reasoning w.r.t. arbitrary models on expressive description logics ($\mathcal{ALCQI}$ included) are based on the correspondence between DLs and propositional

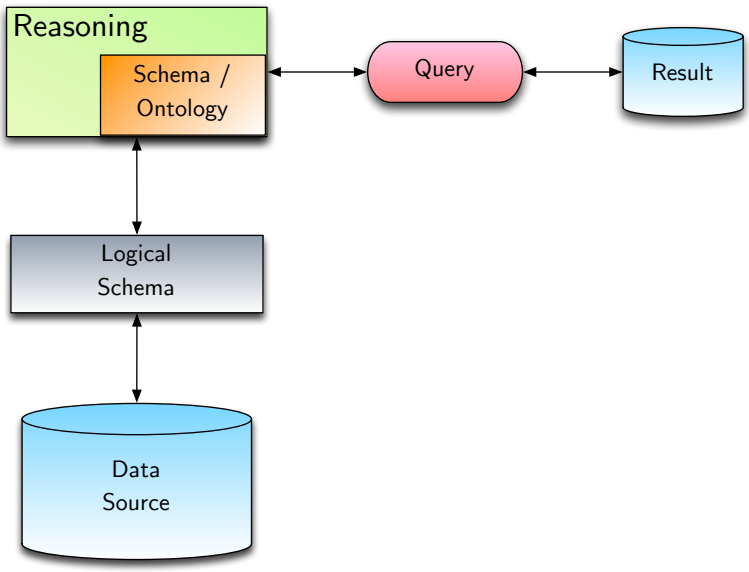


Figure 1.5: OBDA system architecture

dynamic logics (PDLs). PDLs are formalisms specifically designed for reasoning about program schemes [87], and their correspondence with DLs was first described in [121] and extended to more expressive logics in [46]. This correspondence allows to reduce concept consistency to satisfiability of a formula of some PDL. The methods for checking satisfiability of PDL formulas exploit the fundamental *tree model property*, which states that every satisfiable formula admits a particular model that is tree-shaped and of bounded degree. If the domain is required to be finite, however, the existence of a tree-like model is not guaranteed, and the known reasoning methods are not applicable.

To cope with the problem of finite model reasoning on $\mathcal{ALCQI}$, Calvanese [31] developed a method based on the idea of constructing a system of linear equations from an $\mathcal{ALCQI}$ ontology, and relating the existence of particular solutions of the system to the existence of a finite model of the ontology. This method reduces the problem of concept finite satisfiability to that of finding a solution over the integers for a system of equations. Further, the method relies on an expansion of the original $\mathcal{ALCQI}$ ontology that causes a double exponential blow up. Since the size of the linear equations depends on the size of the ontology, and the linear equations produced can be solved in polynomial time, applying linear programing techniques, Calvanese's method provided a 2ExpTime upper bound for the complexity of finite concept satisfiability in $\mathcal{ALCQI}$.

The upper bound provided by Calvanese was improved by the results on finite satisfiability of the *two variable fragment of first order logic with counting quantifiers (C2)* from which $\mathcal{ALCQI}$ is a fragment. In particular, finite satisfiability of C2 is NExpTime-complete [110]. Further, Lutz et al. [94] showed that finite satisfiability of $\mathcal{ALCQI}$ is ExpTime-complete, i.e., it has the same complexity of reasoning w.r.t. arbitrary models. Similar to Calvanese's approach, the idea behind their algorithm is to translate a given satisfiability problem into a system of inequations that

can be solved using linear programing methods. In this translation, each variable represents the number of elements described by so-called *mosaics*, which describe the (unary) type of a domain element together with its neighborhood, i.e., the numbers and types of (relevant) role successors. Using a rather strict notion of mosaics and an appropriate data structure to represent mosaics allows to keep the number of mosaics exponential in the size of the input. This yields an exponential bound on the number of variables and also on the size of systems of inequations, which yields the ExpTime upper bound. Moreover, their results extend also to the problem of reasoning in the presence of assertional knowledge, which shows that finite ABox consistency in $\mathcal{ALCQI}$ is also ExpTime-complete. The result follows from a reduction to finite concept satisfiability, which in particular also shows that the complexity bound is independent of the way in which the numbers occurring in the TBox are coded. Notably, the algorithm is *best-case* ExpTime since it constructs an exponentially large system of inequations, and thus, it is not expected to have an acceptable runtime behavior if implemented in a naive way. Efficient implementations of the algorithm are not known and are not apparent from the approach.

Interestingly, for less expressive description logics, finite model reasoning has resulted to be more feasible. *DL-Lite*$_\mathcal{F}$ is a comparably inexpressive DL tailored specifically for database applications, but that also lacks the FMP because it includes both inverse roles and functionality restrictions. Building on a technique that was developed in database theory by Cosmadakis, Kanellakis and Vardi to decide the implication of inclusion dependencies and functional dependencies in the finite [45], Rosati has shown that finite model reasoning in *DL-Lite*$_\mathcal{F}$ can be reduced (in polynomial time) to unrestricted reasoning in *DL-Lite*$_\mathcal{F}$ [117]. In fact, the reduction relies on completing the TBox by finding certain cyclic inclusions and 'reversing' them. For example, the cycle

$$\exists r^- \sqsubseteq \exists s \qquad \exists s^- \sqsubseteq \exists r \qquad (\mathsf{funct}\ r^-) \qquad (\mathsf{funct}\ s^-)$$

that consists of existential restrictions in the 'forward direction' and functionality statements in the 'backward direction' would lead to the addition of the reversed cycle

$$\exists s \sqsubseteq \exists r^- \qquad \exists r \sqsubseteq \exists s^- \qquad (\mathsf{funct}\ r) \qquad (\mathsf{funct}\ s).$$

We will explain in detail the cycle reversion technique in Chapter 3. A consequence of using the reduction illustrated above is that finite model reasoning in *DL-Lite*$_\mathcal{F}$ does not require any new algorithmic techniques and can be implemented as efficiently as unrestricted reasoning. Given that *DL-Lite*$_\mathcal{F}$ is a very small fragment of $\mathcal{ALCQI}$ these observations raise the question whether the cycle reversion technique extends also to larger fragments of these DLs. Specifically, to the so-called *Horn DLs* which are well-known to be algorithmically better behaved than non-Horn DLs such as $\mathcal{ALCQI}$ [71]. This is actually one of the main topics investigated in this thesis. We will discuss in more detail the cycle reversion technique and its extensions to Horn DLs in Chapter 3.

1.7 Query Answering under the Finite Model Assumption

Other results involving reasoning w.r.t. finite domains in DLs and related formalisms, such as those described in [17, 118], concern *finite controllability* [118] of answering (Boolean) conjunctive queries (CQs) in the presence of a set of constraints. More precisely, for a given class of constraints C, answering a query q w.r.t. a set of constraints φ is *finitely controllable* if deciding whether all finite models of φ satisfy q is equivalent to the problem of deciding whether all (unrestricted) models of φ satisfy q, in symbols $\varphi \models q$ iff $\varphi \models_{\mathsf{fin}} q$. Notably, Rosati showed that answering CQs w.r.t. inclusion dependencies is indeed finitely controllable, closing then a long standing open problem left in [77]. To solve the problem, Rosati developed a finite model generation procedure called *finite chase*; and then showed that for every database D and set Σ of IDs, and for every N there exists a finite structure $\mathcal{I}$ extending D and satisfying Σ such that for every (Boolean) CQ q comprised of at most N atoms $\mathcal{I} \models q$ iff $\Sigma, \mathcal{I} \models q$. Notably, description logics such as *DL-Lite*$_{core}$ and *DL-Lite*$_{core}^{\mathcal{H}}$ are essentially based on IDs, and are thus finitely controllable.

Bárány et al. [17] studied finite controllability in the scenario where the constraints are expressed using the *guarded fragment of first-order logic (GF)* [5] (which subsumes expressive DLs such as $\mathcal{ALC}$ and $\mathcal{ALCI}$– but not $\mathcal{ALCQI}$ [59]). Since CQs may not be guarded, the FMP of the GF is not sufficient to establish finite controllability in that case. Rather, the problem amounts to establish the FMP of an extension of the GF with universally quantified Boolean combinations of negative atoms. Indeed, this is the case because evaluating a Boolean conjunctive query q against a guarded first-order theory φ is equivalent to checking whether $\varphi \wedge \neg q$ is unsatisfiable. Using the fundamental idea of the finite chase introduced by Rosati, the authors show that the GF is finitely controllable. The main technical result presented in [17] is that one can construct a family of finite models of a given satisfiable extended GF theory $\varphi \wedge \neg q$. The role of these so-called *Rosati covers* in proving finite controllability can be informally explained as follows. Given a GF theory φ and a query q Bárány et al. show that there is an acyclic query $\widehat{q}$, such that $\varphi \models q$ iff $\varphi \models \widehat{q}$. Now, since $\widehat{q}$ is acyclic, it can be reformulated as a GF-sentence. Then, the authors prove that the same reduction is also valid over finite models, i.e., that $\varphi \models_{\mathsf{fin}} q$ iff $\varphi \models_{\mathsf{fin}} \widehat{q}$. In particular, that given a finite model $\mathfrak{A}$ of $\varphi \wedge \neg\widehat{q}$, a finite model of $\varphi \wedge \neg q$ can also be found. Since q is not acyclic, $\mathfrak{A}$ is not in general the desired model, i.e., it may be a model of q. However, the existence of Rosati covers $\mathfrak{A}$ provides finite models of $\varphi \wedge \neg\widehat{q}$ retaining "sufficient degree of acyclicity" so as not to render a model of q.

Constructions of finite models with certain degree of acyclicity have been also studied by Otto in [105]. The construction by Otto is provided in the context of finite transition systems and Kripke structures. In general terms, the main result provided in [105] establishes that for a finite transition system $\mathfrak{A}$ (which can be regarded as a relational structure), and a given $N \geq 3$, it is always possible to construct a bisimilar transition system $\widehat{\mathfrak{A}}$ that does not contain cycles of length less or equal to N. This construction is obtained from a product of $\mathfrak{A}$ with a cyclic group G of girth greater than N. We will come back to this construction in Chapter 5, where we investigate ontological query answering under the finite model assumption.

1.8 Results and Structure of the Thesis

The discussion in the previous sections puts in evidence that reasoning in DLs can and have been used advantageously in data management applications. In that context, the need to provide support for modeling features, such as cardinality constraints and inverse of relations, requires the use of (expressive) DLs that lack the finite model property. In such cases, whenever reasoning is used for data management applications one needs to consider finite model reasoning in DLs instead of reasoning w.r.t. unrestricted (i.e., possibly infinite) models. The available results on the subject have shown that this is a difficult task, and that in some cases requires algorithmic techniques that are not transparent from the modeling perspective. Bearing this motivation in mind, we investigate finite model reasoning in the so-called Horn DLs. These logics are of interest because they provide sufficient expressive power as to capture some interesting features in conceptual modeling (although they are not able to express covering constraints) and also because, due to their model theoretical properties, they are usually well behaved w.r.t. complexity of reasoning. We also investigate ontological query answering over Horn ontologies considering queries with negation. We will particularly focus on queries allowing for so-called safe negation, guarded negation and inequalities. We can summarize the results and contributions in this thesis as follows:

- we determine the complexity of finite model reasoning in Horn fragments of expressive DLs by reducing finite model reasoning to unrestricted reasoning (Chapter 3);
- given that such reduction may result in an inefficient procedure for finite model reasoning, we provide a consequence-driven algorithm for finite model reasoning in Horn DLs (Chapter 4);
- we then extend these results for ontological query answering under the finite model assumption (Chapter 5);
- in Chapter 6, we determine the data complexity of answering queries with negations over Horn ontologies (see Table below);

<table>
<tr><th></th><th>DL-Lite$_{core}$</th><th>DL-Lite$_{core}^{\mathcal{H}}$</th><th>$\mathcal{ELI}_{\bot}$</th></tr>
<tr><td>UCQ$^{\neg s}$</td><td colspan="3">undec. Cor. 6.4</td></tr>
<tr><td>CQ$^{\neg s}$</td><td>coNP-hard</td><td>undec.
Thm. 6.6</td><td>undec.
Thm. 6.2</td></tr>
<tr><td>1 guarded neg</td><td colspan="3">PTime-hard Lemma 6.8</td></tr>
<tr><td>$\geq$ 2 guarded negs</td><td colspan="3">coNP-hard Lemma 6.9</td></tr>
<tr><td>$\geq$ 2-CQ$^{\neq}$</td><td>coNP-hard
Thm. 6.14</td><td colspan="2" rowspan="2">undec. Thm. 6.12</td></tr>
<tr><td>1-CQ$^{\neq}$</td><td>PTime-hard
Thm. 6.13</td></tr>
</table>

- finally, we put together the obtained results and argue on their use in database applications (Chapter 7).

All the results presented in this thesis as original contributions have been published in the following venues:

1. Ibáñez-García, Lutz, and Schneider. Finite Model Reasoning in Horn-$\mathcal{ALCQI}$. In Proc. of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR 2014).

2. Ibáñez-García, Lutz, and Schneider. Finite Model Reasoning in Horn-$\mathcal{SHIQ}$. In Proc. of the 26th Int. Workshop on Description Logics (DL 2013).

3. Gutiérrez-Basulto, Ibáñez-García, Kontchakov, and Kostylev. Conjunctive Queries with Negation over *DL-Lite*: A Closer Look. In Proc. of the 7th International Conference on Web Reasoning and Rule Systems (RR 2013). *Best Student Paper Award.*

4. Gutiérrez-Basulto, Ibáñez-García, and Kontchakov. An Update on Query Answering with Restricted Forms of Negation. In Proc. of the 6th International Conference on Web Reasoning and Rule Systems (RR 2012).

5. Ibáñez-García. Finite Model Reasoning in *DL-Lite* with Cardinality Constraints. In Proc. of the 25th Int. Workshop on Description Logics (DL 2012).

6. Artale, Calvanese, and Ibanez-Garcia. Full satisfiability of UML class diagrams. In Proc. of the 29th Int. Conf. on Conceptual Modeling (ER 2010).

7. Artale, Calvanese, and Ibanez-Garcia. Checking full satisfiability of conceptual models. In Proc. of the 23nd Int. Workshop on Description Logics (DL 2010).

Chapter

2 Preliminaries

In this Chapter, we will formally introduce description logics and conceptual modeling languages. We do this in a comparative manner. In more detail, we start with an introduction to the field of description logics, and we then discuss the relation of description logics and conceptual modeling languages. We introduce two of the main languages for conceptual modeling, namely, ER schema and UML class diagrams, and show how the semantics of these modeling languages can be captured by expressive description logics. Moreover, we provide an overview of the main reasoning tasks defined for description logics systems, and discuss how this reasoning capabilities have been used advantageously in conceptual modeling and database access.

2.1 Description Logics

Description logics (DLs) evolved from early knowledge representation formalisms such as *semantic networks* [113] and *frames* [99], in which knowledge was represented by means of directed graphs or structured objects. Unfortunately, the lack of formal semantics in these early formalisms made them application-dependent, that is, the meaning of, e.g., a directed graph depended on the implementation of a particular reasoning system. The use of formal logic for representing and reasoning about knowledge constituted a crucial step towards providing semantic means to knowledge representation formalisms. In fact, many aspects of semantic networks and frames admit a translation into *first-order logic (FOL)* [66]. Building on that correspondence, subsequent research in knowledge representation and reasoning focused on overcoming the problems inherent to the early knowledge representation formalisms.

Research in DLs [101] was initially developed under the name of *terminological systems* with the aim of representing the basic terminology of an application domain. Subsequently, when the interest focused on the set of concept constructors admitted in a language, they were known as *concept languages*. Finally, the term *description logics* became popular when the attention moved towards the properties of the underlying logical systems.

Formally, DLs are fragments of first-order logic providing decidable reasoning services. The syntax of DLs is designed for representing the important notions of the domain of interest in a structured way. On the other hand, since they are provided with formal semantics, it is possible to define *reasoning services* that allow to infer implicit knowledge from the explicit representation.

In DLs, the main building blocks are basic entities called *concepts* and *roles*, from which *(complex) concept descriptions* are built using the constructors provided by a particular *description logic language*. We will introduce in the remaining of this section the main description logic languages that have been used for conceptual modeling and ontology-based data access. We divide the presentation into two classes of DLs according to their expressive power:

Expressive DLs. These languages are propositionally closed, i.e., provide all the Boolean constructors for building concept descriptions. In addition to Boolean constructors, expressive DLs provide some other interesting modeling features such as *qualified numbers restrictions* and *inverse roles*, which are useful in conceptual modeling applications. Although reasoning over expressive DLs remains decidable, the complexity is usually high.

Horn DLs These are sub-Boolean languages, and thus rather unexpressive. The virtue of Horn DLs is that they provide a good trade-off between expressive power and complexity of reasoning. Indeed, Horn DLs are not able to express disjunctive statements, which is a source of complexity of reasoning. Furthermore, these logics have low *data* complexity, which makes them an ideal choice for reasoning about large amounts of data.

In addition to a description language, description logics provide formalisms for representing *intentional* and *assertional* knowledge. The intensional knowledge regards the terminological aspects of the domain of discourse, and it is asserted in the *TBox*. Typically, TBoxes contain assertions specifying concept definitions, concept properties, as well as relations among concepts. The assertional knowledge, on the other hand, describes the state of affairs in a particular 'world', and it is stored in the *ABox* in the form of assertions about *(named) individuals* such as membership to a concept or participation to a relation from the terminology. A *DL knowledge base (KB)*, also known as *ontology*, is a pair $(\mathcal{T}, \mathcal{A})$ where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox in some description language. In Sections 2.1.2 and 2.1.3 we present a general view of TBox and ABox formalisms, respectively. Further, we will discuss variants of TBox formalisms in Section 2.1.4. In particular, we will introduce Horn formalisms, which are know to influence the complexity of reasoning. Indeed, the study of the trade-off between the expressiveness of a DL language and the complexity of performing reasoning in the corresponding logic has constituted a major area of research in DLs [9, 12, 13, 15, 72]. In Section 2.1.6, we present the main reasoning tasks defined in the literature, as well as the known procedures and complexity results for performing such tasks.

2.1.1 Description Logic Languages

We next fix the vocabulary of all description logic languages used in this thesis. The *vocabulary* is based on three disjoint countably infinite sets $\mathsf{N_C}$, $\mathsf{N_R}, \mathsf{N_I}$ of *concept names*, *roles names* and *individual names*, respectively. Throughout this thesis we will use the letters (possibly with subindexes) A, B to denote concept names; p, q to denote role names; and $a, b, c, \ldots$ to denote individual names, respectively.

Concept and role names, also known as *atomic* concepts and *atomic* roles, are the syntactic building blocks for describing *complex* concept descriptions using the constructors available in a given *description language*. A specific description logic language is mainly characterized by the constructors it provides to build such complex concepts and roles. For example, the basic $\mathcal{ALC}$ language introduced by Schmidt-Schauß and Smolka [122], contains all the Boolean constructors: *conjunction* ($\sqcap$), *disjunction* ($\sqcup$) and *negation* ($\neg$), as well as *existential* ($\exists$) and *universal* ($\forall$) restrictions. In $\mathcal{ALC}$, for example, one can describe the term *'food products that are made from an ingredient that grows in a developing country and that contain exclusively carbohydrates or fats'* using the following concept:

$$\mathsf{FoodProduct} \sqcap (\exists \mathsf{madeFrom}.\exists \mathsf{growsIn}.\mathsf{DevelopingCountry}) \sqcap \forall \mathsf{contains}.(\mathsf{Carbs} \sqcup \mathsf{Fats}).$$

Syntax and Semantics of the DL $\mathcal{ALCQI}$

The DL $\mathcal{ALCQI}$ [32] is an extension of the basic $\mathcal{ALC}$. In particular, the $\mathcal{ALCQI}$ description language features a rich combination of constructors that include, besides all the Boolean connectives, *qualified number restrictions* and a role constructor for *inverses*.

Definition 2.1 ($\mathcal{ALCQI}$ syntax)**.** An $\mathcal{ALCQI}$ *role* is either a role name $p \in \mathsf{N_R}$ or the inverse p^- of a role name $p \in \mathsf{N_R}$. The set of $\mathcal{ALCQI}$ *concepts* is defined by the following syntax rules:

$$C_1, C_2 \ ::= \ A \ \mid \ \neg C_1 \ \mid \ C_1 \sqcap C_2 \ \mid \ C_1 \sqcup C_2 \ \mid \ (\geqslant n \ r \ C_1) \ \mid \ (\leqslant n \ r \ C_1)$$

where A ranges over the set of concept names $\mathsf{N_C}$, r denotes a role, C_1 and C_2 are $\mathcal{ALCQI}$ concepts and $n \geq 0$ is an integer. $\triangle$

To improve readability we allow to easily switch between a role name and its inverse by identifying $(r^-)^-$ and r. We refer to concepts of the form $(\leqslant n \ r \ C)$ as *at-most* restrictions, and to concepts of the form $(\geqslant n \ r \ C)$ as *at-least* restrictions. The presence of all Boolean connectives in $\mathcal{ALCQI}$ allows to express as usual disjunctions in terms of conjunction and negation:

$$C \sqcup D := \neg(\neg C \sqcap \neg D).$$

Further, using the constructors in the definition above, the following abbreviations can also be defined:

$$\begin{aligned} \top &:= A \sqcap \neg A; \\ \bot &:= A \sqcup \neg A; \\ \exists r.C &:= (\geqslant 1\ r\ C);\ \text{and} \\ \forall r.C &:= (\leqslant 0\ r\ \neg C). \end{aligned}$$

Recall that qualified (universal or existential) restrictions allow to represent the relationships existing between the objects of two classes. For example, for describing *'produce that are grown exclusively in european countries'*

$$\forall \mathsf{growsIn}.\mathsf{EuropeanCountry},$$

or *'places where some cereal is grown'*

$$\exists \mathsf{growsIn}^{-}.\mathsf{Cereal}.$$

Qualified number restrictions, on the other hand, can be used to describe objects connected to a number of instances of certain concepts through a particular role. For example, the concept

$$(\geqslant 3\ \mathsf{growsIn}\ \mathsf{Region})$$

describes the *'produce that grows in at least three regions'*, while

$$(\leqslant 2\ \mathsf{growsIn}\ \mathsf{Region})$$

conversely characterizes *'produce that grows in at most two regions'*. Further, *functionality constraints* can be expressed by a concept of the form $(\leqslant 1\ r\ C)$. For example, we can describe the class of *'endemic crops'* (i.e., crops that grow in a unique region) by the concept

$$\mathsf{Crop} \sqcap (\leqslant 1\ \mathsf{growsIn}\ \mathsf{Region}).$$

The *inverse role* constructor allows to denote the inverse of a given relation. One can for example describe *'the european countries that grow some cereal'* by the concept description:

$$\mathsf{EuropeanCountry} \sqcap \exists \mathsf{growsIn}^{-}.\mathsf{Cereal}$$

It is worth noticing that in a language without inverse of roles, in order to express the latter one may think of using two distinct roles (e.g., growsIn and grows), but unfortunately they cannot be put in the proper (semantic) relation.

Recall that one of the key features of DLs is that they are provided with formal semantics. More precisely, the meaning of concept descriptions is given by Tarsky-style semantics: concepts are interpreted as unary predicates and roles as binary predicates over a set called the *domain*.

Definition 2.2 ($\mathcal{ALCQI}$ semantics)**.** An *interpretation* $\mathcal{I}$ is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a nonempty set called the *domain* of $\mathcal{I}$ and $\cdot^{\mathcal{I}}$ is an *interpretation function* that maps

- every concept name A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and
- each role name p to a binary relation $p^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$.

The interpretation function is extended to complex $\mathcal{ALCQI}$ concept and role descriptions as follows:

$$\begin{aligned}
(p^-)^{\mathcal{I}} &:= \{(d,e) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (e,d) \in p^{\mathcal{I}}\} \\
(\neg C)^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C_1 \sqcap C_2)^{\mathcal{I}} &:= C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(C_1 \sqcup C_2)^{\mathcal{I}} &:= C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}} \\
(\geqslant n\ r\ C)^{\mathcal{I}} &:= \{d \in \Delta \mid \#\{e \mid (d,e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\} \geq n\} \\
(\leqslant n\ r\ C)^{\mathcal{I}} &:= \{d \in \Delta \mid \#\{e \mid (d,e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\} \leq n\}
\end{aligned}$$

where $\#S$ denotes the cardinality of the set S. Table 2.1 presents a condensed view on the syntax and semantics of $\mathcal{ALCQI}$ concepts. △

The fragment $\mathcal{ALCFI}$ of $\mathcal{ALCQI}$ consists of a restricted language in which only existential, universal and functional restrictions (hence the $\mathcal{F}$) can be expressed. More specifically, the syntax of $\mathcal{ALCFI}$ concepts allows only number restrictions of the following form:

$$(\leqslant 0\ r\ C); \qquad (\geqslant 1\ r\ C); \qquad (\leqslant 1\ r\ C).$$

$\mathcal{ELIF}$ is the fragment of $\mathcal{ALCFI}$ that disallows disjunctions. Other fragments of interest in this thesis are the logic $\mathcal{ELI}$ that further disallows for functionality constraints, $\mathcal{ELI}_\bot$ which allows $\bot$ in concept descriptions, and $\mathcal{EL}$ that only supports conjunctions of concepts and qualified existential restrictions.

The *DL-Lite* family

The family of *DL-Lite* description logics was first proposed by Calvanese et al. [40], and later extended by Poggi et al. [109] and Artale et al. [9]. *DL-Lite* was specifically designed to express important types of constraints used in conceptual modeling formalisms such as UML class diagrams and ER models (cf. Section 2.2.3) while maintaining low computational complexity of reasoning [8, 9]. Furthermore, *DL-Lite* has also positioned itself as a prominent ontology language. Notably, it forms the basis of OWL 2 QL, one of the three profiles of OWL 2[1] which is the standard ontology language for specific requirements in the Semantic Web.

We next start by presenting the syntax of the *DL-Lite* concept language *DL-Lite*$^{\mathcal{N}}$, and then we will introduce other members of the *DL-Lite* family as variations of *DL-Lite*$^{\mathcal{N}}$. *DL-Lite*$^{\mathcal{N}}$ allows for representing conjunctions, a limited form of negation, inverse roles and unqualified number restrictions.

[1] http://www.w3.org/TR/owl2-profiles/

Concept and Role Constructors		
Name	**Syntax**	**Semantics**
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
disjunction	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
existential restriction	$\exists r.\, C$	$\{d \in \Delta^{\mathcal{I}} \mid \exists e : (d,e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}$
value restriction	$\forall r.\, C$	$\{d \in \Delta^{\mathcal{I}} \mid \forall e : (d,e) \in r^{\mathcal{I}} \rightarrow e \in C^{\mathcal{I}}\}$
at least restriction	$(\geqslant n\ r\ C)$	$\{d \in \Delta \mid \#\{e \mid (d,e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\} \geq n\}$
at most restriction	$(\leqslant n\ r\ C)$	$\{d \in \Delta \mid \#\{e \mid (d,e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\} \leq n\}$
inverse role	p^-	$\{(d,e) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (e,d) \in p^{\mathcal{I}}\}$
Axioms		
concept assertion	$A(a)$	$a^{\mathcal{I}} \in A^{\mathcal{I}}$
role assertion	$p(a,b)$	$(a,b) \in p^{\mathcal{I}}$
concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
role inclusion	$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$

Table 2.1: Syntax and semantics of description logics

Definition 2.3 (*DL-Lite*$^{\mathcal{N}}$ concepts)**.** A *DL-Lite*$^{\mathcal{N}}$ *role* is either a role name $p \in \mathsf{N_R}$ or the inverse of a role name $p \in \mathsf{N_R}$. *DL-Lite*$^{\mathcal{N}}$ *concepts* are defined by the following rules:

$$B \ ::= \ \bot \ \mid \ A \ \mid \ (\geqslant n\ r)$$

$$C_1, C_2 \ ::= \ B \ \mid \ \neg B \ \mid \ C_1 \sqcap C_2$$

where A ranges over $\mathsf{N_C}$, r is a role, and n is a positive integer.

Concepts of the form $(\geqslant n\ r)$ are abbreviations for concepts $(\geqslant n\ r\ \top)$ and called *unqualified number restrictions*. Analogously, $\exists r$ is as an abbreviation for $(\geqslant 1\ r\ )$, which is called a *domain* (or *range* if r is the inverse of a role name) restriction. Concepts of the form B are called *basic concepts*, and those of the form C_1, C_2 are called *complex concepts*. △

Note that the restricted use of negation in *DL-Lite*$^{\mathcal{N}}$ in complex concepts disallows to express disjunction, and therefore *DL-Lite*$^{\mathcal{N}}$ is not Boolean closed. For some applications in conceptual modeling, as we see later (cf. Section 2.1.6), the presence of Boolean connectors may become useful. Hence, the extension *DL-Lite*$^{\mathcal{N}}_{Bool}$ of *DL-Lite*$^{\mathcal{N}}$ in which complex concepts are either basic

concepts or Boolean combinations of complex concepts has also been considered.

$$C_1, C_2 \ ::= \ B \ \mid \ \neg C \ \mid \ C_1 \sqcap C_2 \qquad [\ \textit{DL-Lite}_{Bool}^{\mathcal{N}} \quad \textit{concepts}\ ]$$

The presence of Boolean combinations in $\textit{DL-Lite}_{Bool}^{\mathcal{N}}$ comes however at the price of increasing the complexity of reasoning as we will discuss later in this section.

2.1.2 Terminological Knowledge: TBoxes

One important aspect of a DL ontology is given by the operations used to build the terminology. Indeed, such operations are closely related to the forms and the meaning of the declarations allowed in the TBox. One of the most basic forms of declarations considered in the literature are *concept definitions*, which as suggested by the name, allow to define a new concept in terms of atomic concepts or other previously defined concepts. For example, we can define *'inorganic fertilizers'* as *'fertilizers with mineral composition'* by including the following declaration in the TBox:

$$\mathsf{InorganicFertilizer} \equiv \mathsf{Fertilizer} \sqcap \exists \mathsf{hasComposition}.\mathsf{Mineral}.$$

Concept definitions are intended to express (logical) equivalence between concepts by asserting both necessary and sufficient conditions required to hold for the instances of a certain concept.

Definition 2.4. (TBox) Let $\mathcal{L}$ be a description language. Let C_1 and C_2 be concepts in $\mathcal{L}$, and r, s roles in $\mathcal{L}$. A *concept inclusion (CI) in* $\mathcal{L}$ is an expression of the form $C_1 \sqsubseteq C_2$. An *$\mathcal{L}$-TBox* $\mathcal{T}$ is a finite set of concept inclusions. In cases where the description language is clear from the context or irrelevant, we will use the term TBox instead of $\mathcal{L}$-TBox. △

Intuitively, a CI $C_1 \sqsubseteq C_2$ states that if some object is an instance of C_1, then it must be also an instance of C_2. For example, the knowledge that *'cereals are a kind of crop, but not all crops are cereals'* can be represented by the concept inclusion:

$$\mathsf{Cereal} \sqsubseteq \mathsf{Crop}$$

Note also that a concept definition can be expressed by two concept inclusions. For example, to establish the equivalence between the terms *'inorganic fertilizer'* and *'mineral fertilizer'* one could assert the following concept inclusions

$$\mathsf{InorganicFertilizer} \sqsubseteq \mathsf{MineralFertilizer} \qquad \mathsf{MineralFertilizer} \sqsubseteq \mathsf{InorganicFertilizer}.$$

The precise meaning of TBox statements is defined in terms of interpretations as follows.

Definition 2.5 (satisfaction relation)**.** Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation and $\mathcal{T}$ a TBox, then

- $\mathcal{I}$ *satisfies* a concept inclusion $C_1 \sqsubseteq C_2 \in \mathcal{T}$, denoted as $\mathcal{I} \models C_1 \sqsubseteq C_2$, if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
- $\mathcal{I}$ is a *model of* $\mathcal{T}$, denoted as $\mathcal{I} \models \mathcal{T}$, if $\mathcal{I}$ satisfies every concept inclusion $C_1 \sqsubseteq C_2 \in \mathcal{T}$.

△

Other forms of TBox axioms that we will consider include *role inclusions* which are expressions of the form

$$r \sqsubseteq s,$$

and whose semantics are defined analogously, i.e., an interpretation $\mathcal{I}$ *satisfies* $r_1 \sqsubseteq r_2$ if $r_1^{\mathcal{I}} \subseteq r_2^{\mathcal{I}}$.

Once the terminology is in place, it can be used to infer (implicit) knowledge about the concepts and roles described in it. One of the basic inference tasks is *classification*, which amounts to place a concept (or role) expression in the proper place in a taxonomic hierarchy of concepts (or roles). Classification can be accomplished by verifying the subsumption relation between each pair of concepts within the terminology. More generally, deduction services for TBoxes can be viewed as *logical implication* which comprise verifying whether a generic relationship (for example, a subsumption relation between two concepts) is a logical consequence of the declarations in the TBox. The following are *standard reasoning tasks* considered in DLs.

Definition 2.6 (TBox reasoning tasks)**.** Let $\mathcal{T}$ be a TBox, and C, D concepts.

TBox satisfiability: $\mathcal{T}$ is satisfiable if there exists a model $\mathcal{I}$ of $\mathcal{T}$.

Concept satisfiability: A concept C is *satisfiable with respect to* $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is nonempty.

Subsumption: A concept C is *subsumed by a concept* D *with respect to* $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. In this case we write $C \sqsubseteq_{\mathcal{T}} D$ or $\mathcal{T} \models C \sqsubseteq D$.

Equivalence: Two concepts C and D are equivalent with respect to $\mathcal{T}$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. In this case we write $C \equiv_{\mathcal{T}} D$ or $\mathcal{T} \models C \equiv D$.

△

For description languages that provide conjunction ($\sqcap$) and the unsatisfiable concept bottom ($\bot$), one can reduce all the previous TBox reasoning tasks to subsumption.

Proposition 2.1 ([50])**.** *Let C and D be arbitrary concepts and $\mathcal{T}$ (a possibly empty) TBox in a description language $\mathcal{L}$ that contains $\sqcap$ and $\bot$, we have*

- *$\mathcal{T}$ is unsatisfiable if $\top \sqsubseteq_{\mathcal{T}} \bot$;*
- *C is unsatisfiable w.r.t. $\mathcal{T}$ if $C \sqsubseteq_{\mathcal{T}} \bot$; and*
- *C and D are equivalent w.r.t. $\mathcal{T}$ if $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$.*

2.1.3 Assertional Knowledge: ABoxes

Apart from representing knowledge about concepts and the relations holding between them, one can also be interested in representing knowledge about specific individuals in the domain of application. As we discussed at the beginning of the section, this is the role of the ABox. In the ABox, normally two types of assertions about individual names (that represent real objects in the domain of interest) are stated. The first type called *concept assertions* state that a certain individual is an instance of a concept (it belongs to a certain class), whereas the second one are called *role assertions*, and state that a pair of individuals belong to a certain relation. For example, we could assert that *'rice is a crop that grows in China'* by asserting

$$\mathsf{Crop}(\mathsf{rice}), \qquad \mathsf{growsIn}(\mathsf{rice}, \mathsf{China}).$$

Definition 2.7. (ABox) Let C and r be a concept and a role in some concept language $\mathcal{L}$, respectively. Further, let $a, b \in \mathsf{N_I}$ be individual names. A *concept assertion* is an expression of the form $C(a)$, and a *role assertion* is an expression of the form $r(a, b)$. *An $\mathcal{L}$ ABox $\mathcal{A}$* is a finite set of concept and role assertions. △

In a simplified view, an ABox can be seen as an instance of a relational database with only unary or binary relations [109]. However, contrary to the *closed-world* semantics of classical databases, the semantics of ABoxes is an *open-world* semantics since normally knowledge representation systems are applied in situations where one cannot assume that the knowledge is complete. Moreover, the TBox imposes semantic relationships between concepts and roles in the ABox that do not have counterparts in database semantics.

An interpretation $\mathcal{I}$ can be extended to provide semantics for ABoxes by demanding that $\cdot^{\mathcal{I}}$ maps every individual name occurring in the ABox to an element $a^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$. In databases, it is commonly assumed that distinct constants (individual names) denote distinct objects from the domain. This assumption corresponds to the so-called *unique name assumption (UNA)*. Given the scope of this thesis, we will adhere to this assumption. Moreover, we will make the further assumption that every individual name occurring in the ABox is interpreted as itself, that is, $a^{\mathcal{I}} = a$. This corresponds to the *standard name assumption (SNA)*, which clearly implies the UNA.

The semantics of ABox assertion is defined in terms of the satisfaction relation as follows.

Definition 2.8. Let $\mathcal{I}$ be an interpretation. $\mathcal{I}$ *satisfies* $\mathcal{A}$, denoted by $\mathcal{I} \models \mathcal{A}$, if

- for every $C(a) \in \mathcal{A},\ a \in C^{\mathcal{I}}$, and
- for every $p(a, b) \in \mathcal{A},\ (a, b) \in p^{\mathcal{I}}$.

In that case, $\mathcal{I}$ is said to be a *model* of $\mathcal{A}$.

△

A model of an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$ can be understood as an abstraction of a concrete 'world' where the concepts are interpreted as subsets of the domain as required by $\mathcal{T}$ and where the

membership of the individuals to concepts and their relationships with one another in terms of roles respect the assertions in the ABox. Verifying the existence of such a possible world is in fact one of the basic reasoning tasks that involve the ABox.

Definition 2.9 (ABox consistency w.r.t. a TBox)**.** Let $\mathcal{T}$ be a TBox. An ABox $\mathcal{A}$ is *consistent w.r.t.* $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ satisfying $\mathcal{A}$. This is written in symbols as

$$\mathcal{I} \models (\mathcal{T}, \mathcal{A}).$$

△

Besides ABox consistency, one might consider the problem of verifying whether a given individual is an instance of a certain concept. The latter is known as the *instance checking* problem. Formally, an individual a is an *instance* of a concept C if $a \in C^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. One can view an instance check as posing a query over an ontology. In fact, more sophisticated kinds of queries such as database like queries have been studied in the literature. We will discuss ontological query answering in Section 2.1.5.

2.1.4 Horn Description Logics

Horn logics have a long tradition in areas such as logic programing, mainly because of their model theoretical properties and their low computational complexity. In the context of description logics, however, the so-called Horn description logics have been only recently studied and understood [9, 13, 71, 88]. Horn DLs are characterized for having low complexity (tractable) of reasoning with respect to the size of the data (ABox).

Horn-$\mathcal{ALCQI}$ is a fragment of $\mathcal{ALCQI}$ in which the form of the axioms in the TBox is restricted. In fact, those restrictions resemble the ones imposed in the Horn fragment of first-order logic.

Definition 2.10 (Horn-$\mathcal{ALCQI}$ TBoxes)**.** A *Horn-$\mathcal{ALCQI}$ TBox* $\mathcal{T}$ is a set of concept inclusions that can take the following forms:

$$K \sqsubseteq A \qquad K \sqsubseteq \bot \qquad K \sqsubseteq \forall r.K' \qquad K \sqsubseteq (\leqslant 1\ r\ K') \qquad K \sqsubseteq (\geqslant n\ r\ K')$$

where K and K' denote a conjunction of concept names, r denotes an $\mathcal{ALCQI}$ role, and $n \geq 1$. The empty conjunction is equivalent to $\top$. △

We will also be interested in the fragment *Horn-$\mathcal{ALCFI}$* of Horn-$\mathcal{ALCQI}$. A *Horn-$\mathcal{ALCFI}$ TBox* restricts the occurrences of number restrictions to those of the form $K \sqsubseteq (\geqslant n\ r\ K')$ for $n = 1$. Recall that such restrictions correspond to existential restrictions of the form $K \sqsubseteq \exists r.K'$.

Other prominent Horn description logics of interest are those from the *DL-Lite* family. We define the so-called *core* and *Horn DL-Lite* TBoxes.

Definition 2.11 (*DL-Lite* TBoxes)**.** Let B_1, B_2 range over basic *DL-Lite*$^{\mathcal{N}}$ concepts. A *DL-Lite core TBox* is a finite set of concept inclusions of the form:

$$B_1 \sqsubseteq B_2 \qquad \text{or} \qquad B_1 \sqcap B_2 \sqsubseteq \bot. \qquad [\ \textit{DL-Lite}^{\mathcal{N}}_{core}\ ]$$

A *DL-Lite Horn TBox* is a finite set of concept inclusions whose form is restricted to:

$$\bigsqcap_i B_i \sqsubseteq B \qquad [\ \textit{DL-Lite}^{\mathcal{N}}_{Horn}\ ]$$

△

Note that *DL-Lite*$^{\mathcal{N}}_{core}$ is a sublogic of *DL-Lite*$^{\mathcal{N}}_{Horn}$. As before, we will also consider *DL-Lite* TBoxes where the occurrences of number restrictions are restricted. In particular, we consider the sublogics *DL-Lite*$^{\mathcal{F}}_{\alpha}$, for $\alpha \in \{core, Horn\}$, where only number restrictions that express existential restrictions and *global functionality* of roles (hence the $\mathcal{F}$) are allowed. More precisely, global functionality of a role r is expressed by the axiom $(\geqslant r\ 2) \sqsubseteq \bot$, alternatively, we will use the notation

$$(\mathsf{funct}\ r) \qquad [\ \textit{global functionality}\ ]$$

We will also consider extensions of *DL-Lite* with role hierarchies ($\mathcal{H}$), i.e., axioms expressing role inclusions $r \sqsubseteq s$. Hence, we obtain the logics *DL-Lite*$^{\beta,\mathcal{H}}_{\alpha}$, with $\alpha \in \{core, Horn\}$ and $\beta \in \{\mathcal{N}, \mathcal{F}\}$.

2.1.5 Ontological Query Answering

One of the most prominent and recent applications of description logics is ontology base data access (OBDA) [109]. The fundamental inference service in OBDA is answering queries posed to the ABox taking into account the constraints in the TBox. The kind of queries that have most often been considered are conjunctive queries, which are a subclass of first-order queries, and correspond to the commonly used *Select-Project-Join SQL* queries [3]. There are at least two key properties needed for making feasible such an approach:

(i) efficiency of query evaluation, with the ideal target being traditional database query processing, and

(ii) that query evaluation can be done by leveraging the relational databases technology already used for storing the data.

A first-order logic (FOL) *query* over an ontology is a, possibly open, FOL formula $\varphi(\vec{x})$ whose predicate symbols are concept names and role names occuring in the ontology. In this thesis, we will be interested in particular forms of FOL queries. We start by introducing positive existential queries.

Definition 2.12 (positive existential queries)**.** Let $\mathsf{N_V}$ be a countably infinite set of variables disjoint from $\mathsf{N_C}, \mathsf{N_R}$ and $\mathsf{N_I}$. A *term* t is an element from $\mathsf{N_V} \cup \mathsf{N_I}$. A *positive existential query (PEQ)* $q(\vec{x})$ is a first-order formula of the form $\exists \vec{y}\, \varphi(\vec{x}, \vec{y})$ with free variables $\vec{x} = x_1, \ldots, x_n$ and the syntax of φ is determined by the following rules:

$$\varphi \;:=\; A(t) \;\mid\; p(t_1, t_2) \;\mid\; \varphi_1 \wedge \varphi_2 \;\mid\; \varphi_1 \vee \varphi_2 \tag{2.1}$$

where $A \in \mathsf{N_C}$ is a concept name, $p \in \mathsf{N_R}$ is a role name, and t_1, t_2 are terms. Expressions of the form $A(t)$ are called *concept atoms* and those of the form $p(t_1, t_2)$ *role atoms*. The *arity* of $q(\vec{x})$ is the number of variables in $\vec{x}$, and those variables are called *answer variables*. A *Boolean PEQ query* is a PEQ containing no answer variables. We write q instead of $q(\vec{x})$ if $\vec{x}$ is clear from the context.

A *conjunctive query (CQ)* $\exists \vec{y}\, \varphi(\vec{x}, \vec{y})$ is a positive existential query where φ is a conjunction of (concept or role) atoms. △

For many applications it becomes useful to associate a labeled graph to a conjunctive query.

Definition 2.13 (query graph)**.** Let q be a (Boolean) CQ, and let $\mathsf{terms}(q)$ denote the set of terms occurring in q. The *query graph* of q is the directed multigraph $G = (V, E)$ with edge labels and multiple node labels such that $V = \mathsf{terms}(q)$. Each node t is labeled with A iff q contains unary the atom $A(t)$, and E contains a directed edge (t_1, r, t_2) iff q contains the binary atom $r(t_1, t_2)$. △

Extensions of CQs with some form of negation have been studied for ontological query answering in DLs, as well as in areas related to management of incomplete information [6, 18, 54, 96, 116]. Fundamental insights from the database area have shown, however, that *negation* tends to make queries difficult to evaluate or even undecidable. For that reason, only limited forms of negation (which may still be of interest in practice) have been considered. In particular, the addition of inequalities ($\neq$) between two terms, and the so-called *safe* and *guarded* negation.

Definition 2.14 (queries with negation)**.** A *conjunctive query with inequalities (CQ$^{\neq}$)* is a first-order formula of the form

$$\exists \vec{y}\, \varphi(\vec{x}, \vec{y}) \wedge \psi(\vec{x}, \vec{y}),$$

where $\varphi(\vec{x}, \vec{y})$ is a conjunction of (concept or role) atoms, and $\psi(\vec{x}, \vec{y})$ is a conjunction of *inequality atoms* of the form $t \neq t'$, with t, t' terms.

A *conjunctive query with safe negation* (CQ$^{\neg s}$) is a formula

$$\exists \vec{y}\, \varphi(\vec{x}, \vec{y}),$$

where $\varphi(\vec{x}, \vec{y})$ is a conjunction of (positive) atoms or negated atoms, and such that each variable in a negated atom occurs in at least one positive atom.

Let $q = \exists \vec{y}\, \varphi(\vec{x}, \vec{y})$ be a first-order formula with $\varphi(\vec{x}, \vec{y})$ a conjunction of positive and negative atoms; and let α be a negative atom in q. A *guard for α in q* is a positive atom in q that

contains *all* the variables occuring in α. The query q is said to be a *conjunctive query with guarded negation* if every negated atom in q has a guard.

A Boolean CQ extended with negations is defined in the natural way. △

Differently from answering queries posed to a (complete) database instance $\mathcal{D}$, where one can regard $\mathcal{D}$ as an interpretation (i.e., a first-order structure), *ontological query answering* can be thought of as querying various databases $\mathcal{D}_1, \mathcal{D}_2, \ldots$ which intuitively correspond to all possible models of an ontology $(\mathcal{T}, \mathcal{A})$. In fact, ontological query answering is comparable to querying over incomplete databases [76, 98, 127], or query answering on databases under the open world assumption [1, 129].

Example 1. For example, let the TBox $\mathcal{T}$ be

$$\mathsf{Fruit} \sqsubseteq \mathsf{Produce} \tag{2.2}$$
$$\mathsf{Crop} \sqsubseteq \mathsf{Produce} \tag{2.3}$$
$$\mathsf{Produce} \sqsubseteq \exists.\mathsf{growsIn} \tag{2.4}$$
$$\exists.\mathsf{growsIn} \sqsubseteq \mathsf{Produce} \tag{2.5}$$
$$\mathsf{Location} \sqsubseteq \mathsf{Country} \sqcup \mathsf{Region} \tag{2.6}$$
$$\mathsf{Country} \sqcap \mathsf{Region} \sqsubseteq \bot \tag{2.7}$$

Let the available (incomplete) database about the concepts in the $\mathcal{T}$ be specified by the following ABox $\mathcal{A}$

$$\mathsf{growsIn}(\mathsf{MANGO}, \mathsf{PAKISTAN}), \qquad \mathsf{Fruit}(\mathsf{MANGO}),$$
$$\mathsf{growsIn}(\mathsf{DURIAN}, \mathsf{SOUTH\text{-}ASIA}).$$

Now, let $q(x) = \mathsf{Produce}(x)$ be the simple query that asks for '*all the known produce*' in the database. Then the answers to q are the ones in the set $\{\mathsf{MANGO}, \mathsf{DURIAN}\}$. Note that although this is not explicitly asserted in $\mathcal{A}$, by (2.2) every instance of Fruit is an instance of Produce, and by (2.5) every object that has a growsIn-successor is an instance of Produce. For a slightly more complex query take $q'(x) = \exists y.\, \mathsf{Fruit}(y) \wedge \mathsf{growsIn}(y, x)$, that asks for '*all the places that are known to grow fruits*'. Then, the only answer to this question is in the set $\{\mathsf{PAKISTAN}\}$. This is the case since the constraints imposed by $(\mathcal{T}, \mathcal{A})$ do not imply that DURIAN is an instance of Fruit; although there is model of the ontology in which this is the case. ⅄

Therefore, in the latter setting, a new notion of query answering is required. In order to capture this notion formally, the so-called *certain answers* semantics has been introduced in the relevant literature [1, 23]. Roughly, the notion of certain answers amounts to logical entailment. We make this more precise in the following.

Definition 2.15. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation, and $q(\vec{x}) = \exists \vec{y}\, \varphi(\vec{x}, \vec{y})$ an n-ary positive existential query. Further let $\mathsf{bvars}(q)$ denote the set $\vec{y}$ of bounded variables in q. A *variable*

assignment for q in $\mathcal{I}$ is a mapping $\pi : \mathsf{bvars}(q) \to \Delta^{\mathcal{I}}$. A tuple of name individuals $\vec{a} = a_1, \ldots, a_n$ is an *answer to q in $\mathcal{I}$*, if there is a variable assignment π such that $\mathcal{I}$ *satisfies* $q(\vec{a}) = \exists \vec{y}.\varphi(\vec{a}, \vec{y})$ under π. We will write

$$\mathcal{I} \models^{\pi} \varphi(\vec{a}, \vec{y});$$

such an assignment π is called a *match* for q in $\mathcal{I}$. The set of answers to q in $\mathcal{I}$ is denoted by $\mathsf{ans}(q, \mathcal{I})$. Further, let $(\mathcal{T}, \mathcal{A})$ be an ontology. $\vec{a}$ is called a *certain answer to q w.r.t.* $(\mathcal{T}, \mathcal{A})$, if for every model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$, there is a match π for q in $\mathcal{I}$, and in that case we write

$$(\mathcal{T}, \mathcal{A}) \models q(\vec{a}).$$

The set of *certain answers to q w.r.t.* $(\mathcal{T}, \mathcal{A})$ is defined as $\mathsf{cert}(q, (\mathcal{T}, \mathcal{A})) := \{\vec{a} \subseteq \mathsf{ind}(\mathcal{A}) \mid (\mathcal{T}, \mathcal{A}) \models q(\vec{a})\}$. △

DL-Lite is particularly well-suited to provide the logical framework for accessing data [109, 114], as most of the *DL-Lite* were developed to meet the requirements (i) and (ii) above described. In spite of being rather unexpressive, DLs of the *DL-Lite* family provide means for capturing interesting modeling features for conceptual data modeling (see next Section). More expressive ontology languages have also been considered for ontology-based data access [29, 53, 104, 126].

Note that computing the certain answers to a given query q amounts to logical entailment w.r.t. a given ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$. Hence, in principle, to effectively compute $\mathsf{cert}(q, \mathcal{O})$ it is necessary to consider all possible models of $\mathcal{O}$, which may be far from feasible. In order to overcome this apparent difficulty, it has been observed that for ontologies expressed in certain description logics it is possible to construct a so-called canonical (or universal) model (provided that the ontology is satisfiable) [40, 51, 88, 104].

Given an ontology $\mathcal{O}$, an interpretation $\mathcal{U}$ is a *canonical model* if the following hold

(i) $\mathcal{U} \models \mathcal{O}$ (it is a model of $\mathcal{O}$); and

(ii) for every model $\mathcal{I}$ of $\mathcal{O}$ there is a *homomorphism* from $\mathcal{U}$ to $\mathcal{I}$.

Hence, procedures for constructing canonical models provide a formal tool for ontological query answering. Since PEQs are preserved under homomorphisms [119], the problem of computing $\mathsf{cert}(q, \mathcal{O})$ can be reduced to the problem of computing the answers to q in the canonical model $\mathcal{U}$ of $\mathcal{O}$.

Most of the approaches for constructing the canonical model of an ontology $\mathcal{O}$ can be seen as adaptations of the *chase procedure* [3, 49], which is a fundamental algorithmic tool in databases introduced for checking implication of dependencies [97], and later used for checking query containment [77]. Similarly to the chase procedure, the basic idea behind the construction of the canonical model of an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ is to *complete in a minimal fashion* the ABox $\mathcal{A}$ in such a way that the axioms in $\mathcal{T}$ are satisfied. However, the construction of canonical models is in general neither convenient nor possible since these models may be infinite due to the presence of cyclic concept inclusions in the TBox $\mathcal{T}$. We will discuss specific constructions for canonical models of Horn ontologies in Chapter 5. The following is a consequence of various results in the relevant literature [9, 40, 55, 88, 95].

Theorem 2.2. *Let $\mathcal{T}$ be an $\mathcal{L}$-TBox for $\mathcal{L} \in \{\text{DL-Lite}_{Horn}^{\mathcal{N}}, \text{DL-Lite}_{Horn}^{\mathcal{H}}, \text{Horn-}\mathcal{ALCQI}\}$, and let $\mathcal{A}$ be an ABox. If $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ is satisfiable, then there exists a canonical model $\mathcal{U}$ of $\mathcal{O}$. Further, let q be a PEQ and $\vec{a}$ a tuple of individuals in $\mathcal{A}$, then the following are equivalent:*

- $\vec{a} \in \mathsf{cert}(q, \mathcal{O})$;
- $\mathcal{U} \models^{\pi} q(\vec{a})$.

2.1.6 Computational Complexity of Reasoning

The computational complexity of the reasoning problems discussed so far can be analyzed with respect to different complexity measures, which depend on those parameters of the problem that are regarded to be the input and those that are considered to be fixed. For satisfiability and instance checking, the parameters to consider are the size of the TBox and the size of the ABox. More precisely, *the size of a TBox* $\mathcal{T}$ (or and ABox $\mathcal{A}$) is the number of symbols in $\mathcal{T}$ (or in $\mathcal{A}$) denoted by $|\mathcal{T}|$ (or by $|\mathcal{A}|$). The *size of an ontology* $(\mathcal{T}, \mathcal{A})$ is simply given by $|\mathcal{T}| + |\mathcal{A}|$. In analyzing the computational complexity of a reasoning problem over an ontology, there is a distinction between data complexity and combined complexity [128]. Whenever the complexity is measured with respect to the size of all inputs to the problem then one is interested in the *combined complexity*. On the other hand, if only the ABox is considered as an input – while the TBox is regarded to be fixed – then *data complexity* is of interest. Determining combined complexity is relevant when one is interested in the complexity of algorithms for developing and testing an ontology. On the other hand, determining data complexity is of interest in all those cases where the size of the ABox (or the data) is considerably bigger than the TBox– and hence the TBox size is inconsequential, and can be considered as fixed. The latter is indeed the case for example in the context of ontology-based data access [68, 109], and other data intensive applications [23, 89]. The complexity classes relevant for the problems discussed in this thesis are the following:

$$\text{AC}^0 \subsetneq \textsc{LogSpace} \subseteq \textsc{NLogSpace} \subseteq \textsc{PTime} \subseteq \text{NP} \subseteq \textsc{ExpTime}$$

We refer the reader to [107] for the formal definition of these classes and an extensive textbook treatment of computational complexity notions, which is out of the scope of this thesis.

We will present the results on the complexity of reasoning for the description logics introduced in this chapter. A summary of those results is presented in Table 2.2. We start with the combined complexity of the decision problems related to standard reasoning tasks from Definitions 2.6 and 2.9, specifically, subsumption and ABox consistency. Recall that other TBox reasoning tasks can be reduced to subsumption (see Proposition 2.1).

Definition 2.16. We consider two main decision problems:

- ***subsumption:*** Given an ontology $(\mathcal{T}, \mathcal{A})$ and concepts C, D as input, *decide* whether

$$(\mathcal{T}, \mathcal{A}) \models C \sqsubseteq D.$$

- ***consistency:*** Given an ontology $(\mathcal{T}, \mathcal{A})$ as input, *decide* whether $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$.

△

Observe that, differently from what is stated in Definition 2.6, ***subsumption*** refers to implication w.r.t. to an ontology $(\mathcal{T}, \mathcal{A})$. Nevertheless, subsumption w.r.t. a TBox alone can be seen as an instance of the problem ***subsumption*** with an empty ABox $\mathcal{A}$.

We summarize next the results regarding the computational complexity of the decision problems in the definition above.

Theorem 2.3 ([9, 15, 32, 79])**.** *Deciding* ***subsumption/consistency*** *for $\mathcal{ALCQI}$, $\mathcal{ELI}$, Horn-$\mathcal{ALCQI}$ and DL-Lite$_\alpha^{\mathcal{N},\mathcal{H}}$ for $\alpha = \{core, Horn, Bool\}$ ontologies is* ExpTime *complete.*

The following results provide the reason why some logics from the *DL-Lite* family are called lightweight.

Theorem 2.4 ([9, 40])**.** *Deciding* ***subsumption/consistency*** *is*

- NLogSpace *complete for DL-Lite$_{core}^{\beta}$,*
- PTime *complete for DL-Lite$_{Horn}^{\beta}$ and $\mathcal{EL}$,*
- NP *complete for DL-Lite$_{Bool}^{\beta}$ ontologies, with $\beta \in \{\mathcal{F}, \mathcal{N}, \mathcal{H}\}$.*

Horn description logics, as mentioned in the previous section, were developed for data intensive applications in which it is desirable that reasoning is at least tractable, i.e., solvable in polynomial time w.r.t. to the size of the data (ABox). The main reasoning task considered in that context is ontological query answering. The main decision problem is the following:

Definition 2.17 (***query answering:***)**.** Given an ontology $(\mathcal{T}, \mathcal{A})$, a tuple of individuals $\vec{a}$, and a query q, *decide* whether

$$\vec{a} \in \mathsf{cert}(q, \mathcal{T}, \mathcal{A}).$$

△

The **query answering** problem has also been denominated as the *recognition problem* associated to the *query evaluation problem* [39], i.e., given a$(\mathcal{T}, \mathcal{A})$ and a query q, compute the set $\mathsf{cert}(\mathcal{T}, \mathcal{A})$. The following theorem summarizes the results on the complexity of ontological query answering on Horn DLs.

Theorem 2.5 ([9, 51])**.**

Data Complexity: *PEQ/ UCQ answering is in* AC^0 *for DL-Lite$_{core}^{\beta}$ and DL-Lite$_{Horn}^{\beta}$ TBoxes, with $\beta \in \{\mathcal{F}, \mathcal{N}, \mathcal{H}\}$; and it is* PTime *complete for $\mathcal{EL}, \mathcal{ELI}$ and Horn-$\mathcal{ALCQI}$ TBoxes.*

Combined Complexity: *PEQ/ UCQ answering is* ExpTime*-complete for Horn-$\mathcal{ALCQI}$ ontologies.*

Language	Combined complexity	Data Complexity
	subsumption	***query answering***
$\mathcal{ALCQI}$ $\mathcal{ALCFI}$ *DL-Lite*$_\beta^{\mathcal{F},\mathcal{H}}$ *DL-Lite*$_\beta^{\mathcal{N},\mathcal{H}}$ $\beta \in \{core,\ Horn,\ Bool\}$	ExpTime [9, 32]	coNP [9, 71]
DL-Lite$_{Bool}^{\mathcal{N}}$ *DL-Lite*$_{Bool}^{\mathcal{F}}$	NP [9]	coNP [9]
$\mathcal{ELI}$ Horn-$\mathcal{ALCQI}$	ExpTime [15]	PTime [51, 88]
$\mathcal{EL}$	PTime [13]	PTime [38, 116]
DL-Lite$_{Horn}^{\mathcal{F}}$ *DL-Lite*$_{Horn}^{\mathcal{N}}$ *DL-Lite*$_{Horn}^{\mathcal{H}}$	PTime [9]	in AC^0 [9]
DL-Lite$_{core}^{\mathcal{N}}$ *DL-Lite*$_{core}^{\mathcal{F}}$ *DL-Lite*$_{core}^{\mathcal{H}}$	NLogSpace [9]	in AC^0 [9]

Table 2.2: Complexity of reasoning in description logics

An important property when considering ontological query answering regards the so-called *FO-rewritability*. The problem of ***query answering*** for a DL language $\mathcal{L}$ is *FO-rewritable* if for every query q and ontology $(\mathcal{T}, \mathcal{A})$ in $\mathcal{L}$, there exists a first-order query $\widehat{q}$ such that $(\mathcal{T}, \mathcal{A}) \models q$ iff $\mathcal{A} \models \widehat{q}$.

Theorem 2.6. ***query answering*** *in DL-Lite*$_\beta^\alpha$*, with* $\alpha \in \{\mathcal{N}, \mathcal{F}, \mathcal{H}\}$ *and* $\beta \in \{core,\ Horn\}$ *is FO-rewritable.*

2.1.7 Finite Model Reasoning in DLs

For description languages containing inverse roles and functionality of roles, a TBox may admit only models with an infinite domain, or the axioms in the TBox may cause that some concepts

are only satisfiable in an infinite model. Let us consider the following example to illustrate this fact.

Example 2. Let $\mathcal{T}$ be the following $\mathcal{ALCFI}$ TBox:

$$A \sqsubseteq B \sqcap \neg \exists s^-$$
$$B \sqsubseteq \exists s \sqcap \forall s.B \sqcap (\leqslant 1\ s^-)$$

In a model of this TBox, an instance of A can have no s-predecessors, while each instance of B can have at most one. Therefore, for a model $\mathcal{I}$ of $\mathcal{T}$, the existence of an object $d \in A^{\mathcal{I}}$ implies the existence of an infinite sequence of objects $e_1, e_2, \ldots \in B^{\mathcal{I}}$, such that $(d, e_1) \in s^{\mathcal{I}}$, and $(e_i, e_{i+1}) \in s^{\mathcal{I}}$. This clearly, means that A can only be satisfied in an interpretation with an infinitely large domain. $\overline{\wedge}$

A logic is said to have the *finite model property (FMP)* if every satisfiable formula of the logic admits a *finite model*, i.e., a model with a finite domain. The example above shows that very simple DLs in which functionality and inverse roles are present, and TBox axioms with existential restrictions on the right hand side are allowed, lack the finite model property.

For all logics that lack the FMP, reasoning with respect to unrestricted and finite models are fundamentally different tasks. Hence, this needs to be explicitly taken into account when devising reasoning procedures.

In this section, we present the known results regarding finite model reasoning in DLs. We say that an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a *finite model* of $\mathcal{T}$ if $\mathcal{I} \models \mathcal{T}$ and $\Delta^{\mathcal{I}}$ is finite. More precisely, when we refer to *finite model reasoning* we address the following reasoning tasks.

Definition 2.18 (finite model reasoning)**.**

finite TBox satisfiability: Given a TBox $\mathcal{T}$, *decide* whether there is a finite model $\mathcal{I}$ of $\mathcal{T}$.

finite concept satisfiability: Given a TBox $\mathcal{T}$ and concept C, *decide* whether there is a finite model $\mathcal{I}$ of $\mathcal{T}$, such that $C^{\mathcal{I}} \neq \emptyset$.

finite subsumption: Given an ontology $(\mathcal{T}, \mathcal{A})$ and concepts C, D, *decide* whether for every finite model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$ it holds that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, which is denoted with $(\mathcal{T}, \mathcal{A}) \models_{\mathsf{fin}} C \sqsubseteq D$.

finite (ABox) consistency: Given an ontology $(\mathcal{T}, \mathcal{A})$, *decide* whether there is a finite model of $\mathcal{T}$ such that $\mathcal{I} \models \mathcal{A}$.

$\triangle$

Notably, when reasoning w.r.t. finite models some properties that are essential for developing algorithms and establishing the complexity on reasoning in the case of unrestricted reasoning fail. Such properties include the tree model property, or similar properties that are based on 'unravelling' structures. Observe that whenever a (finite) model has a cycle, the unravelling of such a model into a tree generates an infinite structure.

In particular, finite model reasoning in $\mathcal{ALCQI}$ can be realized using techniques that are based on the encoding of the TBox using systems of linear inequalities [31, 94]. This technique provides a reduction of finite satisfiability of an $\mathcal{ALCQI}$ TBox to the problem of solving a linear system of inequalities over the integers. Since solving such systems of inequalities can be done using linear programming techniques and the size of the systems is exponential w.r.t. the size of the TBox [94], we have the following complexity result establishing that reasoning is as complex as in the unrestricted case.

Theorem 2.7 ([94]). *Finite concept satisfiability in $\mathcal{ALCQI}$ is* ExpTime*-complete.*

Recall that for $\mathcal{ALCQI}$ all the reasoning tasks are reducible to each other (see, e.g., Proposition 2.1). Moreover, these reductions do not depend on whether the models are finite or not. Hence, finite model reasoning is ExpTime-complete for $\mathcal{ALCQI}$.

For less expressive DLs, the study of finite model reasoning has been limited to *DL-Lite*. Given that *DL-Lite* was designed specifically to express typical constraints of the database scenario, such as *inclusion dependencies* and *unary functional dependencies*, some of the results and techniques for finite entailment for database dependencies [45] can be transferred to *DL-Lite*. Based on the latter observation, Rosati [117] showed that, as in the case of expressive DLs, finite model reasoning in *DL-Lite*$_{core}^{\mathcal{F}}$ has the same complexity that reasoning w.r.t. unrestricted models. More importantly from the algorithmic view point, Rosati provided a reduction of finite model reasoning to unrestricted reasoning for the *DL-Lite*$_{core}^{\mathcal{F}}$ case. In more detail, given a *DL-Lite*$_{core}^{\mathcal{F}}$ TBox $\mathcal{T}$ it is possible to extend $\mathcal{T}$ into a TBox $\mathcal{T}'$ that captures the finite model entailments w.r.t. $\mathcal{T}$.

Theorem 2.8 ([117]). *Finite model reasoning in DL-Lite*$_{core}^{\mathcal{F}}$ *is reducible to reasoning over arbitrary models in DL-Lite*$_{core}^{\mathcal{F}}$*.*

This reduction provides then computational complexity boundaries for finite model reasoning in *DL-Lite*$_{core}^{\mathcal{F}}$. Note that the lower bound follows from the complexity of finite model reasoning in *DL-Lite*$_{core}$, which is the same as for unrestricted reasoning since this logic has the FMP.

Theorem 2.9 ([117]). *Finite TBox satisfiability (and concept subsumption w.r.t.) DL-Lite*$_{core}^{\mathcal{F}}$ *TBoxes is in* NLogSpace*. Further, deciding finite query entailment in DL-Lite*$_{core}^{\mathcal{F}}$ *is in* AC^0 *w.r.t. data complexity.*

One of the advantages of the finite model reasoning approach taken by Rosati is that it is possible to resort to available means for reasoning w.r.t. unrestricted models to perform finite model reasoning.

Surprisingly, for *DL-Lite*$_{Horn}^{\mathcal{N}}$ the complexity of finite model reasoning is not guaranteed to be the same as for unrestricted model reasoning. This is the case as the combination of unqualified numbers restrictions and the finite model assumption are enough to encode disjunction. Hence, it is possible to encode, e.g., the SAT problem, which is known to be NP-complete [107].

Lemma 2.10 ([86]). *Finite model satisfiability of DL-Lite*$_{Horn}^{\mathcal{N}}$ *TBoxes is* NP*-hard.*

Language	Combined complexity	Data Complexity
	subsumption	***query answering***
$\mathcal{ALCQI}$ $\mathcal{ALCFI}$	ExpTime [94]	coNP-hard ($\star$)
DL-Lite$_{core}^{\mathcal{F},\mathcal{H}}$	in ExpTime [94]	?
DL-Lite$_{Bool}^{\mathcal{N}}$ *DL-Lite*$_{Bool}^{\mathcal{F}}$	NP-hard ($\star$)	coNP-hard ($\star$)
Horn-$\mathcal{ALCQI}$	in ExpTime [94]	PTime-hard [38]
DL-Lite$_{Horn}^{\mathcal{F}}$	PTime-hard ($\star$)	?
DL-Lite$_{Horn}^{\mathcal{N}}$	NP-hard [86]	?
DL-Lite$_{core}^{\mathcal{F}}$	NLogSpace [117]	in AC^0 [117]

($\star$)These complexity boundaries follow from those of the corresponding sub-logics without functionality or number restrictions, which have the FMP.

Table 2.3: Complexity of finite model reasoning

Finally, Table 2.3 summarizes the complexity boundaries for finite model reasoning on the DLs lacking the FMP discussed in this section[2].

2.2 Conceptual Modeling Formalisms

We start this section with a brief introduction to the Entity-Relationship model, which is the most popular high-level conceptual data model. Afterwards, we introduce the class diagrams of the Unified Modeling Language (UML), which is the most prominent and popular object modeling methodology.

2.2.1 Entity-Relationship Schemata

The *Entity-Relationship (ER) model* is one of the best known and widely used semantic models in industrial applications. The ER model was proposed by Chen in 1976 [44], and subsequently several variants and extensions which differ in minor aspects related to expressiveness and notation have been introduced [124]. We will present the syntax of *Enhanced Entity Relationship (EER) schemata* as presented, e.g., in [36].

The ER model was originally proposed as a data model adopting the somewhat natural view that the real world consists of entities and relationships. In the ER model the domain of interest

[2]Clearly, for DLs enjoying the FMP the boundaries in Table 2.2 apply.

is modeled by means of an ER *schema*, which can be represented graphically thus making them useful for the visualization and design of the data dependencies. ER schema have a first-order logic formalization, and moreover they can also be formalized by description logics as we will describe later on.

The basic modeling elements used to define EER schemata are *entities, and relationships*. An *entity* denotes a set of objects that share common properties. A *relationship* represents an association among various entities. Relationships usually have certain constraints that limit the possible combinations of entities that may participate in the relationship. *Cardinality constraints* restrict the minimal and maximal number of times each instance of an entity is allowed to participate as certain component of a relationship. Such constraints can be used to specify, for example, mandatory participation on a relation $(1 : \infty)$ and functionality $(1 : 1)$. Additionally, isa relations are used to represent inclusion between entities and relationships, and therefore the inheritance of properties from a more general entity to a more specific one.

In more detail, an *EER schema* $\mathcal{S}$ is a collection of entity, relationship, and attribute definitions over an *alphabet of symbols* partitioned into a set of *entity symbols* $\mathbf{E}$, a set of *relationship symbols* $\mathbf{R}$ and a set of *domain symbols* $\mathbf{D}$.

An *entity definition* has the form:

```
define entity E
    isa: E_1, ... E_k
    participates: (m_1..n_1) as R_1[c_1],
                  ...
                  (m_ℓ..n_ℓ) as R_ℓ[c_ℓ]
end
```

where $E \in \mathbf{E}$ is the entity to be defined, the isa clause specifies the set of entities of which E is a subset, and each statement in the participates clause specifies that E participates in at least m_i and at most n_i tuples as the component c_i of relationship R_i, for $i \in \{1, \ldots, \ell\}$.

A *relationship definition* is declared as follows:

```
define relationship R among E_1, ..., E_n
    isa: R_1, ..., R_ℓ
end
```

where $R \in \mathbf{R}$ is the relationship to be defined, the entities $E_1, \ldots, E_n \in \mathbf{E}$ listed in the *among* clause are those among which R is defined, i.e., the component c_i of R is an instance of E_i, for $i \in \{1, \ldots, n\}$. The number of entities in the among clause is the *arity* of R. Note that there might be indices i, j with $E_i = E_j$, hence the need to specify the label of the component in R. The isa clause specifies that R is related via the isa relationship to $R_1, \ldots, R_\ell$. For example, Figure 2.1 shows an EER schema for modeling our agricultural domain.

EER schemata have a diagrammatic notation associated to them. In the commonly accepted notation for *EER diagrams*, entities are depicted as rectangles and relationships are represented

define entity *Produce*
participates: $(1..*)$ as growsIn[1],
end

define entity *Location*
participates: $(1..*)$ as *growsIn*[2],
end

define entity *Fruit*
isa: *Produce*
end

define entity *Country*
isa: *Location*
end

define entity *Crop*
isa: *Produce*
end

Figure 2.1: Example of EER schema for the agricultural domain

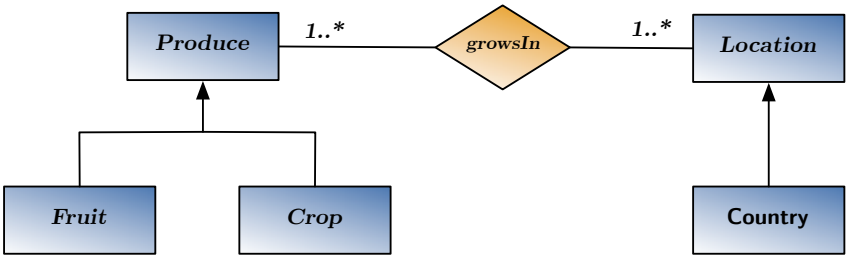


Figure 2.2: EER diagram

by diamond-shaped objects connected to the participating entities. The connectors between relationships and entities are labeled with the component of the relationship, as well as the cardinality constraints declared in the participates clause. An isa relation between two entities (or relationships) is denoted by an arrow from the more specific to the more general entity (relationship). For example, the diagram in Figure 2.2 corresponds to the EER schema in Figure 2.1.

EER Semantics

The semantics of EER schemata is defined by specifying when a database satisfies all constraints imposed by the schema, which is formalized by the notion of database instance. A *database instance* $\mathcal{B}$ corresponding to an EER schema is constituted by a nonempty *finite set* $\Delta^{\mathcal{B}}$ and a function $\cdot^{\mathcal{B}}$ that maps

- each entity $E \in \mathbf{E}$ to a subset $E^{\mathcal{B}}$ of $\Delta^{\mathcal{B}}$,
- every relationship $R \in \mathbf{R}$ to a set $R^{\mathcal{B}}$ of tuples over $\Delta^{\mathcal{B}}$.

The elements $E^{\mathcal{B}}, R^{\mathcal{B}}$ are called *instances* of E and R, respectively.

Definition 2.19. A database instance $\mathcal{B}$ is called *legal for a schema* $\mathcal{S}$ if it satisfies all integrity constraints specified in $\mathcal{S}$, that is,

1. for each pair of entities $E_1, E_2 \in \mathbf{E}$ such that E_1 isa E_2, it holds that $E_1^{\mathcal{B}} \subseteq E_2^{\mathcal{B}}$;
2. for each relationship $R \in \mathbf{R}$ among entities $E_1, \ldots, E_k$, $(e_1, \ldots, e_k) \in R^{\mathcal{B}}$ implies $e_i \in E_i^{\mathcal{B}}$ for every $i \in \{1, \ldots, k\}$; and
3. for each entity E participating in R as the $R[i]$ component with cardinality constraint $(m..n)$ it holds that
$$m \leq \#\{r \in R^{\mathcal{B}} \mid r[i] = e\} \leq n.$$

△

For example the following set of atoms is a database instance of the schema from Figure 2.2.

Produce(mango), Produce(durian), Produce(rice)
Fruit(mango, Pakistan), Fruit(durian, south-Asia),
Crop(rice),
Location(Pakistan), Location(south-Asia)

One of the basic properties to verity for a given schema $\mathcal{S}$ is the existence of a legal database instance. Further, for a more detailed analysis of the quality of $\mathcal{S}$ one may be interested in verifying whether the extension of a particular entity E from $\mathcal{S}$ is non empty in a legal database instance of $\mathcal{S}$.

It has been shown that EER schemata can be translated to $\mathcal{ALCQI}$ TBoxes, and that there is a correspondence between legal database instances and models of the obtained TBox [35]. Entities and domain symbols correspond to sets of objects (or unary relations) and can be modeled in DLs by concept names. On the other hand, for modeling relations it is necessary to resort to the so-called *reification* since relationships in EER schemata may have arbitrary arity and DLs only have binary relations (i.e, roles). Reifying an EER schema means that each relationship in the schema is modeled by a concept B_R (whose instances represent the tuples of the relationship) and for each component $R[i]$ of the relationship a (binary) role connects B_R with the concepts B_E used to model the entity E that participates as the component $R[i]$. Since EER schemata express only necessary conditions for objects of the domain to be instances of entities, they can be captured by means of concept inclusions in the TBox.

More precisely, for a given EER schema $\mathcal{S}$, the TBox $\mathcal{T}_{\mathcal{S}}$ is defined using the following vocabulary:

- for each entity symbol or relationship symbol X, there is a concept name B_X;
- for each relationship component Y there is a role P_Y.

Each constraint specified in $\mathcal{S}$ is modeled in $\mathcal{T}_{\mathcal{S}}$ using concept inclusions (see Table 2.4 as quick reference):

- for each pair of entities E_1, E_2 such that $E_1 \mathsf{isa} E_2$,
$$B_{E_1} \sqsubseteq B_{E_2};$$

EER construct	$\mathcal{ALCQI}$ **axioms**
Relationship R with entity E as the component $R[i]$	$B_R \sqsubseteq \exists P_{R[i]}.B_{E_i} \sqcap (\leqslant 1\ P_{R[i]}\ B_{E_i})$
isa between entities E_1 and E_2	$B_{E_1} \sqsubseteq B_{E_2}$
Cardinality constraint $(m..n)$ on E as $R[i]$	$B_E \sqsubseteq (\geqslant m\ P^-_{R[i]}\ B_R) \sqcap (\leqslant n\ P^-_{R[i]}\ B_R)$

Table 2.4: $\mathcal{ALCQI}$ axioms derived from an EER schema

- for each relationship R of arity k among entities $E_1, \ldots, E_k$,

$$B_R \quad \sqsubseteq \quad \textstyle\bigsqcap_{1 \leq i \leq k} \left(\exists P_{R[i]}.B_{E_i} \sqcap (\leqslant 1\ P_{R[i]}\ B_{E_i})\right)$$

- for each entity E participating in a relationship R as $R[i]$ with cardinality constraints $(m..n)$,

$$\begin{array}{ll} B_E \sqsubseteq (\geqslant m\ P^-_{R[i]}\ B_R) & \text{if } m \neq 0 \text{ and} \\ B_E \sqsubseteq (\leqslant n\ P^-_{R[i]}\ B_R) & \text{if } n \neq \infty; \end{array}$$

- for each pair $X_1, X_2 \in \mathbf{E} \cup \mathbf{R}$ such that $X_1 \neq X_2$ and $X_1 \in \mathbf{R}$,

$$B_{X_1} \sqsubseteq \neg B_{X_2}.$$

This translation makes use of inverse roles and number restrictions to capture the semantics of EER schemata. Notice that the resulting TBox $\mathcal{T}_\mathcal{S}$ may contain cyclic concept inclusions, even when the corresponding EER diagrams is acyclic.

Intuitively, the extension of a relationship R in a database instance is a set of tuples. Therefore, it is implicit in the semantics of the EER model that there cannot be two tuples connected through all roles of the relationship to exactly the same elements of the domain. Although the latter constraints cannot be explicitly stated in $\mathcal{ALCQI}$, the tree-model property guarantees that the translation provided by $\mathcal{T}_\mathcal{S}$ is faithful in the sense that there will be no two instances of the concept representing the same tuple of the relation [32, 35].

2.2.2 UML Class Diagrams

The *Unified Modeling Language (UML)* is the standard language for designing software and analyzing its structure. UML is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of a software system. The modeling language is intended to unify past experience about modeling techniques and to incorporate current software best practices into a standard approach. UML includes semantic concepts, notation, and guidelines. It has static, dynamic, environmental, and organizational parts. Moreover, it is intended to be supported by interactive visual modeling tools that have code generators and report writers. The objective of UML is to provide system architects, software engineers,

and software developers with tools for analysis, design, and implementation of software-based systems as well as for modeling business and similar processes.

From the conceptual modeling view point, UML *class diagrams* are the most important component of UML. Indeed, UML class diagrams are used for generating code skeleton and database schemata, as well as means for knowledge representation such as for specifying ontologies, and for defining meta-models of other programming, modeling, and specification languages. UML class diagrams model the domain of interest by visually describing classes of objects and relationships among them.

We will treat UML class diagrams as ontology languages [42, 84], and therefore, we will present them from the conceptual perspective. In particular, we will not deal with features that are more relevant for the software engineering perspective, such as operations (methods) associated to classes, or public, protected and private qualifiers for methods and attributes.

We will describe the constructors for UML class diagrams and provide their semantics. The semantics of UML class diagrams can be described in terms of (the two variable fragment of) FOL with *counting quantifiers* [60]. This formalization has been used extensively in the literature [21, 78]. Since DLs are fragments of FOL, the semantics of UML class diagrams relevant for this thesis can be captured as well using expressive description logics such as $\mathcal{ALCQI}$ [21, 35]. We will present simultaneously the various UML class diagrams constructs along with their DLs formalization.

A *class diagram* (CD) is a graphic representation of the overall structure of the domain of discourse that shows a collection of static model elements. The main constituents of this static view are *classes* and their relationships. Relationships among classes include *associations* and *generalizations*.

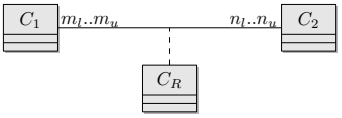
A *class* in UML class diagrams denotes a set of objects with common features. Visually, a class is represented by a rectangle with the specification of the class, which in in our case consists of only of its name. In terms of logic, a class C corresponds to a unary predicate C. Thus, in description logics, classes can be formalized by atomic concepts.

An *n-association* in UML class diagrams represents a relation between the instances of $n \geq 2$ classes. Usually, an association has a related *association class* (see Fig. 2.3b) that describes the properties of the association. Visually, an association class is connected to each class participating in the association. Each connection of an association to a class is called an *association end*. A binary association between two classes C_1 and C_2 is shown in Figure 2.3b.

Each association end can have a label indicating *cardinality constraints* on the participation of classes in associations. In Figure 2.3b the label $n_l..n_u$ specifies that each instance of the class C_1 participates at least n_l times and at most n_u times in the association R; in the same manner, the label $m_l..m_u$ specifies an analogous constraint for each instance of the class C_2. The label $0..*$, indicates that the participation of a class is unconstrained.

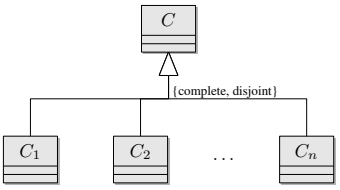
In UML a *generalization* relation between a *parent class* C_1 and a *child class* C_2 specifies that each instance of C_1 is also an instance of C_2. Hence, the instances of the child class inherit the properties of the parent class. Typically, a child class satisfies additional properties that in general do not hold for the parent class. In a class diagram, a generalization relation is

(a) UML class C



$$
\begin{array}{rl}
A & \sqsubseteq \exists r_1.C_1 \sqcap \exists r_2.C_2 \\
A & \sqsubseteq (\leqslant 1\ r_1) \sqcap (\leqslant 1\ r_2) \\
C_1 & \sqsubseteq (\geqslant n_\ell\ r_1^-\ A) \sqcap (\leqslant n_u\ r_1^-\ A) \\
C_2 & \sqsubseteq (\geqslant m_\ell\ r_2^-\ A) \sqcap (\leqslant m_u\ r_2^-\ A)
\end{array}
$$

(b) Binary association class



$$
\begin{array}{rll}
C_i & \sqsubseteq C & i \in \{1, \ldots, n\} \\
C_i & \sqsubseteq \neg C_j & i, j \in \{1, \ldots, n\} \text{ with } i \neq j \\
C & \sqsubseteq C_1 \sqcup \cdots \sqcup C_n &
\end{array}
$$

(c) A class hierarchy in UML

Figure 2.3: $\mathcal{ALCQI}$ formalization of UML class diagrams

indicated by an arrow from C_2 to C_1 (C_2 is a child of C_1). The formalization of the semantics of generalization in DLs can be done in a very natural way by concept inclusions

$$C_2 \sqsubseteq C_1.$$

Note that inclusion between DL concepts captures exactly the inheritance notion between UML classes. In fact, this is a straightforward consequence of the semantics of $\sqsubseteq$. As a consequence, in the formalization provided, each association involving C_1 is correctly inherited by C_2. Moreover, the formalization in DL also captures directly multiple inheritance between classes. In UML, several generalizations can be grouped together to form a class *hierarchy*, as shown in Fig. 2.3c. Such a hierarchy can be formalized by a set of concept inclusions:

$$C_i \sqsubseteq C \qquad \text{for each } i \in \{1, \ldots, n\}.$$

UML allows to specify constraints to describe, e.g., class identifiers, functional dependencies for associations. These constraints are usually expressed through the use of *OCL constraints* [102],

which amount to constraints expressible in FOL. In fact, due to their expressive power, OCL constraints capture the semantics of the standard UML class diagrams. A liberal use of OCL constraints can compromise the understandability of the diagram, and thus their use is typically limited. Finally, observe that unrestricted use of OCL constraints makes reasoning on class diagrams undecidable since it amounts to full reasoning in FOL.

In ontological knowledge representation, *disjointness* and *covering* constraints are the most commonly used in practice. In UML diagrams these constraints are expressed by labels 'disjoint', and 'complete' on generalization relationships. For example, in the diagram shown in Figure 2.3c, the constraint {complete, disjoint} expresses that every instance of C must be an instance of some of the class C_i, and that each pair of classes C_i, C_j do not have instances in common. To formalize these constraints in DLs we can use the following concept inclusions

$$C_i \sqsubseteq \neg C_j, \qquad \text{for each } i, j \in \{1, \ldots, n\} \text{ with } i \neq j \qquad \text{(disjoint)}$$

$$C \sqsubseteq C_1 \sqcup \cdots C_n \qquad \text{(complete)}$$

Naturally, it is also possible to impose constraints on hierarchies of association classes.

2.2.3 Class Diagrams with restricted expressivity

The formalization using description logics of EER schema and UML diagrams not only provides formal semantics, but also provides an unified view for both modeling languages [35]. For that reason, from now on, we will use the term *class diagrams (CDs)* to refer to UML class diagrams or EER diagrams. We will consider CDs with restricted expressivity along the following:

- isa relations between classes and relations,
- disjoint and complete constraints on generalization (isa) hierarchies, and
- cardinality constraints on relation participation.

We denote with $\mathcal{D}_{full}$ the CDs supporting all the constructs described above. Further, we will use $\mathcal{D}_{bool}$ to denote the language without generalization between relations, and finally, $\mathcal{D}_{ref}$ to denote the further constrained language that disallows completeness constraints over generalization hierarchies. We also consider a restriction of $\mathcal{D}_{full}$ diagrams in which 'complete' constraints are dropped as well as at-most constraints $[m..n]$ for $n > 2$ on relation participation; and we denote this class of diagrams as $\mathcal{D}^{-}_{Horn}$. These diagrams are able to support functionality constraints and *integrity constraints* given by Horn DLs axioms of the form $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq B$. Further, $\mathcal{D}^{-}_{ref}$ denotes the subclass of $\mathcal{D}_{ref}$ diagrams that disallow for at-most restrictions $[m..n]$ for $n > 1$ in relation participation, i.e., only constraints of the form $[m..*]$ and $[m..1]$ for $m \geq 0$ are supported. See Table 2.5, for reference.

2.2.4 Reasoning on Class Diagrams

Traditional database modeling tools are not designed to support the verification of certain properties of the schemata such as consistency or redundancy. Supporting these verification

Language	Constraints						
	Classes			Relations			
	isa	disjoint	complete	isa	card.	refinement	functionality
$\mathcal{D}_{full}$	✓	✓	✓	✓	✓	✓	✓
$\mathcal{D}_{bool}$	✓	✓	✓	✗	✓	✓	✓
$\mathcal{D}^{-}_{Horn}$	✓	✓	✓	✓	✗	✓	✓
$\mathcal{D}_{ref}$	✓	✓	✗	✗	✓	✓	✓
$\mathcal{D}^{-}_{ref}$	✓	✓	✗	✗	✗	✓	✓

Table 2.5: Class diagrams

tasks can be carried out by DL reasoners that take as input the translation of the schemata into DLs. An inconsistency in a conceptual model may be the result of a design error or due to over-constraining.

Recall that DLs are based on open-world semantics, which make them appropriate for conceptual modeling. Indeed, the open-world semantics is natural for a conceptual schema language since a schema is intended to determine the legal instantiations of a data oriented application (e.g., a legal database instance). In fact, computing the subsumption relation between concepts does not require to fix an instantiation; and thus the closed-world assumption is not appropriate for verifying subsumption. Instead, one must employ the open-world assumption to consider all possible instantiations.

From the discussion above, we can conclude that the formalization of the semantics of CDs using DLs has the advantage that one can use inferences in DLs for verifying properties of diagrams, in other words one can preform *reasoning on CDs*. *Reasoning tasks* of interest for CDs include the following:

- *subsumption* between two classes C_1, C_2, i.e., verify whether in every possible instantiation of a given diagram each instance of C_1 is also an instance of class C_2;
- *satisfiability* of a class C (or relation R) in a diagram, i.e., verify whether there exists an instantiation of the diagram where the set of instances of C is non-empty;
- *diagram satisfiability*, which amounts to verify that there is an instantiation of the diagram different from the empty one; and
- *full satisfiability* of a diagram, which corresponds to verify that there exists an instantiation of the diagram where the set of instances of *each* class and relation is non-empty.

Formally, both DLs and CDs are interpreted using first-order semantics, according to which one can distinguish between *legal* and *illegal* relational structures. A legal structure (or model) is one satisfying all constraints (or axioms), while an illegal one violates at least one constraint (or axiom). In DLs, the legal structures are called models, whereas for CDs they are called instantiations. Nonetheless, the underlying principle is the same.

The following known result relates reasoning in DLs and with reasoning on CDs.

Theorem 2.11 ([35]). *Let $\mathcal{C}$ be a conceptual schema, C_1, C_2 classes from $\mathcal{C}$, α a constraint expressed in FOL, and $\mathcal{T}_\mathcal{C}$ its DL formalization.*

- *$\mathcal{C}$ is satisfiable if $\mathcal{T}_\mathcal{C}$ is satisfiable;*
- *C_1 is satisfiable if B_{C_1} is satisfiable w.r.t. $\mathcal{T}_\mathcal{C}$;*
- *C_1 subsumes C_2 if $\mathcal{T}_\mathcal{C} \models B_{C_1} \sqsubseteq B_{C_2}$;*
- *α is a consequence of $\mathcal{C}$ if $\mathcal{T}_\mathcal{C} \models \alpha$.*

Chapter

3 Finite Model Reasoning in Horn DLs

One of the main lines of research in DLs has been dedicated to augment the expressivity of the description languages without compromising the computational complexity of reasoning. Moreover, some DL languages have been tailored for applications in Databases, where expressive data models have been developed for representing the data manipulated by commercial applications. There is indeed a strong similarity between the approaches in knowledge representation using description logics and conceptual modeling in databases [8, 34, 35]. In both scenarios the knowledge is represented in form of a schema by defining classes denoting subsets of the domain of interest, and by specifying various types of relations between classes which establish their structural properties.

While in DBs it is usually assumed that the underlying domain is finite, reasoning in DLs disregards such hypothesis. Although this seems to be in contrast with the purpose of representing real world structures, which are inherently finite, it can be justified by the fact that most DLs studied in the early days have the *finite model property (FMP)*. The FMP of a logic implies that it can always be assumed that the domain is finite when performing reasoning tasks. Still, the FMP does not hold for all DLs; this is the case for DLs (see Section 2.1.7) allowing for inverse roles and functionality of roles. Indeed, even rather unexpressive DL languages including these constructors lack the FMP.

Example 3. Consider the following *DL-Lite*$^{\mathcal{F}}_{core}$ TBox $\mathcal{T}$:

$$A_1 \sqsubseteq \exists p_1 \qquad \exists p_1^- \sqsubseteq B_1 \qquad B_1 \sqsubseteq \exists p_1$$
$$A_1 \sqcap \exists p_1^- \sqsubseteq \bot \qquad (\mathsf{funct}\ p_1^-)$$

Observe that in every model $\mathcal{I}$ of $\mathcal{T}$ satisfying A_1, there is an infinite sequence of objects $d_1, d_2, \ldots \in \Delta^{\mathcal{I}}$, with $d_1 \in A_1^{\mathcal{I}}$, $d_j \in B_1^{\mathcal{I}}$ for $j > 1$, and $(d_i, d_{i+1}) \in p_1^{\mathcal{I}}$, for $1 \leq i$. Note that $(\mathsf{funct}\ p_1^-)$ and $A_1 \sqcap \exists p_1^- \sqsubseteq \bot$ imply that $d_i \neq d_j$ for $i \neq j$. Thus, $\Delta^{\mathcal{I}}$ is an infinite set, and A_1 is not finitely satisfiable w.r.t. $\mathcal{T}$ (as it is only satisfiable by infinite models of $\mathcal{T}$). However,

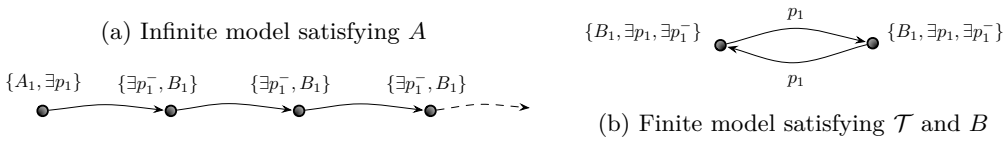


Figure 3.1: Models of $\mathcal{T}$ from Example 3

$\mathcal{T}$ is finitely satisfiable, and moreover $\mathcal{T} \models_{\mathsf{fin}} \exists p_1 \sqsubseteq \exists B_1$. See, e.g., the finite model of $\mathcal{T}$ in Figure 3.1b. $\overline{\wedge}$

Although the study of finite model reasoning in expressive description logics has been addressed in the literature [31, 94, 117], finite model reasoning is rarely used in practice. This is mainly because for many of the expressive DLs that lack the FMP, no algorithmic approaches to finite model reasoning that lend themselves towards efficient implementation are known. Indeed, this is the case for a wide spectrum of DLs: $\mathcal{ALCFI}, \mathcal{ALCQI}$, Horn-$\mathcal{ALCFI}$, and Horn-$\mathcal{ALCQI}$. While for reasoning w.r.t. unrestricted models there is a large amount of applicable algorithms such as *tableaux* and *resolution calculi*, which perform rather well in practical implementations, the approaches investigated so far for finite model reasoning in expressive DLs [31, 94] have provided only tools for showing that the complexity of finite model reasoning is the same as the complexity of unrestricted reasoning –ExpTime-complete. In particular, all known approaches rely on the construction of some systems of inequalities and them over the integers. The main difficulty for using those approaches in practice is that the system of inequalities is of exponential size in the best case, which falls far form obvious efficient implementations.

On the other end of the expressive power spectrum, the situation is quite different. For DLs of the *DL-Lite* family that lack the FMP, and particularly for *DL-Lite*$^{\mathcal{F}}_{core}$, Rosati [117] has shown that finite model reasoning can be reduced to unrestricted reasoning, and hence that the complexity of both problems coincides. The approach proposed by Rosati builds on a technique developed in database theory by Cosmadakis et al. [45] for deciding the finite implication of unary inclusion dependencies and functional dependencies. The virtue of Rosati's approach is that finite model reasoning does not require any new algorithmic technique and can be implemented as efficiently as unrestricted reasoning.

In this Chapter, we investigate (specific) reasoning methods for *finite model reasoning* in DLs lacking the FMP. We will extend Rosati's technique to larger fragments of Horn DLs. We start with a detailed explanation of the so-called cycle reversion technique in Section 3.1. Afterwards, we will show how cycle reversion is used for reducing finite model reasoning to unrestricted reasoning in *DL-Lite*$^{\mathcal{F}}_{Horn}$. In Section 3.3, we move on to the yet more expressive Horn-$\mathcal{ALCQI}$. As we will see, proving that for the Horn-$\mathcal{ALCQI}$ case cycle reversion captures completely finite model entailment turns out to be quite challenging.

Notation

We will introduce some notation that will be used along this chapter in order to ease the presentation. Let $\mathcal{T}$ be a TBox, and $\mathcal{A}$ an ABox. We will write

- $\mathsf{CN}(\mathcal{T})$ to denote the set of concept names occurring in $\mathcal{T}$;
- $\mathsf{BC}(\mathcal{T})$ to denote the set of *basic* concepts occurring in a *DL-Lite* TBox $\mathcal{T}$;
- $\mathsf{role}(\mathcal{T})$ to denote the set of role names and inverse roles occurring in $\mathcal{T}$; and
- $\mathsf{ind}(\mathcal{A})$ to denote the set of individual names occurring in $\mathcal{A}$.

3.1 Cycle Reversion

We start our investigation of finite model reasoning in Horn DLs with an overview of the cycle reversion technique. The main idea behind this technique stems from the axiomatization of so-called *finite implication unary inclusion dependencies (UINDs)* and *unary functional dependencies (UFDs)* presented in [45]. The essence of that axiomatization consists of detecting strong connected components of a directed graph representing UINDs and UFDs. As observed by Rosati [117], UINDs and UFDs can be seen as *DL-Lite*$_{core}^{\mathcal{F}}$ concepts of the form:

$$B_1 \sqsubseteq B_2 \tag{UIND}$$
$$(\mathsf{funct}\ r) \tag{UFD}$$

Recall that $\#S$ denotes the cardinality of a set S. The following is a simple observation implied by the DLs semantics.

Observation 1. Let $\mathcal{T}$ be a *DL-Lite*$_{core}^{\mathcal{F}}$ TBox,

1. if $\mathcal{T} \models B_1 \sqsubseteq B_2$, then for every model $\mathcal{I}$ of $\mathcal{T}$ $\#B_1^{\mathcal{I}} \leq \#B_2^{\mathcal{I}}$; and
2. if $\mathcal{T} \models (\mathsf{funct}\ r)$, then for every model $\mathcal{I}$ of $\mathcal{T}$ $\#(\exists r^-)^{\mathcal{I}} \leq \#(\exists r)^{\mathcal{I}}$.

We associate a directed graph $G = (V, E)$ to a given *DL-Lite*$_{core}^{\mathcal{F}}$ TBox $\mathcal{T}$, such that each node in V is labeled with a basic concept occurring in $\mathcal{T}$ and there is an edge $(B_1, B_2) \in E$ whenever $\#B_1^{\mathcal{I}} \leq \#B_2^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{T}$. We will make this more precise below, but first let us consider an example.

Example 4. For example, consider the following TBox $\mathcal{T}_1$:

$$\begin{array}{lll} \exists s \sqsubseteq \exists p_1, & \exists p_2^- \sqsubseteq \exists p_3^-, & (\mathsf{funct}\ p_1^-), \\ \exists p_1^- \sqsubseteq \exists p_2, & \exists p_3 \sqsubseteq \exists s^-, & (\mathsf{funct}\ p_2^-), \\ \exists p_2^- \sqsubseteq \exists p_1, & B \sqsubseteq \exists p_1, & (\mathsf{funct}\ s); \end{array}$$

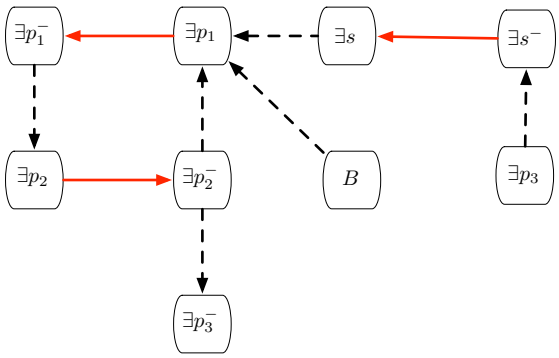


Figure 3.2: Graph G_1 corresponding to $\mathcal{T}_1$

the directed graph G_1 depicted in Figure 3.2 represents the '$\leq$' relations implied among the basic concepts in $\mathcal{T}_1$, according to Observation 1. The edges originated from Point 1 are drawn as dashed arrows, while the solid ones represent edges originated by Point 2. Using Observation 1 along the cycle $\exists p_1^-,\ \exists p_2,\ \exists p_2^-,\ \exists p_1,\ \exists p_1^-$ in G_1 it can be inferred that

$$\#(\exists p_1^-)^{\mathcal{I}} \ \leq\ \#(\exists p_2)^{\mathcal{I}} \ \leq\ \#(\exists p_2^-)^{\mathcal{I}} \ \leq\ \#(\exists p_1)^{\mathcal{I}} \ \leq\ \#(\exists p_1^-)^{\mathcal{I}}$$

That means that the cardinality of the extensions of all the concepts in the cycle is the same in every model of $\mathcal{T}_1$. Further, let us consider a finite model $\mathcal{I}$ of $\mathcal{T}_1$. We have that $(\exists p_1^-)^{\mathcal{I}} \subseteq (\exists p_2)^{\mathcal{I}}$ since $\exists p_1^- \sqsubseteq \exists p_2 \in \mathcal{T}_1$, now, from $\#(\exists p_1^-)^{\mathcal{I}} = \#(\exists p_2)^{\mathcal{I}}$ and $\mathcal{I}$ being finite we can conclude that $(\exists p_2)^{\mathcal{I}} \subseteq (\exists p_1^-)^{\mathcal{I}}$, which means that $\exists p_2 \sqsubseteq \exists p_1^-$ is finitely entailed by $\mathcal{T}_1$. Analogously, from $(\mathsf{funct}\ p_1^-) \in \mathcal{T}_1$ and $\#(\exists p_1)^{\mathcal{I}} = \#(\exists p_1^-)^{\mathcal{I}}$ we can conclude that $(p_1^-)^{\mathcal{I}}$ is a bijection and hence, its inverse $(p_1)^{\mathcal{I}}$ is also a function, which means that $(\mathsf{funct}\ p_1)$ if finitely entailed by $\mathcal{T}_1$. $\overline{\wedge}$

The example above suggests a way to detect concept inclusions holding in finite models of a given TBox $\mathcal{T}$. Indeed, if we analyze the cycles in the directed graph G related to $\mathcal{T}$, the axioms that the result from 'reversing' the edges in a cycle are entailed in finite models. We next formalize all the relevant notions to describe the cycle reversion procedure.

Definition 3.1. For a given *DL-Lite*$_{core}^{\mathcal{F}}$ TBox $\mathcal{T}$, we define the directed graph $G = (V, E)$ as follows: V is the set of basic concepts in $\mathcal{T}$ and

1. if $B_1 \sqsubseteq B_2 \in \mathcal{T}$, then $(B_1, B_2) \in E$;
2. if $(\mathsf{funct}\ r) \in \mathcal{T}$, then $(\exists r^-, \exists r) \in E$.

The *completion* $\mathsf{finClosure}(\mathcal{T})$ of $\mathcal{T}$ is the TBox obtained by applying the following rule:

(cycle-reversion) if $B_1, B_2, \ldots, B_n = B_1$ is a cycle in G, then extend $\mathcal{T}$ with the following concept inclusions, for every $1 \leq i < n$

- $B_{i+1} \sqsubseteq B_i$ if (B_i, B_{i+1}) was added by Point 1; and
- (funct r^-), if (B_i, B_{i+1}) was added by Point 2.

△

Going back to the TBox $\mathcal{T}_1$ from Example 4, consider the (only) cycle:

$$\exists p_1^-,\ \exists p_2,\ \exists p_2^-,\ \exists p_1,\ \exists p_1^-$$

which corresponds to the axioms:

$$\exists p_1^- \sqsubseteq \exists p_2, \qquad (\mathsf{funct}\ p_1^-), \qquad \exists p_2^- \sqsubseteq \exists p_1, \quad \text{and} \quad (\mathsf{funct}\ p_2^-). \tag{3.1}$$

Then, applying **cycle-revertion** results in adding the following axioms to $\mathsf{finClosure}(\mathcal{T}_1)$:

$$\exists p_2 \sqsubseteq \exists p_1^-, \qquad (\mathsf{funct}\ p_1), \qquad \exists p_1 \sqsubseteq \exists p_2^-, \quad \text{and} \quad (\mathsf{funct}\ p_2). \tag{3.2}$$

From Definition 3.1, it follows that to compute the $\mathsf{finClosure}(\mathcal{T})$ of a TBox $\mathcal{T}$ it is enough to analyze the cycles in its corresponding graph G. We then obtain the following result.

Lemma 3.1. *Let $\mathcal{T}$ be a DL-Lite$_{core}^{\mathcal{F}}$ TBox. $\mathsf{finClosure}(\mathcal{T})$ can be computed in polynomial time on the size of $\mathcal{T}$.*

It has been shown that the completion of a *DL-Lite*$_{core}^{\mathcal{F}}$ TBox $\mathcal{T}$ using cycle-reversion provides an axiomatization of the finite model entailments of $\mathcal{T}$ [117]. We *reproof* this result here for didactic purposes. We start by showing that cycle-reversion is sound w.r.t. finite model entailments.

Lemma 3.2. *Let $\mathcal{T}$ be a DL-Lite$_{core}^{\mathcal{F}}$-TBox. We obtain then the following.*

- *if* $\mathsf{finClosure}(\mathcal{T}) \models B_1 \sqsubseteq B_2$ *then* $\mathcal{T} \models_{\mathsf{fin}} B_1 \sqsubseteq B_2$*; and*
- *if* $\mathsf{finClosure}(\mathcal{T}) \models (\mathsf{funct}\ r)$ *then* $\mathcal{T} \models_{\mathsf{fin}} (\mathsf{funct}\ r)$.

Proof. It is enough to show that if $B_1, \ldots, B_n$ is a cycle in G, then the axioms added by reversing this cycle are entailed in every finite model $\mathcal{I}$ of $\mathcal{T}$. Thus, let $\mathcal{I}$ be a finite model of $\mathcal{T}$, we show that for every $1 \leq i < n$:

(i) if $B_{i+1} \sqsubseteq B_i$ was added by cycle-reversion, then $B_{i+1}^{\mathcal{I}} \subseteq B_i^{\mathcal{I}}$.

(ii) if (funct r^-) was added by cycle-reversion, then $\mathcal{I} \models (\mathsf{funct}\ r^-)$.

Observe that the semantics of *DL-Lite*$^{\mathcal{F}}_{core}$ implies that $\#B_1^{\mathcal{I}} \leq \#B_2^{\mathcal{I}} \cdots \leq \#B_n^{\mathcal{I}}$. Since $B_1 = B_n$, we can conclude

$$\#B_1^{\mathcal{I}} = \#B_2^{\mathcal{I}} \cdots = \#B_n^{\mathcal{I}} \qquad (*)$$

Let us fix some $1 \leq i < n$; we treat each of the cases above:

- **Case (i):** By Definition 3.1, $(B_i, B_{i+1}) \in E$ was added because $B_i \sqsubseteq B_{i+1} \in \mathcal{T}$. Since $\mathcal{I}$ is a model of $\mathcal{T}$, we have $B_i^{\mathcal{I}} \subseteq B_{i+1}^{\mathcal{I}}$. Further, $(*)$ gives that $\#B_i^{\mathcal{I}} = \#B_{i+1}^{\mathcal{I}}$, and then since $\mathcal{I}$ is finite we can conclude that $B_{i+1}^{\mathcal{I}} \subseteq B_i^{\mathcal{I}}$, as required.
- **Case (ii):** By definition 3.1, $(B_i, B_{i+1}) \in E$ was added because $(\mathsf{funct}\ r) \in \mathcal{T}$. Furthermore, we have that $B_i = \exists r^-$ and $B_{i+1} = \exists r$; $(*)$ yields that $\#(\exists r^-)^{\mathcal{I}} = \#(\exists r)^{\mathcal{I}}$. This means that $r^{\mathcal{I}}$ is bijection, hence $(r^-)^{\mathcal{I}}$ is a function, and thus $\mathcal{I} \models (\mathsf{funct}\ r^-)$ as required.

□

We now show that the implications in the 'other direction' from Lemma 3.2 also hold, that is, $\mathsf{finClosure}(\mathcal{T})$ is complete for finite model entailment w.r.t. $\mathcal{T}$. In order to do so, we need to introduce first some useful notions. Recall that for any two concepts B_1, B_2, we write $B_1 \sqsubseteq_{\mathcal{T}} B_2$ whenever $\mathcal{T} \models B_1 \sqsubseteq B_2$. To ease notation, for a given TBox $\mathcal{T}$ we sometimes will use $\mathcal{T}_{\mathsf{f}}$ to denote $\mathsf{finClosure}(\mathcal{T})$.

Definition 3.2. Let $\mathcal{T}$ be a *DL-Lite*$^{\mathcal{F}}_{core}$ TBox and B_i a basic concept, the *lite-type of* B_i *w.r.t.* $\mathcal{T}_{\mathsf{f}}$ is the set of basic concepts

$$[B_i] = \{B_j \in \mathsf{BC}(\mathcal{T}_{\mathsf{f}}) \mid B_i \sqsubseteq_{\mathcal{T}_{\mathsf{f}}} B_j\}.$$

We denote with $\mathsf{lite\text{-}TP}(\mathcal{T}_{\mathsf{f}})$ the set of *lite-types w.r.t.* $\mathcal{T}_{\mathsf{f}}$. Further, we write

- $[B_i] \to_r [B_j]$ iff $B_i \sqsubseteq_{\mathcal{T}_{\mathsf{f}}} \exists r$ and $[B_j] = [\exists r^-]$.
- $[B_i] \to^1_r [B_j]$ iff $[B_i] \to_r [B_j]$ and $(\mathsf{funct}\ r^-) \in \mathcal{T}_{\mathsf{f}}$.
- $[B_i]\ {}^1\!\leftrightarrow^1_r [B_j]$ iff $[B_i] \to^1_r [B_j]$ and $[B_j] \to^1_{r^-} [B_i]$.

We use ${}^1\!\leftrightarrow^1_*$ to denote the transitive closure of the relation $\bigcup_{r \in \mathsf{role}(\mathcal{T}_{\mathsf{f}})} {}^1\!\leftrightarrow^1_r$. For a lite-type $[B_i]$, the *cluster* of $[B_i]$ is the set $\mathfrak{C}([B_i]) := \{[B_j] \in \mathsf{lite\text{-}TP}(\mathcal{T}_{\mathsf{f}}) \mid [B_i]\ {}^1\!\leftrightarrow^1_*\ [B_j]\}$. △

The intuition behind considering the lite-types $[B_i]$ for each basic concept B_i is that we can simplify the graph G'_1 corresponding to $\mathcal{T}_{\mathsf{f}}$ by adding one node for each $[B_i]$, which will aid to have more compact finite models. For example, consider the graph corresponding to $\mathsf{finClosure}(\mathcal{T}_1)$ depicted in Figure 3.3. Since $\exists p_1^- \sqsubseteq \exists p_2$, $\exists p_2 \sqsubseteq \exists p_1^- \in \mathsf{finClosure}(\mathcal{T}_1)$, we have that $[\exists p_1^-] = [\exists p_2]$ Furthermore, as $(\mathsf{funct}\ p_2)$, $(\mathsf{funct}\ p_2^-) \in \mathsf{finClosure}(\mathcal{T}_1)$ (hence the solid edges $([\exists p_1], [\exists p_2]), ([\exists p_2], [\exists p_1])$ are in G'_1) we have also that $[\exists p_1^-]\ {}^1\!\leftrightarrow^1_{p_2}\ [\exists p_1]$.

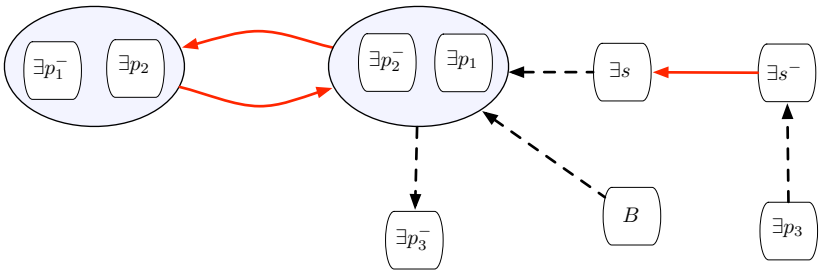


Figure 3.3: Graph of $\mathsf{finClosure}(\mathcal{T}_1)$

For our completeness proof, we will construct an interpretation $\mathcal{I}^{\mathsf{f}}$ whose domain contains at least one object for each satisfiable concept B_i w.r.t. $\mathcal{T}_{\mathsf{f}}$. Thus, the domain of $\mathcal{I}^{\mathsf{f}}$ will consist of finite words from Σ^*, with the alphabet $\Sigma := \mathsf{lite\text{-}TP}(\mathcal{T}_{\mathsf{f}})$. For a word $\sigma = [B_1] \cdots [B_n]$, we write $\mathsf{tail}(\sigma)$ to denote $[B_n]$, and write ε to denote the empty word; by definition $\mathsf{tail}(\varepsilon) = [\bot]$. We construct $\mathcal{I}^{\mathsf{f}}$ inductively. We start by adding to the domain of $\mathcal{I}^{\mathsf{f}}$ exactly one object for each element in Σ. More specifically, we set

$$\Delta_0 := \{[B_i] \mid B_i \in \mathsf{BC}(\mathcal{T}_{\mathsf{f}}), B_i \not\sqsubseteq_{\mathcal{T}_{\mathsf{f}}} \bot\}.$$

Assuming that Δ_k is already defined. For all elements $(\sigma \cdot [B_i])$ that are new in Δ_k, we do the following. For each class $[B_j]$ that is reachable along the relation ${}^1{\leftrightarrow}^1_*$ from $[B_i]$, we add a fresh domain element. More precisely, we add the following set of elements:

$$\Gamma_k = \{(\sigma \cdot [B_j]) \mid (\sigma \cdot [B_i]) \in \Delta_k, [B_j] \in \mathfrak{C}([B_i]), \mathsf{tail}(\sigma) \not\sqsubseteq_{\mathcal{T}_{\mathsf{f}}} [B_j]\}.$$

Finally, for the inductive step from Δ_k to Δ_{k+1} we add objects along the relation

$$\bigcup_{r \in \mathsf{role}(\mathcal{T}_{\mathsf{f}})} \rightarrow^1_r .$$

For all domain objects σ that are new in $\Delta_k \cup \Gamma_k$, we add a fresh object $(\sigma \cdot [B_j])$ whenever $\mathsf{tail}(\sigma) \rightarrow^1_r [B_j]$ for some $r \in \mathsf{role}(\mathcal{T}_{\mathsf{f}})$. We assume in this case that $[B_j] \notin \mathfrak{C}(\mathsf{tail}(\sigma))$, otherwise, there is already an object in Γ_k that can be used as an r-successor for σ.

$$\begin{aligned}\Delta_{k+1} :=& \{(\sigma \cdot [B_j]) \mid \sigma \in \Delta_k \cup \Gamma_k, \text{ s.t. } \mathsf{tail}(\sigma) \rightarrow^1_r [B_j], \text{ for } r \in \mathsf{role}(\mathcal{T}_{\mathsf{f}})\} \\ & \cup \Delta_k \cup \Gamma_k.\end{aligned}$$

The step of the construction from Δ_k to Γ_k ensures that there is an equal number of objects from each lite-type belonging to a cluster. On the other hand, in the step from Δ_k to Δ_{k+1} we add fresh r-successors for objects $\sigma \cdot [B_i]$ whenever we have that $[B_i] \rightarrow^1_r [B_j]$. Note that since $(\mathsf{funct}\ r^-) \in \mathcal{T}_{\mathsf{f}}$, no object already present in Δ_k can be reused as r-successor for σ without potentially violating the functionality of r^-.

Proposition 3.3. *The above construction eventually reaches a fixed-point, i.e., there is an N such that $\Delta_N = \Delta_{N+1}$. Moreover, $N \leq \#\mathsf{role}(\mathcal{T}_\mathsf{f}) + 1$.*

Proof. We first note that if an element $(\sigma \cdot [B_j])$ is added in the second part of the inductive step (i.e., $(\sigma \cdot [B_j]) \notin \Gamma_k$) then $\mathsf{tail}(\sigma) \rightarrow^1_r [B_j]$ for some role $r \in \mathsf{role}(\mathcal{T}_\mathsf{f})$; and $\mathsf{tail}(\sigma)\ {}^1{\leftrightarrow}^1_* [B_j]$ does *not* hold. Further, by construction for every $\sigma = [B_1] \cdots [B_k]$, we have that $[B_1] \rightarrow^1_{r_1} [B_2] \dots [B_{k-1}] \rightarrow^1_{r_k} [B_k]$. Moreover, it can be shown by induction on the structure of σ that:

1. $r_i \neq r_{i+1}$ for every $i \in \{1, \dots, k-1\}$; and
2. $r_{i+i} \neq r_i^-$ for every $i \in \{1, \dots, k-1\}$.

Hence, the length of every σ is bounded by $N = \#\mathsf{role}(\mathcal{T}_\mathsf{f})+1$; and since the number of equivalence lite-types for $\mathcal{T}_\mathsf{f}$ is bounded by the cardinality of $\mathsf{BC}(\mathcal{T})$, only a finite number of words can be produced in the construction. □

Let $\Delta^{\mathcal{I}^\mathsf{f}} = \Delta^N$. Now, we fix the interpretation of the atomic concept and roles. The interpretation for atomic concepts is straightforward, for each atomic concept A, we set

$$A^{\mathcal{I}^\mathsf{f}} = \{\sigma \in \Delta^{\mathcal{I}^\mathsf{f}} \mid \mathsf{tail}(\sigma) \sqsubseteq_{\mathcal{T}_\mathsf{f}} A\}.$$

We define the interpretation of roles along the inductive steps for constructing $\Delta^{\mathcal{I}^\mathsf{f}}$. For each atomic role $p \in \mathsf{role}(\mathcal{T}_\mathsf{f})$, we define:

$$p^{\mathcal{I}^\mathsf{f}_0} = \emptyset \tag{3.3}$$

$$p^{\mathcal{I}^\mathsf{f}_{k+1}} = p^{\mathcal{I}^\mathsf{f}_k} \tag{3.4}$$

$$\cup\, \{((\sigma \cdot [B_i]), (\sigma \cdot [B_j])) \in (\Delta_k \times \Delta_k) \mid [B_i]\ {}^1{\leftrightarrow}^1_p [B_j]\} \tag{3.5}$$

$$\cup\, \{((\sigma \cdot [B_i]), (\sigma \cdot [B_i][B_j])) \in (\Delta_k \times \Delta_{k+1}) \mid [B_i] \rightarrow_p [B_j]\} \tag{3.6}$$

$$\cup\, \{((\sigma \cdot [B_i][B_j]), (\sigma \cdot [B_i])) \in (\Delta_{k+1} \times \Delta_k) \mid [B_i] \rightarrow_{p^-} [B_j]\} \tag{3.7}$$

$$\cup\, \{((\sigma \cdot [B_i]), [\exists p^-]) \in (\Delta_k \times \Delta_0) \mid B_i \sqsubseteq_{\mathcal{T}_\mathsf{f}} \exists p, (\mathsf{funct}\ p^-) \notin \mathcal{T}_\mathsf{f}\} \tag{3.8}$$

$$\cup\, \{([\exists p], (\sigma \cdot [B_i]) \in (\Delta_0 \times \Delta_k) \mid B_i \sqsubseteq_{\mathcal{T}_\mathsf{f}} \exists p^-, (\mathsf{funct}\ p) \notin \mathcal{T}_\mathsf{f}\}. \tag{3.9}$$

At the beginning of the construction the model is disconnected (3.3). Once $p^{\mathcal{I}^\mathsf{f}_k}$ is defined (3.4), we add all pairs that result by connecting the objects added in $\Delta_k \cup \Gamma_k$ (3.5). Further, we add all the pairs that result by connecting the objects created in the step from Δ_k to Δ_{k+1} (3.6 and 3.7). Finally, we connect all the objects σ and $[B_j]$ such that $\mathsf{tail}(\sigma) \rightarrow_r [B_j]$ and $(\mathsf{funct}\ r^-) \notin \mathcal{T}_\mathsf{f}$ (3.8) and (3.9). Note that in this case no fresh element is created in the step from Δ_k to Δ_{k+1}, but we can use the object $[\exists r^-]$ already present in Δ_0.

Lemma 3.4. *$\mathcal{I}^\mathsf{f}$ is a finite model of $\mathcal{T}_\mathsf{f}$.*

Proof of Lemma 3.4. We proceed by analyzing the various kinds of statements in $\mathcal{T}_\mathsf{f}$. Let $(\sigma \cdot [B_i]) \in \Delta_k, k < N$, we distinguish the following cases:

- *case* $B \sqsubseteq \bot$*:* It follows from the construction. Indeed, $\mathcal{T}_\mathsf{f} \models B \sqsubseteq \bot$, implies that there is no $\sigma \in \Delta_0$ with $\sigma \in B^{\mathcal{I}^\mathsf{f}}$. Furthermore, no instance of B is introduced by the construction in any Δ_i with $i > 0$. Indeed, assume towards a contradiction that this is the case. Then, there is some $\sigma' \in \Delta_j$ with $\mathsf{tail}(\sigma') \rightarrow_r [B]$. This means that $\mathcal{T}_\mathsf{f} \models \mathsf{tail}(\sigma) \sqsubseteq \bot$, which contradicts the fact that $\sigma' \in \Delta^{\mathcal{I}^\mathsf{f}}$.

- *case* $B \sqsubseteq A$*:* If $\sigma \in B^{\mathcal{I}^\mathsf{f}}$, then $A \in \mathsf{tail}(\sigma)$, then the construction ensures that $\sigma \in A^{\mathcal{I}^\mathsf{f}}$.

- *case* $B \sqsubseteq \exists r$. If $\sigma \cdot [B_i] \in C^{\mathcal{I}^\mathsf{f}}$, then since $C \sqsubseteq_{\mathcal{T}_\mathsf{f}} \exists r$, $[B_i] \rightarrow_r [B_j]$, and $\exists r^- \in [B_i]$. By construction, there is an object $\sigma' \in \Delta_{k+1}$ such that $((\sigma \cdot [B_i]), \sigma') \in r^{\mathcal{I}^\mathsf{f}_{k+1}}$. Indeed, if $[B_i] \rightarrow^1_r [B_j]$ then either (i) $\sigma' = (\sigma \cdot [B_j])$ was introduced in Γ_k and $(\sigma, \sigma' \cdot [B_i]) \in r^{\mathcal{I}^\mathsf{f}}$ by (3.5); or (ii) an object $\sigma' = (\sigma \cdot [B_i] \cdot [B_j])$ was added in the step from Δ_k to Δ_{k+1} and $(\sigma, \sigma \cdot [B_i]) \in r^{\mathcal{I}^\mathsf{f}}$ by (3.6) or (3.7). On the other hand, if $(\mathsf{funct}\ r^-) \notin \mathcal{T}_\mathsf{f}$, $\sigma' = [\exists r^-]$, and $(\sigma, [\exists r-]) \in r^{\mathcal{I}^\mathsf{f}}$ by (3.8) or (3.9). Therefore, $(\sigma \cdot [B_i]) \in (\exists r)^{\mathcal{I}^\mathsf{f}}$ as desired.

- *case* $(\mathsf{funct}\ r)$. We observe that in the construction, whenever we have an object $\sigma \in \Delta_k$ such that $\mathsf{tail}(\sigma) \sqsubseteq_{\mathcal{T}_\mathsf{f}} \exists r$, and there is no object $\sigma' \in \Delta^{\mathcal{I}^\mathsf{f}_k}$ with $(\sigma, \sigma') \in r^{\mathcal{I}^\mathsf{f}_k}$ then *exactly one pair* (σ, σ') is added to $r^{\mathcal{I}^\mathsf{f}}$. Indeed, if $(\mathsf{funct}\ r^-) \in \mathcal{T}_\mathsf{f}$, then σ' is created in the step from Δ_k to Δ_{k+1}: either $[\mathsf{tail}(\sigma)]\ {}^1\!\leftrightarrow^1_r [\exists r^-]$, and then $\sigma' \in \Gamma_k$; or $\mathsf{tail}(\sigma) \rightarrow^1_r [B_i]$ and then $\sigma' = (\sigma \cdot [\exists r])$. Otherwise, if $(\mathsf{funct}\ r^-) \notin \mathcal{T}_\mathsf{f}$, then $\sigma' = [\exists r^-] \in \Delta_0$.

□

Moreover, the construction of $\mathcal{I}^\mathsf{f}$ ensures the following.

Lemma 3.5. *Let B_1, B_2 be basic concepts. If $\mathcal{T}_\mathsf{f} \not\models B_1 \sqsubseteq B_2$, then there is $d \in \Delta^{\mathcal{I}^\mathsf{f}}$ such that $d \in B_1^{\mathcal{I}^\mathsf{f}}$ and $d \notin B_2^{\mathcal{I}_\mathsf{f}}$.*

Proof. We can assume w.l.o.g. that that B_1 is satisfiable, as otherwise the statement holds trivially. Assume $\mathcal{T}_\mathsf{f} \not\models B_1 \sqsubseteq B_2$. By construction, there is an element $d \in \Delta^{\mathcal{I}^\mathsf{f}}$ realizing the lite-type $[B_1]$. Further, $d \in B_1^{\mathcal{I}^\mathsf{f}}$ and since $\mathcal{T}_\mathsf{f} \not\models B_1 \sqsubseteq B_2$, then $B_2 \notin [B_1]$ and hence $d \notin B_2^{\mathcal{I}^\mathsf{f}}$. □

We are now ready to prove that cycle-reversion is complete w.r.t. finite entailments.

Theorem 3.6 (completeness)**.** *Let $\mathcal{T}$ be a DL-Lite$^\mathcal{F}_{core}$ TBox, if $\mathcal{T} \models_\mathsf{fin} B_1 \sqsubseteq B_2$ then $\mathcal{T}_\mathsf{f} \models B_1 \sqsubseteq B_2$.*

Proof. We prove the contraposition: $\mathcal{T}_{\mathsf{f}} \not\models B_1 \sqsubseteq B_2$ implies $\mathcal{T} \not\models_{\mathsf{fin}} B_1 \sqsubseteq B_2$. If $\mathcal{T}_{\mathsf{f}} \not\models B_1 \sqsubseteq B_2$, then by Lemma 3.5 there is some $e \in \Delta^{\mathcal{I}^{\mathsf{f}}}$ with $d \in B_1^{\mathcal{I}^{\mathsf{f}}}$ and $d \notin B_2^{\mathcal{I}^{\mathsf{f}}}$. Since $\mathcal{T} \subseteq \mathcal{T}_{\mathsf{f}}$, Lemma 3.4 yields $\mathcal{I}^{\mathsf{f}} \models \mathcal{T}$. Finally since $\mathcal{I}^{\mathsf{f}}$ is finite by Proposition 3.3 we can conclude $\mathcal{T} \not\models_{\mathsf{fin}} B_1 \sqsubseteq B_2$. □

3.2 Cycle reversion in *DL-Lite*$^{\mathcal{F}}_{Horn}$

In this section, we consider finite model reasoning in *DL-Lite*$^{\mathcal{F}}_{Horn}$. More precisely, we discuss how the cycle-reversion from the previous section can be almost straightforwardly extended to the case of *DL-Lite*$^{\mathcal{F}}_{Horn}$.

In the case of *DL-Lite*$^{\mathcal{F}}_{Horn}$, we will be dealing with (arbitrary) conjunctions of basic concepts instead of just basic concepts. Thus, an attempt to extend the previous notion of cycle, which only involved basic concepts to (arbitrary) conjunctions of basic concepts, is to take care of this fact. We first extend the relation $\to_r$ to conjunctions of basic concepts.

Definition 3.3. Let $\mathcal{T}$ be a *DL-Lite*$^{\mathcal{F}}_{Horn}$ TBox. For a pair of conjunctions $K_1 = \bigsqcap_i B_i$ and $K_2 = \bigsqcap_j B'_j$ of basic concepts and r a (possibly inverse) role, we write

- $K_1 \to_r K_2$ if $\mathcal{T} \models K_1 \sqsubseteq \exists r$, and $\mathcal{T} \models \exists r^- \sqsubseteq K_2$.
- $K_1 \to^1_r K_2$ if $K_1 \to_r K_2$, and $\mathcal{T} \models (\mathsf{funct}\ r^-)$.

△

We use now these relations to define a suitable notion of finmod cycle.

Definition 3.4 (finmod cycle)**.** Let $\mathcal{T}$ be a *DL-Lite*$^{\mathcal{F}}_{Horn}$ TBox. A *finmod-cycle* in $\mathcal{T}$ is a sequence

$$K_1, r_1, K_2, r_2, \ldots, r_{n-1}, K_n$$

where $K_1, \ldots, K_n$ are conjunctions of basic concepts from $\mathcal{T}$ and $r_1, \ldots, r_{n-1} \in \mathsf{roles}(\mathcal{T})$ such that $K_n = K_1$, and for every $i \in \{1, \ldots, n-1\}$:

$$K_i \to^1_{r_i} K_{i+1}.$$

△

Observe that according to the previous definition, we can have various cycles in a TBox that are related. To illustrate this we consider the following example.

Example 5. Consider the the following *DL-Lite*$^{\mathcal{F}}_{Horn}$TBox $\mathcal{T}_2$:

$$A_1 \sqcap A_2 \sqsubseteq \exists r_1 \qquad \exists r_1^- \sqsubseteq B_1 \sqcap B_2 \sqcap B_3 \sqcap B_4 \qquad (\mathsf{funct}\ r_1^-)$$
$$B_1 \sqcap B_2 \sqsubseteq \exists r_2 \qquad \exists r_2^- \sqsubseteq A_1 \sqcap A_2 \qquad (\mathsf{funct}\ r_2^-)$$

$\mathcal{T}_1$ contains the finmod-cycles:

$$\{A_1, A_2\}, r_1, \{B_1, B_2\}, r_2, \{A_1, A_2\} \qquad (\mathcal{C}_1)$$
$$\{A_1, A_2\}, r_1, \{B_1, B_2, B_3\}, r_2, \{A_1, A_2\} \qquad (\mathcal{C}_2)$$
$$\{A_1, A_2\}, r_1, \{B_1, B_2, B_3, B_4\}, r_2, \{A_1, A_2\} \qquad (\mathcal{C}_3)$$

Following the same intuition as for the *core* case, we could conclude, for example, that in every finite model $\mathcal{I}$ of $\mathcal{T}_2$ the presence of the cycle $(\mathcal{C}_1)$ implies that $\#(A_1 \sqcap A_2)^{\mathcal{I}} = \#(B_1 \sqcap B_2)^{\mathcal{I}}$. But then, by applying the same reasoning to cycles $(\mathcal{C}_2)$ and $(\mathcal{C}_3)$, we would also be able to conclude that

$$\#(B_1 \sqcap B_2)^{\mathcal{I}} = \#(B_1 \sqcap B_2 \sqcap B_3)^{\mathcal{I}} = \#(B_1 \sqcap B_2 \sqcap B_3 \sqcap B_4)^{\mathcal{I}}.$$

Moreover, using a similar analysis one can conclude that $(B_1 \sqcap B_2)^{\mathcal{I}} \subseteq (\exists r_1^-)^{\mathcal{I}}$, and that $(B_1 \sqcap B_2)^{\mathcal{I}} \subseteq (B_3 \sqcap B_4)^{\mathcal{I}}$. Then, the question is, what are the axioms that are needed in $\mathcal{T}_2$ to capture those consequences? $\overline{\wedge}$

The fact that various finmod-cycles are related to each other as in the example above, can be explained by observing that in *DL-Lite*$^{\mathcal{F}}_{Horn}$ as well as in *DL-Lite*$^{\mathcal{F}}_{core}$ one cannot distinguish among the r-successors of a given object in a model of a TBox. Essentially, for any (finite or unrestricted) model $\mathcal{I}$ of a given TBox $\mathcal{T}$, the r-successor of an object $d \in \Delta^{\mathcal{I}}$ has lite-type $[\exists r^-]$ (w.r.t. $\mathcal{T}$). The latter observation, gives us a hint that in order to reverse *all* the 'relevant cycles' it suffices to reverse finmod-cycles of the form:

$$[\exists r_1^-], r_2, [\exists r_2^-], r_3 \ldots, r_1, [\exists r_1^-].$$

Definition 3.5 (cycle reversion rule)**.** Let $\mathcal{T}$ be a *DL-Lite*$^{\mathcal{F}}_{Horn}$ TBox. We say that an finmod-cycle is simple if it has the form:

$$[\exists r_1^-], r_2, [\exists r_2^-], r_3 \ldots, r_n, [\exists r_1^-], \quad \text{s.t. } r_n = r_1.$$

The *finite model closure* $\mathsf{finClosure}(\mathcal{T})$ of $\mathcal{T}$ is obtained from $\mathcal{T}$ by exhaustively applying the following rule:

(simple-cycle-reversion) For every simple finmod-cycle

$$[\exists r_1^-], r_2, [\exists r_2^-], r_3 \ldots, r_n, [\exists r_1^-]$$

in $\mathcal{T}$, extend $\mathcal{T}$ with the following axioms, for $1 \leq j < n$:

$$\exists r_{j+1} \sqsubseteq \exists r_j^- \quad \text{and} \quad (\mathsf{funct}\ r_j) \tag{3.10}$$

$\triangle$

According to the previous definition, the TBox $\mathcal{T}_2$ contains the simple finmod-cycle:

$$[\exists r_1^-], r_2, [\exists r_2^-], r_1, [\exists r_1^-]$$

and the application of simple-cycle reversion adds the axioms:

$$\exists r_2 \sqsubseteq \exists r_1^-, \qquad (\mathsf{funct}\ r_1), \qquad \exists r_1 \sqsubseteq \exists r_2^-, \qquad (\mathsf{funct}\ r_2).$$

Note that in order to find all the simple finmod-cycles in a *DL-Lite*$^{\mathcal{F}}_{Horn}$ TBox $\mathcal{T}$ we rely on TBox reasoning w.r.t. arbitrary models on *DL-Lite*$^{\mathcal{F}}_{Horn}$. Thus, checking the existence of a simple finmod-cycle can be done in polynomial time on the size of $\mathcal{T}$ since TBox reasoning in *DL-Lite*$^{\mathcal{F}}_{Horn}$ is PTime-complete [9]. We then get the following complexity boundary.

Lemma 3.7. *Let $\mathcal{T}$ be a DL-Lite$^{\mathcal{F}}_{Horn}$ TBox. $\mathcal{T}_{\mathsf{f}}$ can be computed in polynomial time on the size of $\mathcal{T}$.*

Furthermore, soundness of simple-cycle-reversion, follows from Lemma 3.2.

Lemma 3.8 (soundness)**.** *Let $\mathcal{T}$ be a DL-Lite$^{\mathcal{F}}_{Horn}$-TBox.*

- *if $\mathcal{T}_{\mathsf{f}} \models \bigsqcap_k B_k \sqsubseteq B$ then $\mathcal{T} \models_{\mathsf{fin}} \bigsqcap_k B_k \sqsubseteq B$;*
- *if $(\mathsf{funct}\ R) \in \mathcal{T}_{\mathsf{f}}$ then $\mathcal{T} \models_{\mathsf{fin}} (\mathsf{funct}\ R)$.*

To prove the completeness of simple-cycle-reversion we use the model construction of the previous section. More specifically, we show the following.

Lemma 3.9. *For every DL-Lite$^{\mathcal{F}}_{Horn}$ axiom of the form $C \sqsubseteq B$, where C is a concept of the form $B_1 \sqcap \ldots \sqcap B_n$. $\mathcal{T} \models_{\mathsf{fin}} C \sqsubseteq B$ implies $\mathcal{T}_{\mathsf{f}} \models C \sqsubseteq B$.*

Proof. We show that if $\mathcal{T}_{\mathsf{f}} \not\models C \sqsubseteq B$ then $\mathcal{T} \not\models_{\mathsf{fin}} C \sqsubseteq B$. To prove the latter, we construct a finite model $\mathcal{J}$ of $\mathcal{T}_{\mathsf{f}}$ such that there is an object $\sigma \in \Delta^{\mathcal{J}}$ with $\sigma \in C^{\mathcal{J}}$ and $\sigma \notin B^{\mathcal{J}}$. We assume that $B_i \neq B_j$ for $i \neq j$, and that $n \geq 2$, we can make this assumption w.l.o.g. Indeed, every conjunction with some $B_i = B_j$ is logically equivalent to the one without repetitions. The latter assumption allows us to regard C as the conjunction of lite-types $\{[B_1], \ldots, [B_n]\}$. In what follows C will denote such conjunction of lite-types.

To construct $\mathcal{J}$, we modify the construction of the interpretation $\mathcal{I}^{\mathsf{f}}$ from Definition 3.2. We start with an initial interpretation $\mathcal{J}_0$ with domain

$$\Delta_0 := \{[B_i] \mid B_i \in \mathsf{BC}(\mathcal{T}_{\mathsf{f}}), B_i \not\sqsubseteq_{\mathcal{T}_{\mathsf{f}}} \bot\} \cup \{[C]\}$$

and setting for every concept name $A \in \mathsf{CN}(\mathcal{T}_{\mathsf{f}})$,

$$[B_i] \in A^{\mathcal{J}_0} \text{ iff } \mathcal{T}_{\mathsf{f}} \models B_i \sqsubseteq A, \qquad \text{and}$$
$$[C] \in A^{\mathcal{J}_0} \text{ iff } \mathcal{T}_{\mathsf{f}} \models C \sqsubseteq A,$$

and for every role name p, $p^{\mathcal{J}_0} = \emptyset$.

Using the relations in Definition 3.3 we can extend $\mathcal{J}_0$ into the interpretation $\mathcal{J}$ similarly as we did it for $\mathcal{I}^{\mathsf{f}}$. Further, the same argument as in the proof of Lemma 3.4 goes through to show that the constructed interpretation is also a finite model of $\mathcal{T}_{\mathsf{f}}$. By construction the object $[C] \notin B^{\mathcal{J}}$, and therefore $\mathcal{J} \not\models C \sqsubseteq B$ as required. □

Soundness and completeness of simple-cycle-reversion yield the following result.

Theorem 3.10. *For a given DL-Lite$^{\mathcal{F}}_{Horn}$ TBox $\mathcal{T}$, we have that the following hold:*

1. *$\mathcal{T}$ is finitely satisfiable iff* $\mathsf{finClosure}(\mathcal{T})$ *is satisfiable.*
2. *$\mathcal{T} \models_{\mathsf{fin}} C \sqsubseteq B$ iff* $\mathsf{finClosure}(\mathcal{T}) \models C \sqsubseteq B$.

We conclude that finite model TBox reasoning in *DL-Lite$^{\mathcal{F}}_{Horn}$* can be reduced to arbitrary TBox reasoning. Furthermore, Lemma 3.7 and the complexity of unrestricted reasoning in *DL-Lite$^{\mathcal{F}}_{Horn}$* imply the following complexity result.

Theorem 3.11. *Finite model satisfiability and subsumption in DL-Lite$^{\mathcal{F}}_{Horn}$ can be decided in* PTIME *on the size of the TBox.*

Since satisfiability and subsumption in *DL-Lite$_{Horn}$* is PTIME-complete, and that logic has the finite model property, we obtain a lower PTIME complexity bound for finite satisfiability and subsumption in *DL-Lite$^{\mathcal{F}}_{Horn}$*.

Theorem 3.12. *Deciding finite model satisfiability and subsumption in DL-Lite$^{\mathcal{F}}_{Horn}$ is* PTIME*-complete.*

3.3 Cycle reversion in Horn-$\mathcal{ALCFI}$

In what follows, we present the main contribution of this chapter: we show how finite model reasoning in Horn-$\mathcal{ALCQI}$ can be reduced to unrestricted reasoning by reversing cycles in the TBox. In order to simplify the presentation, we start with the sub logic Horn-$\mathcal{ALCFI}$ instead Horn-$\mathcal{ALCQI}$. In Section 3.5, we then extend our results to the full logic Horn-$\mathcal{ALCQI}$. In Section 3.4, we will show that the cycle reversion technique provides decision procedures for finite model reasoning in Horn-$\mathcal{ALCFI}$ and Horn-$\mathcal{ALCQI}$.

We start by defining a suitable notion of cycle in Horn-$\mathcal{ALCFI}$. Due to the presence of qualified existential restrictions, this logic is expressive enough to distinguish the type of successors of a given object in a model. Therefore, we need to proceed in a more sophisticated way than in Section 3.2 for defining and reversing cycles in a TBox. Recall from Definition 2.10 that Horn-$\mathcal{ALCFI}$ TBox axioms have the following (normal) form.

$$K \sqsubseteq A \qquad K \sqsubseteq \bot \qquad K \sqsubseteq \exists r.K' \qquad K \sqsubseteq \forall r.K' \qquad K \sqsubseteq (\leqslant 1\ r\ K')$$

where K and K' are conjunctions of concept names. For convenience, we will treat such conjunctions as sets, and hence write $A \in K$ or $K' \subseteq K$, to denote that a concept name A occurs in K and that K' is a conjunction of concept names occurring in K, respectively.

Definition 3.6. Let $\mathcal{T}$ be a Horn-$\mathcal{ALCFI}$ TBox. A *finmod cycle in $\mathcal{T}$* is a sequence $K_1, r_1, K_2, r_2, \ldots, r_{n-1}, K_n$, with $K_1, \ldots, K_n$ conjunctions of concept names and $r_1, \ldots, r_{n-1}$ (potentially inverse) roles such that $K_n = K_1$ and, for $1 \leq i < n$:

$$\mathcal{T} \models K_i \sqsubseteq \exists r_i.K_{i+1} \text{ and } \mathcal{T} \models K_{i+1} \sqsubseteq (\leqslant 1\ r_i^-\ K_i).$$

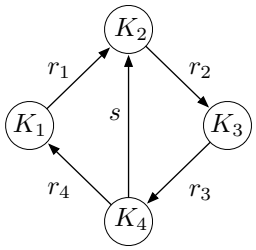


Figure 3.4: Cycles in a Horn $\mathcal{ALCQI}$ TBox

The *completion* $\mathcal{T}_\mathsf{f}$ of $\mathcal{T}$ is obtained by exhaustively applying the following rule:

(cycle-reversion) if $K_1, r_1, K_2, r_2, \ldots, r_{n-1}, K_n$ is a finmod cycle in $\mathcal{T}$, then extend $\mathcal{T}$ with the following concept inclusions, for $1 \leq j < n$:

$$K_{j+1} \sqsubseteq \exists r_j^-.K_j \quad \text{and} \quad K_j \sqsubseteq (\leqslant 1\ r_j\ K_{j+1}). \tag{3.11}$$

△

In principle, according to Definition 3.6 there may be infinitely many finmod cycles in a given TBox since repetition of conjunctions and roles within the cycle is not explicitly disallowed. Consider, for example, a TBox $\mathcal{T}'$ such that $K_1, r_1, K_2, r_2, K_3, r_3, K_4, r_4, K_1$ is a cycle in $\mathcal{T}'$, and $\mathcal{T}' \models K_4 \sqsubseteq \exists s.K_2$ and $\mathcal{T}' \models K_2 \sqsubseteq (\leqslant 1\ s^-\ K)_4$ (use Figure 3.4 as a visual aid). Then every sequence of the form

$$K_1, r_1, K_2, r_2, K_3, r_3, K_4, \sigma, r_4 K_1,$$

with $\sigma \in (s, K_2, r_2, K_3, r_3, K_4)^*$ is a cycle in $\mathcal{T}'$.

However, only finitely many CIs can be added to $\mathcal{T}$ by cycle-reversion since there are only finitely many conjunctions modulo simple equivalence: e.g., $A \sqcap A$ is equivalent to A. Indeed, recall that we regard conjunctions of concept names as *sets* of concept names. In particular, in the worst case exponentially many axioms –in the size of the original TBox– can be added to $\mathcal{T}$. We make the observation that for finding these finitely many CIs, it suffices to consider finmod-cycles in which all triples (K_i, r_{i+1}, K_{i+1}) are distinct. That is, we only consider *simple cycles* in the graph theoretical sense, i.e., cycles with no repetitions of conjunctions (vertices) and roles (edges) other than the starting and the ending conjunction. In our example above $K_1, r_1, K_2, r_2, K_3, r_3, K_4, r_4, K_1$ and $K_2, r_2, K_3, r_3, K_4, s, K_2$ are simple cycles.

Note that since the definition of a finmod-cycle is based on the subsumption entailed by $\mathcal{T}$, finding finmod cycles requires deciding unrestricted subsumption in Horn-$\mathcal{ALCFI}$ TBoxes, which is decidable and ExpTime-complete (in ExpTime since it is a fragment of $\mathcal{ALCQI}$ [47],

and ExpTime hard since it contains $\mathcal{EL}$ with functionality [13]). Let us consider the following example to illustrate cycle-reversion in an Horn-$\mathcal{ALCFI}$ TBox.

Example 6. Consider the TBox $\mathcal{T}'$ consisting of the following axioms:

$$A \sqsubseteq \exists r_1.B \qquad B \sqsubseteq (\leqslant 1\ r_1^-\ A) \qquad B \sqsubseteq \exists r_2.(A \sqcap D) \qquad A \sqsubseteq (\leqslant 1\ r_2^-\ B)$$
$$C_1 \sqsubseteq A \qquad C_1 \sqsubseteq \forall r_1.C_2 \qquad C_2 \sqsubseteq \forall r_2.C_1$$

By inspecting these axioms, it is easy to see that $\mathcal{T}'$ entails the following:

$$A \sqsubseteq \exists r_1.B \quad B \sqsubseteq (\leqslant 1\ r_1^-\ A) \quad B \sqsubseteq \exists r_2.A \quad A \sqsubseteq (\leqslant 1\ r_2^-\ B) \tag{3.12}$$

$$A \sqcap D \sqsubseteq \exists r_1.B \quad B \sqsubseteq (\leqslant 1\ r_1^-\ A \sqcap D) \quad B \sqsubseteq \exists r_2.(A \sqcap D) \quad A \sqcap D \sqsubseteq (\leqslant 1\ r_2^-\ B) \tag{3.13}$$

$$\begin{aligned} &A \sqcap C_1 \sqsubseteq \exists r_1.B && B \sqcap C_2 \sqsubseteq (\leqslant 1\ r_1^-\ A \sqcap C_1) \\ &B \sqcap C_2 \sqsubseteq \exists r_2.(A \sqcap C_1) && A \sqcap C_1 \sqsubseteq (\leqslant 1\ r_2^-\ B \sqcap C_2) \end{aligned} \tag{3.14}$$

Thus, $\mathcal{T}'$ contains the following three finmod-cycles.

From (3.12) :
$$\{A\}, r_1, \{B\}, r_2, \{A\} \tag{$\mathcal{C}_1$}$$
From (3.13) :
$$\{A, D\}, r_1, \{B\}, r_2, \{A, D\} \tag{$\mathcal{C}_2$}$$
From (3.14) :
$$\{A, C_1\}, r_1, \{B, C_2\}, r_2, \{A, C_1\} \tag{$\mathcal{C}_3$}$$

Now, applying cycle-reversion to the finmod-cycle ($\mathcal{C}_1$), the following axioms are added to $\mathcal{T}'_{\mathsf{f}}$:

$$\begin{aligned} &A \sqsubseteq \exists r_2^-.B, && B \sqsubseteq (\leqslant 1\ r_2\ A), \\ &B \sqsubseteq \exists r_1^-.A, && A \sqsubseteq (\leqslant 1\ r_1\ B). \end{aligned}$$

Additionally, for the cycle in ($\mathcal{C}_2$) cycle-revrsion adds the following:

$$\begin{aligned} &A \sqcap D \sqsubseteq \exists r_2^-.B, && B \sqsubseteq (\leqslant 1\ r_2\ A \sqcap D), \\ &B \sqsubseteq \exists r_1^-.A \sqcap D, && A \sqcap D \sqsubseteq (\leqslant 1\ r_1\ B). \end{aligned}$$

Finally, reversing the cycle in ($\mathcal{C}_3$) yields the addition of the axioms:

$$\begin{aligned} &A \sqcap C_1 \sqsubseteq \exists r_2^-.(B \sqcap C_2), && B \sqcap C_2 \sqsubseteq (\leqslant 1\ r_2\ A \sqcap C_1), \\ &B \sqcap C_2 \sqsubseteq \exists r_1^-.(A \sqcap C_1), && A \sqcap C_1 \sqsubseteq (\leqslant 1\ r_1\ B \sqcap C_2). \end{aligned}$$

Note that $\mathcal{T}_\mathsf{f}'$ contains: $A \sqsubseteq \exists r_1.B, \quad B \sqsubseteq \exists r_1^-.(A \sqcap D), \quad$ and $\quad B \sqsubseteq (\leqslant 1\ r_1^-\ A)$. Consequently, $\mathcal{T}_\mathsf{f}' \models A \sqsubseteq D$, which corresponds to the entailment $\mathcal{T}' \models_\mathsf{fin} A \sqsubseteq D$. Moreover, observe that cycles $(\mathcal{C}_1)$ and $(\mathcal{C}_2)$ are related to each other, in the sense that the axioms added by reversing $(\mathcal{C}_2)$ entail those added by reversing $(\mathcal{C}_1)$. On the other hand, this is not the case for $(\mathcal{C}_3)$, the axioms added by its reversion are not implied by any of the other two cycle reversion steps. $\overline{\wedge}$

In general, the CIs added by cycle-reversion are not necessarily implied by the original TBox w.r.t. arbitrary models. As an instance, for the TBox $\mathcal{T}'$ in the previous example, it holds that $\mathcal{T}' \not\models A \sqsubseteq D$. However, it can be shown that for any TBox $\mathcal{T}$ the axioms added by cycle-reversion are indeed entailed in finite models of $\mathcal{T}$.

Lemma 3.13. *Let $K_1, r_1, \ldots, r_{n-1}, K_n$ be a finmod cycle in $\mathcal{T}$, then $\mathcal{T} \models_\mathsf{fin} K_{i+1} \sqsubseteq \exists r_i^-.K_i$ and $\mathcal{T} \models_\mathsf{fin} K_i \sqsubseteq (\leqslant 1\ r_i\ K_{i+1})$ for $1 \leq i < n$.*

Proof. We have to show that, if $K_1, r_1, \ldots, r_{n-1}, K_n$ is a finmod cycle in $\mathcal{T}$, then for every finite model $\mathcal{I}$ of $\mathcal{T}$, it holds that for $1 \leq i < n$:

(i) $\mathcal{I} \models K_{i+1} \sqsubseteq \exists r_i^-.K_i$, and

(ii) $\mathcal{I} \models K_i \sqsubseteq (\leqslant 1\ r_i\ K_{i+1})$.

Indeed, let $\mathcal{I}$ be a finite model of $\mathcal{T}$. Since $K_1, r_1, \ldots, r_{n-1}, K_n$ is a finmod cycle, then $K_i^\mathcal{I} \subseteq (\exists r_i.K_{i+1})^\mathcal{I}$ and $K_{i+1}^\mathcal{I} \subseteq (\leqslant 1\ r_i^-\ K_i)^\mathcal{I}$ for $1 \leq i < n$. We first note that, by the semantics of Horn-$\mathcal{ALCFI}$, we must have $|K_1^\mathcal{I}| \leq \cdots \leq |K_n^\mathcal{I}|$. Furthermore, we have that $|K_1^\mathcal{I}| = \cdots = |K_n^\mathcal{I}|$ since $K_n = K_1$.

Now, fix some i with $1 \leq i < n$. $K_i^\mathcal{I} \subseteq (\leqslant 1\ r_i\ K_{i+1})^\mathcal{I}$ and $K_{i+1}^\mathcal{I} \subseteq (\exists r_i^-.K_i)^\mathcal{I}$ follow from $|K_i^\mathcal{I}| = |K_{i+1}^\mathcal{I}|$, $K_i^\mathcal{I} \subseteq (\exists r_i.K_{i+1})^\mathcal{I}$, and $K_{i+1}^\mathcal{I} \subseteq (\leqslant 1\ r_i^-\ K_i)^\mathcal{I}$ and $\mathcal{I}$ being finite. Thus, (i) and (ii) above hold as required. $\square$

As a consequence of Lemma 3.13 we obtain the following result showing that cycle-reversion is sound w.r.t. finite model entailment.

Theorem 3.14. *Let $\mathcal{T}$ be a Horn-$\mathcal{ALCFI}$ TBox. For every CI $K \sqsubseteq C$, if $\mathcal{T}_\mathsf{f} \models K \sqsubseteq C$ then $\mathcal{T} \models_\mathsf{fin} K \sqsubseteq C$.*

We now aim to prove that cycle-reversion provides a complete axiomatization for finite model entailment. In particular, we show the following.

Theorem 3.15. *Let $\mathcal{T}$ be a Horn-$\mathcal{ALCFI}$ TBox. $\mathcal{T} \models_\mathsf{fin} K \sqsubseteq C$ implies $\mathcal{T}_\mathsf{f} \models K \sqsubseteq C$.*

In order to prove the previous theorem, we provide a construction of a finite model of the closure of a TBox with particular properties. Indeed, this construction constitutes the main technical tool used to prove the results in this Chapter. For this reason, we devote the next Section to describe the details of the construction. Afterwards, in Section 3.4, we will utilize it to establish the results on finite model reasoning in Horn-$\mathcal{ALCQI}$.

3.3.1 Constructing Finite Models of Horn-$\mathcal{ALCFI}$ TBoxes

In this section, we will describe the construction of a finite model of a given (satisfiable) TBox $\mathcal{T}$. The construction will be guided by $\mathcal{T}_\mathsf{f}$, thus we will be constructing actually a finite $\mathcal{I}$ model of $\mathcal{T}_\mathsf{f}$, and show that this model is also a finite model of $\mathcal{T}$. Notably, $\mathcal{I}$ provides a witness for every concept satisfiable w.r.t. $\mathcal{T}_\mathsf{f}$. Later on, we will use this property for showing that cycle-reversion provides a complete axiomatization w.r.t. finite model entailment.

In order to describe the satisfiable concepts w.r.t. a certain TBox $\mathcal{T}$, we will use the notion of *type*. For Horn DLs, and in particular for Horn-$\mathcal{ALCFI}$, a type is determined by a set of satisfiable concept names. It suffices to consider such types since our TBoxes are in normal form. We will describe this and other relevant preliminary notions in detail in what follows.

Definition 3.7 (Horn-$\mathcal{ALCQI}$ type)**.** Let $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ be an interpretation, and $d \in \Delta^\mathcal{I}$. The *type realized at d in $\mathcal{I}$* is defined as the following set of concept names:

$$\mathsf{tp}_\mathcal{I}(d) := \{A \in \mathsf{CN}(\mathcal{T}) \mid d \in A^\mathcal{I}\}.$$

Let $\mathcal{T}$ be an Horn-$\mathcal{ALCQI}$ TBox, and let $\mathsf{CN}(\mathcal{T})$ be the set of concept names occurring in $\mathcal{T}$. A *type for* $\mathcal{T}$ is a subset $t \subseteq \mathsf{CN}(\mathcal{T})$ such that there is a model $\mathcal{I}$ of $\mathcal{T}$ and a $d \in \Delta^\mathcal{I}$ such that

$$\mathsf{tp}_\mathcal{I}(d) = t.$$

We use $\mathsf{TP}(\mathcal{T})$ to denote the set of all types for $\mathcal{T}$. △

Intuitively, one could think of a type t for $\mathcal{T}$ as a satisfiable (maximal) conjunction t w.r.t. $\mathcal{T}$.

Example 7. Consider a TBox $\mathcal{T}_3$ containing the following axioms:

$$D_1 \sqsubseteq \exists r_1.D_2 \qquad\qquad D_2 \sqsubseteq \forall r_1^-.D_3$$

Then $t = \{D_1, D_3\}$ and $t' = \{D_2\}$ are types for $\mathcal{T}_3$, while $\{D_1\}$ is not a type for $\mathcal{T}_3$. Indeed, in every model $\mathcal{I}$ of $\mathcal{T}_3$, $d \in D_1^\mathcal{I}$ implies $d \in D_3^\mathcal{I}$ since $\mathcal{T}_3 \models D_1 \sqsubseteq D_3$, thus $\mathsf{tp}_\mathcal{I}(d) = \{D_1, D_3\}$. ⩚

We observe that in general the types for a TBox differ from the ones for its closure. For example, consider the TBox $\mathcal{T}$ from Example 6; we have that $\{A\}$ and $\{A, C_1\}$ are types for $\mathcal{T}$, but they are not types for $\mathcal{T}_\mathsf{f}$. Note that by the added axioms $\mathcal{T}_\mathsf{f} \models A \sqsubseteq D$, hence those types are dropped and instead we have $\{A, D\}$ and $\{A, C_1, D\}$ as types for $\mathcal{T}_\mathsf{f}$.

For the purpose of constructing a finite model of a given TBox $\mathcal{T}$, we will be interested in the set of types for $\mathcal{T}_\mathsf{f}$, and moreover, we will have special interest in some relations holding among these types.

Definition 3.8. Let $\mathcal{T}$ be an Horn-$\mathcal{ALCFI}$ TBox. For $t, t' \in \mathsf{TP}(\mathcal{T}_\mathsf{f})$ and $r \in \mathsf{role}(\mathcal{T})$, we write

- $t \to_r t'$ if $\mathcal{T}_\mathsf{f} \models t \sqsubseteq \exists r.t'$ and t' is maximal with this property;
- $t \to^1_r t'$ if $t \to_r t'$ and $\mathcal{T}_\mathsf{f} \models t' \sqsubseteq (\leqslant 1\ r^-\ t)$;
- $t\ {}^1{\leftrightarrow}^1_r\ t'$ if $t \to^1_r t'$ and $t' \to^1_{r^-} t$.

△

We will use these relations to formalize what we call $\exists$-*requirement* during the construction of the finite interpretation $\mathcal{I}$. Intuitively, for an element $d \in \Delta^\mathcal{I}$ such that $\mathsf{tp}_\mathcal{I}(d) = t$, if it holds $t \to_r t'$ for some role $r \in \mathsf{role}(\mathcal{T})$, then there must exist some $d' \in \Delta^\mathcal{I}$ with $\mathsf{tp}_\mathcal{I}(d') = t'$, in order for $\mathcal{I}$ to be a model of $\mathcal{T}_\mathsf{f}$.

Observation 2. Whenever

$$t_1 \to^1_{r_1} t_2 \to^1_{r_2} \cdots \to^1_{r_{n-1}} t_n = t_1 \tag{3.15}$$

then $t_1, r_1, \ldots, r_{n-1}, t_n$ is a finmod cycle in $\mathcal{T}_\mathsf{f}$. Further, each $\to^1_{r_i}$ in (3.15) can be replaced with ${}^1{\leftrightarrow}^1_{r_i}$, for $i \in \{1, \ldots, n-1\}$ since all cycles have been reversed in $\mathcal{T}_\mathsf{f}$.

Types related by ${}^1{\leftrightarrow}^1_r$ are connected very tightly by the TBox $\mathcal{T}$. Recall that $t\ {}^1{\leftrightarrow}^1_r\ t'$ implies that r is functional and inverse functional on the instances of t and t', which means that $r^\mathcal{I}$ is a bijection on $t^\mathcal{I} \times t'^\mathcal{I}$ in any model $\mathcal{I}$ of $\mathcal{T}$. Further, if $\mathcal{I}$ is finite, then it holds that the number of instances of t and t' coincide. For that reason, we consider the types related by ${}^1{\leftrightarrow}^1_*$ as a 'whole' in our construction of finite models. We formalize that by the notion of a *type class*.

Definition 3.9. A non-empty set $P \subseteq \mathsf{TP}(\mathcal{T})$ is *type class* if it is the minimal set satisfying the following:

> if $t \in P$ and there is a role $r \in \mathsf{role}(\mathcal{T})$, such that $t\ {}^1{\leftrightarrow}^1_r\ t'$, then $t' \in P$.

△

Note that the set of all type classes is a partition of $\mathsf{TP}(\mathcal{T}_\mathsf{f})$. Indeed, for each type t for $\mathcal{T}_\mathsf{f}$, there is a unique type class P, such that $t \in P$. Thus, for a given type t we can refer to *the type class* of t. We set $P \prec P'$ if there are $t \in P$ and $t' \in P'$ with $t' \subsetneq t$. We will later be referring to the strict partial order that is obtained by taking the transitive closure of $\prec$, denoted by $\prec^+$. We prove the following.

Lemma 3.16. *$\prec^+$ is a strict partial order.*

Proof. Since $\prec^+$ is transitive by definition, it remains to establish irreflexivity and asymmetry. To this end, it suffices to show that $\prec$ is acyclic in the sense that there are no type partitions $P_0, \dots, P_n$, $n \geq 0$, such that $P_0 \prec \cdots \prec P_n = P_0$. Assume to the contrary that there are such $P_0, \dots, P_n$. By reversing the order, we can assume that $P_0 \succ \cdots \succ P_n = P_0$. Then there are, for each $i < n$, types $t_i \in P_i$ and $t'_{i+1} \in P_{i+1}$ such that $t_i \subsetneq t'_{i+1}$. For uniformity, set $t_n = t_0$ and $t'_0 = t'_n$.

Let $i < n$. By definition of type partitions, and because $t \; {}^1\!\leftrightarrow^1_r \; t'$ implies $t' \; {}^1\!\leftrightarrow^1_{r^-} \; t$ for all types t, t' and roles r, we can derive from $t_i, t'_i \in P_i$ the existence of types $s_{0,i}, \dots, s_{k_i,i} \in P_i$, $k_i \geq 0$, and roles $r_{0,i}, \dots, r_{k_i-1,i}$ such that

$$t_i = s_{0,i} \; {}^1\!\leftrightarrow^1_{r_{0,i}} \; s_{1,i} \; {}^1\!\leftrightarrow^1_{r_{1,i}} \; \cdots \; {}^1\!\leftrightarrow^1_{r_{k_i-1,i}} \; s_{k_i,i} = t'_i.$$

For each i, we thus find a sequence

$$t_i, r_{0,i}, s_{1,i}, \dots, s_{k_i-1,i}, r_{k_i-1,i}, t'_i \tag{$*$}$$

that satisfies the prerequisites for finmod cycles, namely

$$\mathcal{T} \models s_{j,i} \sqsubseteq \exists r_{j,i}.s_{j+1,i} \tag{3.16}$$

$$\mathcal{T} \models s_{j+1,i} \sqsubseteq (\leqslant 1 \; r_{j,i} \; s_{j,i}) \tag{3.17}$$

for all $j = 0, \dots, k_i$ (but this sequence need not be a finmod cycle since $t_i = t'_i$ is not guaranteed). Note that we cannot have $k_i = 0$ for all i since otherwise

$$t_0 \subsetneq t'_1 = t_1 \subsetneq t'_2 = t_2 \subsetneq \cdots \subsetneq t'_n = t_n,$$

in contradiction to $t_n = t_0$. In the following, we can thus assume that $k_i > 0$ for at least one i.

Because of (3.17), we have $\mathcal{T}_\mathsf{f} \models t_i \sqsubseteq \exists r_{0,i}.s_{1,i}$ and $\mathcal{T}_\mathsf{f} \models s_{1,i} \sqsubseteq (\leqslant 1 \; r^-_{0,i} \; t_i)$. Because of $t_i \subsetneq t'_{i+1}$, we thus obtain $\mathcal{T}_\mathsf{f} \models t'_{i+1} \sqsubseteq \exists r_{0,i}.s_{1,i}$ and $\mathcal{T}_\mathsf{f} \models s_{1,i} \sqsubseteq (\leqslant 1 \; r^-_{0,i} \; t'_{i+1})$. Consequently, the following sequences also satisfy conditions (3.16) and (3.17):

$$\begin{gathered} t'_n, r_{0,n-1}, s_{1,n-1}, \dots, s_{k_{n-1}-1,n-1}, r_{k_{n-1}-1,n-1}, t'_{n-1} \\ t'_{n-1}, r_{0,n-2}, s_{1,n-2}, \dots, s_{k_{n-1}-1,n-2}, r_{k_{n-1}-1,n-2}, t'_{n-2} \\ \vdots \\ t'_1, r_{0,0}, s_{1,0}, \dots, s_{k_0-1,0}, r_{k_0-1,0}, t'_0. \end{gathered}$$

Since $t'_0 = t'_n$, we can concatenate all these sequences to a finmod cycle. As $k_i > 0$ for at least one i, this cycle is non-empty, and the construction of $\mathcal{T}_\mathsf{f}$ ensures that the reversed cycle is also present in $\mathcal{T}_\mathsf{f}$. Let us assume w.l.o.g. that $k_{n-1} > 0$. The presence of the reversed cycle yields $\mathcal{T}_\mathsf{f} \models s_{1,n-1} \sqsubseteq \exists r^-_{0,n-1}.t'_n$. Since $t_n \; {}^1\!\leftrightarrow^1_{r_{0,n-1}} \; s_{1,n-1}$, we have $\mathcal{T}_\mathsf{f} \models s_{1,n-1} \sqsubseteq \exists r^-_{0,n-1}.t_{n-1}$ and t_{n-1} is maximal with this property. This is a contradiction to $t_{n-1} \supsetneq t'_n$.

□

Before moving forward, let us illustrate with an example the defined notions to understand their relation to the finmod-cycles in a TBox.

Example 8. Consider the TBox $\mathcal{T}'$ from Example 6. Recall that it contains the following cycles:

$$\begin{aligned} \{A\}, r_1, \{B\}, r_2, \{A\} \\ \{A, D\}, r_1, \{B\}, r_2, \{A, D\} \\ \{A, C_1\}, r_1, \{B, C_2\}, r_2, \{A, C_1\} \end{aligned} \tag{3.18}$$

Some of the relations holding on types for $\mathcal{T}'_{\mathsf{f}}$ are the following:

$$\{A, D\} \; {}^{1}\!\leftrightarrow^{1}_{r_1} \; \{B\} \; {}^{1}\!\leftrightarrow^{1}_{r_2} \; \{A, D\} \tag{3.19}$$

$$\{A, C_1, D\} \; {}^{1}\!\leftrightarrow^{1}_{r_1} \; \{B, C_2\} \; {}^{1}\!\leftrightarrow^{1}_{r_2} \; \{A, C_1, D\} \tag{3.20}$$

Which gives the two type classes

$$P_1 = \{\{A, D\}, \{B\}\} \quad \text{and} \quad P_2 = \{\{A, C_1, D\}, \{B, C_2\}\}$$

with $P_2 \prec^+ P_1$. Intuitively, those type classes capture the two relevant cycles in $\mathcal{T}'_{\mathsf{f}}$, i.e., those in (3.19) and (3.20). This also formalizes the idea that the first two cycles in (3.18) are in some sense 'equivalent'.

$\overline{\wedge}$

We are now ready to describe the construction of a finite interpretation of $\mathcal{T}_{\mathsf{f}}$. The initial version of such an interpretation contains one element for each type for $\mathcal{T}_{\mathsf{f}}$, we then exhaustively apply three *completion rules* denoted with (**c1**) to (**c3**) to that initial interpretation to obtain the desired finite interpretation $\mathcal{I}$. The completion repeatedly introduces elements following $\exists$-requirements determined by $t \to_r t'$ relations and carefully distinguishing several cases to ensure that functionality restrictions are always satisfied. An element $d \in \Delta^{\mathcal{I}}$ has an $\exists$-requirement if $\mathsf{tp}_{\mathcal{I}}(d) \to_r t'$ for some role r and type t'. We will make sure that the following invariants are satisfied after each completion step:

(**i1**) $\mathsf{tp}_{\mathcal{I}}(d) \in \mathsf{TP}(\mathcal{T}_{\mathsf{f}})$ for all $d \in \Delta^{\mathcal{I}}$;

(**i2**) if $(d, d') \in r^{\mathcal{I}}$, then we have $\mathsf{tp}_{\mathcal{I}}(d) \to_r \mathsf{tp}_{\mathcal{I}}(d')$ or $\mathsf{tp}_{\mathcal{I}}(d') \to_{r^-} \mathsf{tp}_{\mathcal{I}}(d)$;

(**i3**) if $\mathcal{T}_{\mathsf{f}} \models K \sqsubseteq (\leqslant 1\ r\ K')$, then $\mathcal{I} \models K \sqsubseteq (\leqslant 1\ r\ K')$.

Intuitively, invariant (**i1**) ensures that every element of the domain is an instance of a type for $\mathcal{T}_{\mathsf{f}}$, hence by adding new elements to the domain we do not realize any unsatisfiable concept w.r.t. $\mathcal{T}_{\mathsf{f}}$. Further, (**i2**) will ensure that domain elements are connected by a role r following $\exists$-requirements entailed by $\mathcal{T}_{\mathsf{f}}$. Finally, (**i3**) ensures that no role functionality entailed by $\mathcal{T}_{\mathsf{f}}$ is violated.

Definition 3.10 (finite completion)**.** The interpretation $\mathcal{I}$ is defined starting from an initial domain which contains an element d_t for every $t \in \mathsf{TP}(\mathcal{T}_\mathsf{f})$. We start by setting

$$\begin{aligned} \Delta^\mathcal{I} &= \{d_t \mid t \in \mathsf{TP}(\mathcal{T}_\mathsf{f})\}; \\ A^\mathcal{I} &= \{\ d_t \in \Delta^\mathcal{I} \mid A \in t\ \}; \\ r^\mathcal{I} &= \emptyset. \end{aligned} \tag{3.21}$$

Next, we describe the completion steps (**c1**)–(**c3**). Briefly, each of these steps selects and element $d \in \Delta^\mathcal{I}$ whenever there is an $\exists$-requirement $t \to_r t'$ that is not satisfied at d in $\mathcal{I}$. In order to construct the desired (finite) model we apply these steps exhaustively, therefore *all* the $\exists$-requirements are satisfied in the limit of the construction. Both (**c1**) and (**c2**) introduce fresh elements into $\Delta^\mathcal{I}$, according to the nature of the $\exists$-requirement. Note that if $t \to^1_r t'$ holds (i.e., r^- is functional on t') then one fresh element d' realizing type t' is required as the r-successor of d to satisfy the requirement; this is the case handled by (**c1**). However, if additionally we have that $t' \to^1_{r^-} t$ (which means that $t\ {}^1{\leftrightarrow}^1_r\ t'$ holds) then (**c2**) is applied. The application of (**c2**) treats the type class P of t, which amounts to satisfy multiple $\exists$-requirements in $\mathcal{I}$. In order to correctly construct the model $\mathcal{I}$, we establish a preference on the application of these two steps by giving priority to applications of (**c1**) over those of (**c2**). We explain in detail the reason for this, once the definition of the steps is in place. On the other hand, (**c3**) does not introduce elements into $\Delta^\mathcal{I}$, but instead reuses an existing element to satisfy the existential requirement for the selected object d. Note that this can be done without violating functionality constraints whenever $t \to^1_r t'$ does not hold. We define the completion steps as follows:

(**c1**) Choose an element $d \in \Delta^\mathcal{I}$ such that $\mathsf{tp}_\mathcal{I}(d) \to^1_r t$, $t \not\to^1_{r^-} \mathsf{tp}_\mathcal{I}(d)$, and $d \notin (\exists r.t)^\mathcal{I}$, then

- add a fresh elements e to Δ, and
- modify the extension of concept and role names such that $\mathsf{tp}_\mathcal{I}(e) = t$ and $(d, e) \in r^\mathcal{I}$.

(**c2**) Choose a type class P that is minimal w.r.t. the order $\prec^+$, a $\lambda = t\ {}^1{\leftrightarrow}^1_r\ t'$ with $t \in P$, and an element $d \in \Delta^\mathcal{I}$ with $\exists$-requirement $t\ {}^1{\leftrightarrow}^1_r\ t'$; and let $n_{\max} = \max\{|s^\mathcal{I}| \mid s \in P\}$, for each $s \in P$, take a fresh set of domain elements

$$\Delta_s := \{d_{s,j} \mid |s^\mathcal{I}| < j \leq n_{\max}\}.$$

For each $\lambda = s\ {}^1{\leftrightarrow}^1_r\ s'$ with $s \in P$, let

$$X^\mathcal{I}_{\lambda,1} = s^\mathcal{I} \setminus (\exists r.s')^\mathcal{I} \qquad \text{and} \qquad X^\mathcal{I}_{\lambda,2} = s'^\mathcal{I} \setminus (\exists r^-.s)^\mathcal{I}$$

and choose a bijection π_λ between $X^\mathcal{I}_{\lambda,1} \cup \Delta_s$ and $X^\mathcal{I}_{\lambda,2} \cup \Delta_{s'}$, [1] then

- add the elements in $\biguplus_{s \in P} \Delta_s$ to Δ;
- for each $\lambda = s\ {}^1{\leftrightarrow}^1_r\ s'$ with $s, s' \in P$, extend $r^\mathcal{I}$ with π_λ, and

[1] The construction of the sets Δ_s clearly ensures that their union has the required cardinality.

– interpret concept names so that $\mathsf{tp}_{\mathcal{I}}(d) = s$ for each $d \in \Delta_s$, and $s \in P$.

(**c3**) Choose an element $d \in \Delta^{\mathcal{I}}$ such that $\mathsf{tp}_{\mathcal{I}}(d) \to_r t$, $\mathsf{tp}_{\mathcal{I}}(d) \not\to^1_r t$, and $d \notin (\exists r.t)^{\mathcal{I}}$. Add (d, d_t) to $r^{\mathcal{I}}$, where $d_t \in \Delta^{\mathcal{I}}$ is the element introduced for type t in the initialization step (3.21). △

In order to show that the described construction of $\mathcal{I}$ is correct for our purposes, we prove that

1. $\mathcal{I}$ is indeed a model of $\mathcal{T}_{\mathsf{f}}$, and
2. $\mathcal{I}$ is finite, that is, $\Delta^{\mathcal{I}}$ is finite.

For the purpose of Point 1, we first show that after each step of the construction all the invariants are preserved. Observe that from (3.21) it follows that the initial interpretation $\mathcal{I}$ satisfies (**i1**), (**i2**) and (**i3**). We show now that each of the completion steps (**c1**) to (**c3**) preserves the invariants.

Lemma 3.17. *Each application of a completion step preserves* (**i1**)*,* (**i2**) *and* (**i3**)*.*

Proof.

(c1) preserves all invariants.

- **case (i1) and (i2):** The definition of (**c1**) ensures that the invariants (**i1**) and (**i2**) are preserved after each single application of (**c1**).
- **case (i3):** Assume that completion treated $d \in \Delta^{\mathcal{I}}$ with $\mathsf{tp}_{\mathcal{I}}(d) \to^1_r t$ and $t \not\to^1_{r^-} \mathsf{tp}_{\mathcal{I}}(d)$, and that after the application $(d, d_1) \in r^{\mathcal{I}}$, with d_1 the fresh domain element added. Assume to the contrary of what is to be proved that $\mathcal{T}_{\mathsf{f}} \models K \sqsubseteq (\leqslant 1\ r\ K')$ and there is a $d_2 \in \Delta^{\mathcal{I}}$ distinct from d_1 such that $d \in K^{\mathcal{I}}$, $(d, d_2) \in r^{\mathcal{I}}$, and $d_1, d_2 \in K'^{\mathcal{I}}$. We aim to show that if such d_2 exists then $t \subseteq \mathsf{tp}_{\mathcal{I}}(d_2)$, which establishes a contradiction to the fact that $d \notin (\exists r.t)^{\mathcal{I}}$ was true before the rule application. According to (**i2**), we can distinguish the following cases:
  - $\mathsf{tp}_{\mathcal{I}}(d) \to_r \mathsf{tp}_{\mathcal{I}}(d_2)$. Then $\mathcal{T}_{\mathsf{f}} \models \mathsf{tp}_{\mathcal{I}}(d) \sqsubseteq \exists r.\mathsf{tp}_{\mathcal{I}}(d_2)$ and $\mathsf{tp}_{\mathcal{I}}(d_2)$ is maximal with this property. From $\mathsf{tp}_{\mathcal{I}}(d) \to_r t$, we additionally get $\mathcal{T}_{\mathsf{f}} \models \mathsf{tp}_{\mathcal{I}}(d) \sqsubseteq \exists r.t$. Furthermore, $K \subseteq \mathsf{tp}_{\mathcal{I}}(d)$ and $d_1, d_2 \in K'^{\mathcal{I}}$ imply $K' \subseteq \mathsf{tp}_{\mathcal{I}}(d_2) \cap t$, a simple semantic argument shows that $\mathcal{T}_{\mathsf{f}} \models K \sqsubseteq \exists r.(\mathsf{tp}_{\mathcal{I}}(d_2) \cup t)$. The maximality of $\mathsf{tp}_{\mathcal{I}}(d_2)$ thus implies $t \subseteq \mathsf{tp}_{\mathcal{I}}(d_2)$.
  - $\mathsf{tp}_{\mathcal{I}}(d_2) \to_{r^-} \mathsf{tp}_{\mathcal{I}}(d)$. Then $\mathcal{T}_{\mathsf{f}} \models \mathsf{tp}_{\mathcal{I}}(d_2) \sqsubseteq \exists r^-.\mathsf{tp}_{\mathcal{I}}(d)$ and, additionally, we have $\mathcal{T}_{\mathsf{f}} \models \mathsf{tp}_{\mathcal{I}}(d) \sqsubseteq \exists r.t$. Since $K \subseteq \mathsf{tp}_{\mathcal{I}}(d)$ and $K' \subseteq \mathsf{tp}_{\mathcal{I}}(d_2) \cap t$, a simple semantic argument shows that $\mathcal{T}_{\mathsf{f}} \models \mathsf{tp}_{\mathcal{I}}(d_2) \sqsubseteq t$. Since $\mathsf{tp}_{\mathcal{I}}(d_2)$ is a type for $\mathcal{T}_{\mathsf{f}}$ by (**i1**), it follows that $t \subseteq \mathsf{tp}_{\mathcal{I}}(d_2)$.

(**c2**) **preserves all the invariants** To eliminate further case distinctions later on, for each $\lambda = s\ {}^{1}\!\leftrightarrow^{1}_{r}\ s'$ let λ^- denote $s'\ {}^{1}\!\leftrightarrow^{1}_{r^-}\ s$. Indeed, note that λ holds if and only if λ^- does. Assume that (**c2**) is applied to a type class P. Then define π_{λ^-} to be the converse of π_λ, for all $\lambda = s\ {}^{1}\!\leftrightarrow^{1}_{r}\ s'$ with r an inverse role. Note that π_λ is a bijection from $X^{\mathcal{I}}_{\lambda,1} \cup \Delta_s$ to $X^{\mathcal{I}}_{\lambda,2} \cup \Delta_{s'}$, just as in the case where r is a role name. Also note that whenever $(d, e) \in r^{\mathcal{I}}$ is added by the current application of (**c2**) with r a (potentially inverse) role, then there is a $\lambda = s\ {}^{1}\!\leftrightarrow^{1}_{r}\ s'$ such that $(d, d') \in \pi_\lambda(d)$. Moreover, we will need an intermediate technical lemma establishing that if the application of (**c2**) adds a pair (d, d') to $r^{\mathcal{I}}$ in order to satisfy an $\exists$-requirement $s\ {}^{1}\!\leftrightarrow^{1}_{r}\ s'$, then the types of d and d' are exactly s and s' respectively.

- **case (i1):** Invariant (**i1**) is preserved by each single application of (**c2**) by definition.

- **case (i2):** Consider a (potentially inverse) role r and a pair $(d, d') \in r^{\mathcal{I}}$ that has been added in a (**c2**) application. Take $\lambda = s\ {}^{1}\!\leftrightarrow^{1}_{r}\ s'$ such that $(d, d') \in \pi_\lambda(d)$. From Lemma 3.18, we obtain $\mathsf{tp}_{\mathcal{I}}(d) = s$ and $\mathsf{tp}_{\mathcal{I}}(d') = s'$. Consequently, $s\ {}^{1}\!\leftrightarrow^{1}_{r}\ s'$ yields $\mathsf{tp}_{\mathcal{I}}(d) \to_r \mathsf{tp}_{\mathcal{I}}(d')$ *and* $\mathsf{tp}_{\mathcal{I}}(d') \to_{r^-} \mathsf{tp}_{\mathcal{I}}(d)$. Thus, (**i2**) is preserved.

- **case (i3):** Let $\mathcal{T}_{\mathsf{f}} \models K \sqsubseteq (\leqslant 1\ r\ K')$, and assume to the contrary of what is to be shown that, after some application of (**c2**), there are $(d, d_1), (d, d_2) \in r^{\mathcal{I}}$ with $d \in K^{\mathcal{I}}$, $d_1, d_2 \in K'^{\mathcal{I}}$, and $d_1 \neq d_2$. We distinguish the following cases:

  (d, d_1) **was added by an application of (c2),** (d, d_2) **was not**. By the former, there is $\lambda = s\ {}^{1}\!\leftrightarrow^{1}_{r}\ s'$ such that $(d, d_1) \in \pi_\lambda$. By Lemma 3.18, $\mathsf{tp}_{\mathcal{I}}(d) = s$ and $\mathsf{tp}_{\mathcal{I}}(d_1) = s'$.

  We aim to show that $s' \subseteq \mathsf{tp}_{\mathcal{I}}(d_2)$ because this means that $d \in (\exists r.s')^{\mathcal{I}}$ was true before the current appliction of (**c2**), in contradiction to d being in the domain of π_λ. Since (d, d_2) was not added by (**c2**), by (**i2**) we can distinguish the following subcases:

  - $\mathsf{tp}_{\mathcal{I}}(d) \to_r \mathsf{tp}_{\mathcal{I}}(d_2)$. Thus $\mathcal{T}_{\mathsf{f}} \models \mathsf{tp}_{\mathcal{I}}(d) \sqsubseteq \exists r.\mathsf{tp}_{\mathcal{I}}(d_2)$ and $\mathsf{tp}_{\mathcal{I}}(d_2)$ is maximal with this property. Since $\mathsf{tp}_{\mathcal{I}}(d) = s$ and by λ, $\mathcal{T}_{\mathsf{f}} \models \mathsf{tp}_{\mathcal{I}}(d) \sqsubseteq \exists r.s'$. Using the facts that $\mathcal{T}_{\mathsf{f}} \models K \sqsubseteq (\leqslant 1\ r\ K')$, $K \subseteq \mathsf{tp}_{\mathcal{I}}(d) = s$, $K' \subseteq \mathsf{tp}_{\mathcal{I}}(d_2)$, and $K' \subseteq \mathsf{tp}_{\mathcal{I}}(d_1) = s'$, an easy semantic argument shows that $\mathcal{T}_{\mathsf{f}} \models \mathsf{tp}_{\mathcal{I}}(d) \sqsubseteq \exists r.(\mathsf{tp}_{\mathcal{I}}(d_2) \cup s')$. The maximality of $\mathsf{tp}_{\mathcal{I}}(d_2)$ thus yields $s' \subseteq \mathsf{tp}_{\mathcal{I}}(d_2)$.

  - $\mathsf{tp}_{\mathcal{I}}(d_2) \to_{r^-} \mathsf{tp}_{\mathcal{I}}(d)$. Then $\mathcal{T}_{\mathsf{f}} \models \mathsf{tp}_{\mathcal{I}}(d_2) \sqsubseteq \exists r^-.s$. By λ, we have $\mathcal{T}_{\mathsf{f}} \models s \sqsubseteq \exists r.s'$. Since $K \subseteq s$, $K \subseteq \mathsf{tp}_{\mathcal{I}}(d_2)$, $K \subseteq \mathsf{tp}_{\mathcal{I}}(d_1) = s'$, and $\mathcal{T}_{\mathsf{f}} \models K \sqsubseteq (\leqslant 1\ r\ K')$, a simple semantic argument shows that $s' \subseteq \mathsf{tp}_{\mathcal{I}}(d_2)$.

  - $r(d, d_2) \in \mathcal{A}$. Since $d \in K^{\mathcal{I}}$ and $d_2 \in K'^{\mathcal{I}}$, we have $K \subseteq \mathsf{tp}_{\mathcal{A}}(d)$ and $K' \subseteq \mathsf{tp}_{\mathcal{A}}(d_2)$ by definition of the initial interpretation $\mathcal{I}$. Also, $\mathsf{tp}_{\mathcal{A}}(d) = s$. By λ, we thus have $\mathcal{T}_{\mathsf{f}} \models \mathsf{tp}_{\mathcal{A}}(d) \sqsubseteq \exists r.s'$. With $\mathcal{T}_{\mathsf{f}} \models K \sqsubseteq (\leqslant 1\ r\ K')$ and $r(d, d_2) \in \mathcal{A}$, the semantics yields $s' \subseteq \mathsf{tp}_{\mathcal{A}}(d_2)$, thus $s' \subseteq \mathsf{tp}_{\mathcal{I}}(d_2)$.

  both (d, d_1) **and** (d, d_2) **were added by an application of (c2)**. Then there are λ_1 and λ_2, such that, for $i \in \{1, 2\}$, we have $\lambda_i = s_i\ {}^{1}\!\leftrightarrow^{1}_{r}\ s'_i$ and $(d, d_i) \in \pi_{\lambda_i}$. Applying

Lemma 3.18 to λ_i yields $s_i = \mathsf{tp}_{\mathcal{I}}(d)$, for $i \in \{1,2\}$. Consequently, $s_1 = s_2$. We next show $s_1' = s_2'$, thus $\lambda_1 = \lambda_2$.

For uniformity, we use s to denote s_1 and s_2. From λ_i, we obtain $\mathcal{T}_{\mathsf{f}} \models s \sqsubseteq \exists r.s_i'$ and s_i' is maximal with this property, for $i \in \{1,2\}$. Lemma 3.18 yields $\mathsf{tp}_{\mathcal{I}}(d_i) = s_i'$. Using the facts that $\mathcal{T}_{\mathsf{f}} \models s \sqsubseteq \exists r.s_i'$ for $i \in \{1,2\}$, $K \subseteq \mathsf{tp}_{\mathcal{I}}(d) = s$, $K' \subseteq \mathsf{tp}_{\mathcal{I}}(d_i) = s_i'$ for $i \in \{1,2\}$, and $\mathcal{T}_{\mathsf{f}} \models K \sqsubseteq (\leqslant 1\ r\ K')$, an easy semantic argument shows that $\mathcal{T}_{\mathsf{f}} \models s \sqsubseteq \exists r.(s_1' \cup s_2')$. The maximality of s_1' and s_2' thus implies $s_1' = s_2'$ as desired.

Hence, $\lambda_1 = \lambda_2$ and $(d, d_1), (d, d_2) \in \pi_{\lambda_1}$. Since π_{λ_1} is a bijection, we obtain $d_1 = d_2$, a contradiction.

We proof now prove the following auxiliary Lemma.

Lemma 3.18. *Let $\lambda = s\ {}^1\!\leftrightarrow_r^1 s'$ be with $s, s' \in P$ and $(d, d') \in \pi_\lambda$, then $\mathsf{tp}_{\mathcal{I}}(d) = s$ and $\mathsf{tp}_{\mathcal{I}}(d') = s'$.*

Proof. Let $\lambda = s\ {}^1\!\leftrightarrow_r^1 s'$ and $(d, d') \in \pi_\lambda$. We first show that $\mathsf{tp}_{\mathcal{I}}(d) = s$. Since d is in the domain of π_λ, we have $d \in \Delta_s$ or $d \in X_{\lambda,1}^{\mathcal{I}}$. In the former case, $\mathsf{tp}_{\mathcal{I}}(d) = s$ is immediate by construction of $\mathcal{I}$. Thus assume that $d \in X_{\lambda,1}^{\mathcal{I}}$. Then $s \subseteq \mathsf{tp}_{\mathcal{I}}(d)$. From λ, we obtain $\mathcal{T}_{\mathsf{f}} \models \mathsf{tp}_{\mathcal{I}}(d) \sqsubseteq \exists r.s'$. Let $\hat{s}' \supseteq s'$ be maximal such that $\mathcal{T}_{\mathsf{f}} \models \mathsf{tp}_{\mathcal{I}}(d) \sqsubseteq \exists r.\hat{s}'$. Note that, by λ, $s \subseteq \mathsf{tp}_{\mathcal{I}}(d)$ and $s' \subseteq \hat{s}'$ we have $\mathcal{T}_{\mathsf{f}} \models \hat{s}' \sqsubseteq (\leqslant 1\ r\ \mathsf{tp}_{\mathcal{I}}(d))$. Thus $\mathsf{tp}_{\mathcal{I}}(d) \rightarrow_r^1 \hat{s}'$.

Next, observe that $\hat{s}' \rightarrow_{r^-}^1 \mathsf{tp}_{\mathcal{I}}(d)$. If this was not the case, then (**c1**) would be applicable to d. Since (**c1**) applications are preferred over applications of (**c2**), such (**c1**) application to d would generate an $e \in \Delta^{\mathcal{I}}$ with $(d, e) \in r^{\mathcal{I}}$ and $e \in (\hat{s}')^{\mathcal{I}}$ before the (**c2**) application considered here. This contradicts the fact that $d \in X_{\lambda_1}^{\mathcal{I}}$, which implies that $d \notin (\exists r.s)^{\mathcal{I}}$ when (**c2**) was applied.

In summary, we have established that $\lambda' = \mathsf{tp}_{\mathcal{I}}(d)\ {}^1\!\leftrightarrow_r^1 \hat{s}'$ holds. Now assume to the contrary of what we have to show that $s \subsetneq \mathsf{tp}_{\mathcal{I}}(d)$. Recall that P is the type class that the current (**c2**) application treats, and that $s, s' \in P$. By λ', there is a type class P' with $\mathsf{tp}_{\mathcal{I}}(d), \hat{s}' \in P'$. Since $s \subsetneq \mathsf{tp}_{\mathcal{I}}(d)$, we have $P' \prec P$. Since $d \in X_{\lambda,1}^{\mathcal{I}}$, we had $d \notin (\exists r.s')^{\mathcal{I}}$ before the current rule application, thus also $d \notin (\exists r.\hat{s}')^{\mathcal{I}}$. Summing up, before the current rule application we had $\mathsf{tp}_{\mathcal{I}}(d), \hat{s}' \in P'$, $\lambda' = \mathsf{tp}_{\mathcal{I}}(d)\ {}^1\!\leftrightarrow_r^1 \hat{s}'$, $d \in \mathsf{tp}_{\mathcal{I}}(d)$, and $d \notin (\exists r.\hat{s}')^{\mathcal{I}}$. Consequently, rule (**c2**) was applicable also to type class P'. Since $P' \prec P$ and with the preference order that (**c2**) imposes on type classes, this contradicts that the current application is treating P.

It remains to show that $\mathsf{tp}_{\mathcal{I}}(d') = s'$. The argument is exactly the same as above, with r^- playing the role of r, s' playing the role of s and vice versa, λ^- playing the role of λ, and $\mathsf{tp}_{\mathcal{I}}(d')$ playing the role of $\mathsf{tp}_{\mathcal{I}}(d)$ and vice versa. □

(c3) preserves all invariants. Again by definition of the completion step the invariants (**i1**) and (**i2**) are preserved with each single application of (**c3**). We show that this is also the case for (**i3**). Assume that the step treated $d \in \Delta^{\mathcal{I}}$ with $\mathsf{tp}_{\mathcal{I}}(d) \rightarrow_r t$ and $\mathsf{tp}_{\mathcal{I}}(d) \not\rightarrow_r^1 t$, adding the edge (d, d_t) to $r^{\mathcal{I}}$. Since $\mathsf{tp}_{\mathcal{I}}(d_t) = t$ and $\mathsf{tp}_{\mathcal{I}}(d) \not\rightarrow_r^1 t$, there is no $K \sqsubseteq (\leqslant 1\ r^-\ K') \in \mathcal{T}_{\mathsf{f}}$ such that $K \subseteq t$ and $K' \subseteq \mathsf{tp}_{\mathcal{I}}(d)$. Take a $K \sqsubseteq (\leqslant 1\ r\ K') \in \mathcal{T}_{\mathsf{f}}$ with $K \subseteq \mathsf{tp}_{\mathcal{I}}(d)$ and $K' \subseteq t$. We

have to prove that there is no $e \in \Delta^{\mathcal{I}}$ distinct from d_t such that $(d, e) \in r^{\mathcal{I}}$ and $e \in K'^{\mathcal{I}}$. This can be done exactly as in the case of the completion rule (**c1**).

This concludes the proof of Lemma 3.17. □

Now we can move on to the task of showing that the constructed interpretation $\mathcal{I}$ is a model of $\mathcal{T}_{\mathsf{f}}$.

Lemma 3.19. *$\mathcal{I}$ is a model of $\mathcal{T}_{\mathsf{f}}$.*

Proof. We show that for every axiom $K \sqsubseteq C \in \mathcal{T}_{\mathsf{f}}$, we have that $\mathcal{I} \models K \sqsubseteq C$. We distinguish the following cases according to the structure of concept C:

- **case '$C = A$':** Let $d \in K^{\mathcal{I}}$. Then $K \subseteq \mathsf{tp}_{\mathcal{I}}(d)$ and by (**i1**) $\mathsf{tp}_{\mathcal{I}}(d) \in \mathsf{TP}(\mathcal{T}_{\mathsf{f}})$. Since $\mathcal{T}_{\mathsf{f}} \models K \sqsubseteq A$, this yields $A \in \mathsf{tp}_{\mathcal{I}}(d)$ and thus $d \in A^{\mathcal{I}}$.
- **case '$C = \bot$':** Follows from (**i1**) since for every $d \in \Delta^{\mathcal{I}}$, $\mathsf{tp}_{\mathcal{I}}(d) \in \mathsf{TP}(\mathcal{T}_{\mathsf{f}})$ $K^{\mathcal{I}} = \emptyset$.
- **case '$C = \exists r.K'$':** Let $d \in K^{\mathcal{I}}$. Then we have that $K \subseteq \mathsf{tp}_{\mathcal{I}}(d)$. Since $\mathcal{T}_{\mathsf{f}} \models K \sqsubseteq \exists r.K'$, we have that $\mathsf{tp}_{\mathcal{I}}(d) \to_r t'$ for some t' with $K' \subseteq t'$. It suffices to show that there is some d' with $(d, d') \in r^{\mathcal{I}}$ and $\mathsf{tp}_{\mathcal{I}}(d') = t'$. Note that one of the following cases must apply: (1) $\mathsf{tp}_{\mathcal{I}}(d) \to^1_r t'$ and $t' \not\to^1_{r^-} \mathsf{tp}_{\mathcal{I}}(d)$, (2) $\mathsf{tp}_{\mathcal{I}}(d) \to^1_r t'$ and $t' \to^1_{r^-} \mathsf{tp}_{\mathcal{I}}(d)$, and (3) $\mathsf{tp}_{\mathcal{I}}(d) \not\to^1_r t'$. These cases correspond exactly to the completion rules (**c1**) to (**c3**). Thus, one of these rules will add the required successor.
- **case '$C = \forall r.K'$':** Let $d \in K^{\mathcal{I}}$ and $(d, d') \in r^{\mathcal{I}}$, We have that $K \subseteq \mathsf{tp}_{\mathcal{I}}(d)$. Further, by (**i2**), we can distinguish the following cases:
  - $\mathsf{tp}_{\mathcal{I}}(d) \to_r \mathsf{tp}_{\mathcal{I}}(d')$. Then $\mathcal{T}_{\mathsf{f}} \models \mathsf{tp}_{\mathcal{I}}(d) \sqsubseteq \exists r.\mathsf{tp}_{\mathcal{I}}(d')$ and $\mathsf{tp}_{\mathcal{I}}(d')$ is maximal with this property. Since $\mathcal{T}_{\mathsf{f}} \models K \sqsubseteq \forall r.K'$, we have that $\mathcal{T}_{\mathsf{f}} \models \mathsf{tp}_{\mathcal{I}}(d) \sqsubseteq \exists r.\mathsf{tp}_{\mathcal{I}}(d') \cup K'$, and the maximality of $\mathsf{tp}_{\mathcal{I}}(d')$ yields $K' \subseteq \mathsf{tp}_{\mathcal{I}}(d')$, and thus $d' \in K'^{\mathcal{I}}$.
  - $\mathsf{tp}_{\mathcal{I}}(d') \to_{r^-} \mathsf{tp}_{\mathcal{I}}(d)$. Then we have $\mathcal{T}_{\mathsf{f}} \models \mathsf{tp}_{\mathcal{I}}(d') \sqsubseteq \exists r^-.\mathsf{tp}_{\mathcal{I}}(d)$. Together with $\mathcal{T}_{\mathsf{f}} \models K \sqsubseteq \forall r.K'$, we obtain $\mathcal{T}_{\mathsf{f}} \models \mathsf{tp}_{\mathcal{I}}(d') \sqsubseteq K'$. Since $\mathsf{tp}_{\mathcal{I}}(d') \in \mathsf{TP}(\mathcal{T}_{\mathsf{f}})$ by (**i1**), we obtain $K' \subseteq \mathsf{tp}_{\mathcal{I}}(d')$ and thus $d' \in K'^{\mathcal{I}}$.
- **case '$C = (\leqslant 1\ r\ K)'$':** Follows from (**i3**).

□

It now remains to show that $\mathcal{I}$ is finite. We will proceed by proving that the construction of the model $\mathcal{I}$ terminates. For this purpose, we associate a directed tree $T = (V, E)$ to the model $\mathcal{I}$ that makes explicit the way in which $\mathcal{I}$ is constructed. Recall that (**c1**) and (**c2**) are the only completion steps that introduce new domain elements and while (**c1**) introduces a single new element with each application, (**c2**) introduces a whole (finite) set of fresh elements. Moreover, the construction of $\mathcal{I}$ is defined in such a way that each application of a completion rule is *triggered* by a single domain element for which some $\exists$-requirement is not yet satisfied.

Definition 3.11. Let $\mathcal{I}$ be the interpretation constructed according to Definition 3.10. The *trace tree* $T = (V, E)$ *of* $\mathcal{I}$ is the labelled tree defined as follows:

- the domain of the initial interpretation $\Delta_0^{\mathcal{I}} \in V$, and is the *root node* of T;
- each $v \subseteq \Delta^{\mathcal{I}}$ introduced by a single application of either (**c1**) or (**c2**) is a node in V;
- $(v, v', 1) \in E$ iff v' is the singleton set introduced by an application of (**c1**) treating some $d \in v$; and
- $(v, v', 2) \in E$ iff v' is the set of elements introduced by an application of (**c2**) treating some $d \in v$.

An element d as above is called the *trigger* of v' and will be denoted by $\mathsf{tg}(v')$. △

Intuitively, the trace tree of the interpretation $\mathcal{I}$ makes explicit the growth of the domain of $\mathcal{I}$, and records the completion steps –(**c1**) or (**c2**)– responsible for such growth. From the definition of trace tree and a careful analysis of the completion steps we can make the following observation.

Observation 3. Let $T = (V, E)$ be the trace tree of $\mathcal{I}$. For each $(v_1, v_2, \ell), (v_2, v_3, \ell') \in E$ there are $d_1, \ldots, d_k \in \Delta^{\mathcal{I}}$ and roles $r_1, \ldots, r_{k-1}$ such that

1. $d_1 = \mathsf{tg}(v_2) \in v_1$ and $d_2, \ldots, d_k \in v_2$, with $d_k = \mathsf{tg}(v_2)$; and
2. $\mathsf{tp}_{\mathcal{I}}(d_i) \rightarrow^1_{r_i} \mathsf{tp}_{\mathcal{I}}(d_{i+1})$ for all $i < k$;
3. if $\ell = \ell'$ then $\ell = 1$.

Using the structural properties of T, we show that $\Delta^{\mathcal{I}}$ is finite. The intuition behind the proof is as follows: since each single application of either (**c1**) or (**c2**) introduces only a finite number of objects to the domain $\Delta^{\mathcal{I}}$, then each $v \in V$ is finite; and moreover, each such v has finitely many descendants. Hence, we only need to show that every branch in T has finite length to prove that V is finite (and thus $\Delta^{\mathcal{I}}$ is finite too). What we have just explained, it is actually a well-known result.

Lemma 3.20 (König's Lemma)**.** *Every finitely branching tree has infinitely many vertices iff it has at least one infinite simple path.*

Proposition 3.21. *Let $T = (V, E)$ be the trace of $\mathcal{I}$. Every simple path $v_0 \ldots, v_n$ in T has length at most $2|\mathsf{TP}(\mathcal{T}_{\mathsf{f}})|$.*

Proof. Assume towards a contradiction that there is a path in $v_0 \cdots v_n$ with v_0 the root of T and $n > 2|\mathsf{TP}(\mathcal{T}_\mathsf{f})|$. Let us consider the sequence of triggers associated to this path $\mathsf{tg}(v_1), \ldots, \mathsf{tg}(v_n)$. Since the number of triggers exceeds $2|\mathsf{TP}(\mathcal{T}_\mathsf{f})|$, there must be i, j with $1 \leq i < j \leq n$ and such that $\mathsf{tp}_\mathcal{I}(\mathsf{tg}(v_i)) = \mathsf{tp}_\mathcal{I}(\mathsf{tg}(v_j))$ and $j > i+1$. Using repeatedly the arguments from Observation 3 along the path $v_{i-1} \ldots v_j$, we obtain that there is a sequence of domain elements $d_0, \ldots, d_k$ and roles $r_0, \ldots, r_{k-1}$ such that

1. $d_0 = \mathsf{tg}(v_i) \in v_{i-1}$, $d_1 \in v_i$, and $d_k = \mathsf{tg}(v_j) \in v_{j-1}$;
2. $\mathsf{tp}_\mathcal{I}(d_\ell) \rightarrow^1_{r_\ell} \mathsf{tp}_\mathcal{I}(d_{\ell+1})$ for $\ell < k$.
3. $d_0, \ldots, d_k$ contains all elements $\mathsf{tg}(v_i), \mathsf{tg}(v_{i+1}) \ldots, \mathsf{tg}(v_j)$;

From $\mathsf{tp}_\mathcal{I}(\mathsf{tg}(v_i)) = \mathsf{tp}_\mathcal{I}(\mathsf{tg}(v_j))$ and Point 2 above we can conclude that

$$\mathsf{tp}_\mathcal{I}(d_0), r_0, \ldots, r_{k-1}, \mathsf{tp}_\mathcal{I}(d_k)$$

is a finmod cycle in $\mathcal{T}_\mathsf{f}$. Since all finmod cycles in $\mathcal{T}_\mathsf{f}$ have been reversed, we have

$$\mathsf{tp}_\mathcal{I}(d_0) \; {}^1\!\leftrightarrow^1_{r_0} \mathsf{tp}_\mathcal{I}(d_1) \; {}^1\!\leftrightarrow^1_{r_1} \cdots {}^1\!\leftrightarrow^1_{r_{k-1}} \mathsf{tp}_\mathcal{I}(d_k). \tag{3.22}$$

Since $d_0 = \mathsf{tg}(v_i) \in v_{i-1}$ and $d_1 \in v_i$, d_1 was generated by the application of a completion step treating d_0. We claim that, this completion rule must be (**c2**). Indeed, assume set v_i was introduced by an application of (**c1**) then $v_i = \{d_1\}$, and $\mathcal{T}_\mathsf{f}$ entails that $\mathsf{tp}_\mathcal{I}(d_0) \rightarrow^1_s \mathsf{tp}_\mathcal{I}(d_1)$ for some role s. From the latter and (3.22) we get

$$\mathsf{tp}_\mathcal{I}(d_0) \rightarrow^1_s \mathsf{tp}_\mathcal{I}(d_1) \rightarrow^1_{r_0^-} \mathsf{tp}_\mathcal{I}(d_0)$$

which then would imply that there is a finmod cyle whose reversal ensures $\mathsf{tp}_\mathcal{I}(d_1) \rightarrow^1_{s^-} \mathsf{tp}_\mathcal{I}(d_0)$ holds. This contradicts the assumption that d_1 was introduced by an application of (**c1**).

From (3.22), and point 3 in Observation 3 we can conclude that all elements $d_1, \ldots, d_k$ were introduced by the same application of (**c2**). Now, observe that we have $d_1 \in v_i$ and $d_k \in v_{j-1}$ and $j > i+1$, $v_i \neq v_{j-1}$, which implies that d_1 and d_k were introduced by two different completion steps, hence a contradiction. □

Proposition 3.21 give us the missing argument for proving the desired result.

Lemma 3.22. *$\Delta^\mathcal{I}$ is finite.*

We are now in the position to prove the main result of this section.

Proof of Theorem 3.15. Let $\mathcal{T}$ be a Horn-$\mathcal{ALCFI}$ TBox, and $\mathcal{T}_\mathsf{f} = \mathsf{finClosure}(\mathcal{T})$.

For the proof of Theorem 3.15, we show the equivalent statement:

($\star$) if $\mathcal{T}_\mathsf{f} \not\models K \sqsubseteq C$, then $\mathcal{T} \not\models_\mathsf{fin} K \sqsubseteq C$.

Assume that $\mathcal{T}_\mathsf{f} \not\models K \sqsubseteq C$. Then, there is a model $\mathcal{J}$ of $\mathcal{T}_\mathsf{f}$ such that there is some $d \in \Delta^\mathcal{J}$ with $d \in K^\mathcal{J}$ and $d \notin C^\mathcal{J}$. Let $t = \mathsf{tp}_\mathcal{J}(d)$, note that by definition 3.7 $t \in \mathsf{TP}(\mathcal{T}_\mathsf{f})$. Now, let $\mathcal{I}$ be the interpretation constructed in Definition 3.10. By (3.21), there exists $d' \in \Delta^\mathcal{I}$ such that $\mathsf{tp}_\mathcal{I}(d') = t$. Further, by lemmata 3.19 and 3.22, $\mathcal{I}$ is a finite model of $\mathcal{T}_\mathsf{f}$, and since $\mathcal{T} \subseteq \mathcal{T}_\mathsf{f}$, $\mathcal{I} \models \mathcal{T}$. By choice we have that $d \in K^\mathcal{I}$ and $d' \notin C^\mathcal{I}$, and consequently $\mathcal{I}$ witnesses $\mathcal{T} \not\models_\mathsf{fin} K \sqsubseteq C$. □

It is worth noticing that although the cycle reversion for Horn-$\mathcal{ALCFI}$ might result in an exponential blow-up of the input TBox, which would lead to a double exponential procedure for finite model reasoning on Horn-$\mathcal{ALCFI}$. In Chapter 4 we will show that the cycle reversion technique can be optimized for providing computationally optimal procedure. Recall that an ExpTime upper bound follows already from the complexity of finite model reasoning in $\mathcal{ALCFI}$.

3.4 Finite Model Reasoning in Horn-$\mathcal{ALCFI}$

In this section, we will show how the cycle reversion technique in the previous section and the construction of finite models can be extended to show that finite model reasoning on Horn-$\mathcal{ALCQI}$ can be reduced to unrestricted reasoning. In fact, an easy consequence of Theorem 3.15 and Theorem 3.14 is the following.

Corollary 3.23. *Finite model subsumption w.r.t. a Horn-$\mathcal{ALCFI}$ TBox $\mathcal{T}$ can be reduced to subsumption w.r.t.* $\mathsf{finClosure}(\mathcal{T})$.

Furthermore, using the construction of finite models in Section 3.3.1 we can show that finite model reasoning w.r.t. a TBox $\mathcal{T}$ is equivalent to reasoning (on unrestricted models) w.r.t. $\mathcal{T}_\mathsf{f}$. Since all (finite) TBox reasoning tasks are reducible to (finite) concept satisfiability it suffices to show the following.

Theorem 3.24. *Let $\mathcal{T}$ be a Horn-$\mathcal{ALCFI}$ TBox, and A a concept name.*

A is finitely satisfiable w.r.t. $\mathcal{T}$ iff A is satisfiable w.r.t. $\mathcal{T}_\mathsf{f}$.

Proof.

(⇒) Assume that A is finitely satisfiable w.r.t. $\mathcal{T}$. We have by Theorem 3.14 that all finite models of $\mathcal{T}$ are models of $\mathcal{T}_\mathsf{f}$ since all the added axioms are entailed by $\mathcal{T}$ in finite models, the it follows that A is satisfiable w.r.t. $\mathcal{T}_\mathsf{f}$.

(⇐) Assume that A is satisfiable w.r.t. $\mathcal{T}_\mathsf{f}$. Then, there is a type t for $\mathcal{T}_\mathsf{f}$ with $A \in t$, that is realized in the interpretation $\mathcal{I}$ constructed as in Definition 3.10, i.e., there is some $d \in \Delta^\mathcal{I}$ with $\mathsf{tp}_\mathcal{I} = t$, and moreover, $d \in A^\mathcal{I}$. Since $\mathcal{T} \subseteq \mathcal{T}_\mathsf{f}$ we get that $\mathcal{I}$ is also a *finite* model of $\mathcal{T}$ by Lemmata 3.22 and 3.19. Hence, A is finitely satisfiable w.r.t. $\mathcal{T}$.

□

We cannot obtain directly the same result for reasoning tasks involving the ABox, namely, for consistency and instance checking. However, we can modify the construction from Definition 3.10 as to consider also an ABox. More precisely, let $(\mathcal{T}, \mathcal{A})$ be a Horn $\mathcal{ALCFI}$ ontology. Further, assume that $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}_\mathsf{f}$. Our aim is to construct a finite model $\mathcal{J}$ of $\mathcal{A}$ (and thus also of $\mathcal{T}$).

Definition 3.12 (finite ABox completion)**.** The interpretation $\mathcal{J}$ is defined starting from the interpretation $\mathcal{J}_0$ whose domain $\Delta^{\mathcal{J}_0}$ contains one element for every ABox individual, and an element d_t for each $t \in \mathsf{TP}(\mathcal{T}_\mathsf{f})$. More precisely, we set

$$\begin{aligned} \Delta^{\mathcal{J}_0} &= \mathsf{Ind}(\mathcal{A}) \cup \{\ d_t \mid t \in \mathsf{TP}(\mathcal{T}_\mathsf{f})\ \} \\ A^{\mathcal{J}_0} &= \{\ a \in \mathsf{Ind}(\mathcal{A}) \mid A \in \mathsf{tp}_\mathcal{A}(a)\ \} \cup \{d_t \mid A \in t\ \} \\ r^{\mathcal{J}_0} &= \{\ (a,b) \mid r(a,b) \in \mathcal{A}\ \} \end{aligned} \tag{3.23}$$

where $\mathsf{tp}_\mathcal{A}(a) := \{A \in \mathsf{CN}(\mathcal{T}) \mid \mathcal{A}, \mathcal{T}_\mathsf{f} \models A(a)\}$.

$\mathcal{J}$ is then the result obtained from *completing* $\mathcal{J}_0$ using the same completion steps (**c1**), (**c2**), and (**c3**) from Definition 3.10 respecting the priority of the step applications. Recall that application of (**c1**) has priority over applications of (**c2**). △

In order to to show that the construction of $\mathcal{J}$ is well-defined and yields a finite model of $\mathcal{A}$ and $\mathcal{T}_\mathsf{f}$, we proceed as in Section 3.3.1. However, we need to make some adjustments to the arguments. In particular, we cannot ensure that $\mathcal{J}$ satisfies invariant (**i2**). Indeed, for an assertion $r(a,b) \in \mathcal{A}$ it might be the case that neither $\mathsf{tp}_\mathcal{J}(a) \to_r \mathsf{tp}_\mathcal{I}(b)$ nor $\mathsf{tp}_\mathcal{J}(b) \to_{r^-} \mathsf{tp}_\mathcal{J}(a)$. Still, we can prove that the construction of $\mathcal{J}$ satisfies the following invariant.

(**i2′**) if $(d, d') \in r^\mathcal{I} \setminus (\mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}))$, then we have $\mathsf{tp}_\mathcal{I}(d) \to_r \mathsf{tp}_\mathcal{I}(d')$ or $\mathsf{tp}_\mathcal{I}(d') \to_{r^-} \mathsf{tp}_\mathcal{I}(d)$.

With that adjustment in place, one can prove that the construction of $\mathcal{J}$ satisfies the invariants (**i1**), (**i2′**) and (**i3**). The initial interpretation $\mathcal{J}_0$ satisfies all the invariants. Indeed, (**i1**) is trivially satisfied by the definition in (3.23). Further, since $r^{\mathcal{J}_0} \setminus (\mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A})) = \emptyset$, (**i2′**) is satisfied too. Moreover, given that $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}_\mathsf{f}$, the standard name assumption ensures that (**i3**) is satisfied. To show that each of the rules (**c1**) to (**c3**) preserves the invariants the exact same arguments as in the proof of Lemma 3.17 apply. Hence we have the following.

Proposition 3.25. *Each application of a completion step during the construction of $\mathcal{J}$ preserves the invariants* (**i1**)*,* (**i2′**) *and* (**i3**)*.*

Using Proposition 3.25, we can now show that $\mathcal{J}$ is a model of $\mathcal{A}$ and $\mathcal{T}_\mathsf{f}$.

Proposition 3.26. *$\mathcal{J}$ is a model of $\mathcal{A}$ and $\mathcal{T}_\mathsf{f}$.*

Proof. The proof that $\mathcal{J} \models \mathcal{T}_\mathsf{f}$ is the same as in the proof of Lemma 3.19, using Proposition 3.25. It remains to show that for every assertion $\alpha \in \mathcal{A}$, $\mathcal{J} \models \alpha$. This is a consequence of the definition of $\mathcal{J}$. Indeed, for every individual a, if $\alpha = A(a) \in \mathcal{A}$, then $A \in \mathsf{tp}_\mathcal{A}(a)$ which by the definition of $\mathcal{J}$ implies that $a \in A^\mathcal{I}$. Further, if $\alpha = r(a,b) \in \mathcal{A}$ then $(a,b) \in r^\mathcal{J}$. □

Lastly, we need to show that $\mathcal{J}$ is finite, but this follows directly from Proposition 3.21.

Proposition 3.27. *$\Delta^\mathcal{J}$ is finite*

We now have all the ingredients to show the main result of this section.

Theorem 3.28. *Let $\mathcal{T}$ be a Horn-$\mathcal{ALCFI}$ TBox and $\mathcal{A}$ an ABox. Then $\mathcal{A}$ is finitely consistent w.r.t. $\mathcal{T}$ iff $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}_\mathsf{f}$.*

Proof.

(⇒) Assume that $\mathcal{A}$ is finitely consistent w.r.t. $\mathcal{T}$. We need to show that $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}_\mathsf{f}$. This is a consequence of the observation that all CIs in $\mathcal{T}_\mathsf{f} \setminus \mathcal{T}$ are entailed by the original TBox in finite models, as provided by Theorem 3.14.

(⇐) Assume that $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}_\mathsf{f}$. Since $\mathcal{T} \subseteq \mathcal{T}_\mathsf{f}$, we get that the interpretation $\mathcal{J}$ constructed above is also a *finite* model of $\mathcal{A}$ and $\mathcal{T}$ by Propositions 3.27 and 3.26. Hence, $\mathcal{A}$ is finitely satisfiable w.r.t. $\mathcal{T}$.

□

3.5 From Horn-$\mathcal{ALCFI}$ to Horn-$\mathcal{ALCQI}$

We now show that our results for finite satisfiability and finite subsumption, i.e., the reasoning tasks that do not involve an ABox extend straightforwardly from Horn-$\mathcal{ALCFI}$ to Horn-$\mathcal{ALCQI}$. In particular, we can convert a Horn-$\mathcal{ALCQI}$ TBox $\mathcal{T}$ into a Horn-$\mathcal{ALCFI}$ TBox $\mathcal{T}'$ such that finite (un)satisfiability is preserved by replacing each CI $K \sqsubseteq (\geqslant n\ r\ K')$ in $\mathcal{T}$ with the following inclusions, for $1 \leq i < j \leq n$:

$$K \sqsubseteq \exists r.B_i, \quad B_i \sqsubseteq K', \quad B_i \sqcap B_j \sqsubseteq \bot \tag{3.24}$$

While an easy unraveling argument can be used to prove that this reduction is correct in the presence of infinite models, more care is required in the finite case.

Proposition 3.29. *$\mathcal{T}$ is finitely satisfiable iff $\mathcal{T}'$ is finitely satisfiable.*

Proof. The "$\Leftarrow$" direction is trivial since every model of $\mathcal{T}'$ is also a model of $\mathcal{T}$. For the "$\Rightarrow$" direction, let $\mathcal{J}$ be a finite model of $\mathcal{T}$. We construct a finite model $\widehat{\mathcal{J}}$of $\mathcal{T}'$ by taking n copies of $\mathcal{J}$ and 'rewiring' all role edges across the different copies such that the concept names B_i can be interpreted in a non-conflicting way.

Specifically, since $\mathcal{J}$ satisfies $K \sqsubseteq (\geqslant n\ r\ K')$, we can choose a function

$$\mathsf{succ} : K^{\mathcal{J}} \times \{0, \dots, n-1\} \to \Delta^{\mathcal{J}}$$

such that the following conditions are satisfied:

- for all $d \in K^{\mathcal{J}}$ and $i < n$: $(d, \mathsf{succ}(d,i)) \in r^{\mathcal{J}}$ and $\mathsf{succ}(d,i) \in (K')^{\mathcal{J}}$;
- for all $d \in K^{\mathcal{J}}$ and $i < j < n$: $\mathsf{succ}(d,i) \neq \mathsf{succ}(d,j)$.

Then define the desired interpretation $\widehat{\mathcal{J}}$ by setting

$$\begin{aligned}
\Delta^{\widehat{\mathcal{J}}} &= \{d_i \mid d \in \Delta^{\mathcal{J}} \text{ and } i < n\} \\
E^{\widehat{\mathcal{J}}} &= \{d_i \mid d \in E^{\mathcal{J}} \text{ and } i < n\}, \quad \text{for all } E \in \mathsf{N_C} \setminus \{B_0, \dots, B_{n-1}\} \\
B_i^{\widehat{\mathcal{J}}} &= \{d_i \mid d \in \Delta^{\mathcal{J}}\} \text{ for all } i < n \\
s^{\widehat{\mathcal{J}}} &= \{(d_i, e_i) \mid (d,e) \in s^{\mathcal{J}} \text{ and } i < n\}, \quad \text{for all } s \in \mathsf{N_R} \setminus \{r\} \\
r^{\widehat{\mathcal{J}}} &= \{(d_i, e_i) \mid (d,e) \in r^{\mathcal{J}},\ i < n, \text{ and } d \notin K^{\mathcal{J}} \text{ or } e \neq \mathsf{succ}(d,j) \text{ for any } j\} \\
&\quad \cup \{(d_i, e_{(i+j) \bmod n}) \mid (d,e) \in r^{\mathcal{J}},\ i,j < n, e = \mathsf{succ}(d,j)\}
\end{aligned}$$

It remains to verify that $\mathcal{J}$ is indeed a model of $\mathcal{T}'$. Clearly, the CIs in (3.24) are satisfied. To verify that all concept inclusions in $\mathcal{T}$ are satisfied by $\mathcal{J}$, we observe that the construction ensures that the number of r-successors (and -predecessors) in any $A \in \mathsf{CN}$ of every (x,i) is the same as that for x.

We first claim that, for every $d \in \Delta^{\mathcal{J}}$ and every s-successor e of d in $\mathcal{J}$, the i-th copy of d in $\widehat{\mathcal{J}}$ has exactly one copy of e as an s-successor:

Claim 1. Let s be a role, $d_i \in \Delta^{\mathcal{J}}$, and let $\{e \in \Delta^{\mathcal{J}} \mid (d,e) \in s^{\mathcal{J}}\} = \{e_1, \dots, e_\ell\}$ for some $\ell \geq 0$. Then $\{e^j \in \Delta^{\mathcal{J}} \mid (d^i, e^j) \in s^{\mathcal{J}}\} = \{e_1^{j_1}, \dots, e_\ell^{j_\ell}\}$, for some $j_1, \dots, j_\ell \in \{0, \dots, n-1\}$.

This claim is implied by the construction of $s^{\mathcal{J}}$: consider a given $d_i \in \Delta^{\mathcal{J}}$ and (possibly inverse) role s. If s is neither r nor r^-, then every e_k contributes exactly one s-successor e_k^i of d^i. The same holds if $s = r$ and $d \notin K^{\mathcal{J}}$. If $s = r$ and $d \in K^{\mathcal{J}}$, then each $e_k = \mathsf{succ}(d,j)$ for some j contributes exactly one s-successor $e_k^{(i+j) \bmod n}$ of d^i, and every other e_k contributes e_k^i. For $s = r^-$, then every $e_k \in K^{\mathcal{J}}$ with $d = \mathsf{succ}(e_k, j)$ for some j contributes $e_k^{(i-j) \bmod n}$, and every other e_k contributes e_k^i.

As an immediate consequence, we obtain that all qualified and unqualified number restrictions in $d \in \Delta^{\mathcal{J}}$ are preserved in every $d^i \in \Delta^{\mathcal{J}}$:

Fact. *Let $d^i \in \Delta^{\mathcal{J}}$ and $D = (\bowtie s\ n\ C)$ where $\bowtie \in \{\leqslant, \geqslant\}$, s is a role or inverse role, and C is either a conjunction of concept names, or the negation of such a conjunction, or $\top$, or $\bot$. Then $d \in D^{\mathcal{J}}$ iff $d^i \in D^{\widehat{\mathcal{J}}}$.*

This can be concluded from the previous claim and the observation that e and e^{j_i} satisfy the same concept names. The fact includes the cases $s = r$ and $s = r^-$, and it implies that existential, and universal restrictions are preserved – for the latter it is necessary to allow that C is a negated conjunction.

We are now ready to prove that $\widehat{\mathcal{J}}$ is a model of $\mathcal{T}'$, proceeding by different types of CIs. We distinguish the following cases.

- $L \sqsubseteq A$ and $L \sqsubseteq \bot$, both in $\mathcal{T}$. These are satisfied because they are satisfied by $\mathcal{J}$ and due to the construction: every d in $\mathcal{J}$ and every d^i in $\widehat{\mathcal{J}}$ are instances of the same non-B_i concept names.
- $L \sqsubseteq \exists s.L'$ in $\mathcal{T}$. Let $d^i \in L^{\mathcal{J}}$. Then $d \in L^{\mathcal{J}}$ due to the construction. Since $\mathcal{J}$ satisfies the axiom, $d \in (\geqslant 1\ s\ L')^{\mathcal{J}}$. With the previous fact, we conclude $d^i \in (\geqslant 1\ s\ L')^{\mathcal{J}}$, hence $d^i \in (\exists s.L')^{\mathcal{J}}$. This argument includes the cases $s = r$ and $s = r^-$.
- $L \sqsubseteq \forall s.L'$ in $\mathcal{T}$. In the argument above, replace "$\in (\geqslant 1\ s\ L')^{\cdots}$" with "$\notin (\geqslant 1\ s\ \neg L')^{\cdots}$".
- $L \sqsubseteq (\leqslant 1\ s\ L')$ in $\mathcal{T}$. Then $d^i \in L^{\mathcal{J}}$ implies $d \in L^{\mathcal{J}}$, hence $d \in (\leqslant 1\ s\ L')^{\mathcal{J}}$ and, due to the previous fact, $d^i \in (\leqslant 1\ s\ L')^{\mathcal{J}}$.
- $L \sqsubseteq (\geqslant m\ s\ L')$ in $\mathcal{T}$. Apply the same argument as above.
- $B_i \sqsubseteq K'$ and $B_i \sqcap B_j \sqsubseteq \bot$. Follows from the construction.
- $K \sqsubseteq \exists r.B_i$. Let $d^j \in K^{\mathcal{J}}$, which implies $d \in K^{\mathcal{J}}$.
  Let $e = \mathsf{succ}(d, (i-j) \bmod n)$. Then the construction yields that $(d^j, e^i) \in r^{\mathcal{J}}$ — because $i = (j + (i-j) \bmod n) \bmod n$ — and $e^i \in B_i^{\mathcal{J}}$. Hence $d^j \in (\exists r.B_i)^{\mathcal{J}}$.

□

It follows from Proposition 3.29 and Theorem 3.24 that a Horn-$\mathcal{ALCQI}$ TBox $\mathcal{T}$ is finitely satisfiable if and only if $\mathcal{T}'_{\mathsf{f}}$ is satisfiable.

Conclusions

In this Chapter, we have shown that it is possible to extend the cycle reversion technique introduced for *DL-Lite*$^{\mathcal{F}}_{core}$ by Rosati [117] to more expressive Horn DLs. As described at the beginning of the chapter, the cycle reversion technique amounts to add axioms to the TBox of an ontology in order to axiomatize the entailments w.r.t. finite models of the ontology. Such an axiomatization allows then to reduce *finite* model reasoning to reasoning w.r.t. *unrestricted* reasoning. In particular, we have shown that for *DL-Lite*$^{\mathcal{F}}_{Horn}$ the simple treatment of cycles

involving only concepts describing domain and range restrictions for roles (Definition 3.5) it is enough for providing an axiomatization of finite model entailment w.r.t. *DL-Lite*$^{\mathcal{F}}_{Horn}$ TBoxes (Lemma 3.9). Moreover, to show that the axiomatization given by $\mathcal{T}_{\mathsf{f}}$ is complete for a *DL-Lite*$^{\mathcal{F}}_{Horn}$ TBox $\mathcal{T}$, we provide an explicit construction of a finite model of $\mathcal{T}_{\mathsf{f}}$. The reduction of finite model (TBox) reasoning in *DL-Lite*$^{\mathcal{F}}_{Horn}$ to unrestricted reasoning also shows that the former task is no more difficult than the latter –recall that computing $\mathcal{T}_{\mathsf{f}}$ can be done in polynomial time on the size of $\mathcal{T}$ (Lemma 3.7).

On the other hand, for the more expressive Horn-$\mathcal{ALCFI}$, extending the cycle reversion technique requires a more careful analysis. Indeed, due to the presence of qualified number restrictions and universal restrictions, cycles involve more complex concepts than the simple domain and range restrictions for roles. We 'capture' these concepts using the notion of types for a Horn-$\mathcal{ALCFI}$ TBox $\mathcal{T}$, which are essentially satisfiable concepts w.r.t. $\mathcal{T}$. Using this more sophisticated notion of cycles involving types, we show that reversing those cycles provides a complete axiomatization, $\mathcal{T}_{\mathsf{f}}$, of finite model entailment w.r.t. Horn-$\mathcal{ALCFI}$ TBoxes (Theorem 3.15). The latter requires a careful construction of a finite model of $\mathcal{T}_{\mathsf{f}}$. The major difficulty to realize the latter construction is the interaction between different finmod-cycles, which is not present in the less expressive *DL-Lite*$^{\mathcal{F}}_{Horn}$. Using the axiomatization provided by $\mathcal{T}_{\mathsf{f}}$, we can also reduce finite model reasoning in Horn-$\mathcal{ALCFI}$ to unrestricted reasoning in Horn-$\mathcal{ALCFI}$. Although this reduction does not give interesting complexity boundaries for finite model reasoning in Horn-$\mathcal{ALCFI}$, it gives a more suitable basis of efficient implementation than the techniques for full $\mathcal{ALCFI}$ based on systems of inequalities. Indeed, recall that an upper bound for combined complexity of finite model reasoning on Horn-$\mathcal{ALCFI}$ follows already from the complexity of finite model reasoning in $\mathcal{ALCFI}$, which is ExpTime-complete. Further, note that the definition of cycle-reversion may cause an exponential blow-up on the size of $\mathcal{T}_{\mathsf{f}}$ w.r.t. the size of the original TBox $\mathcal{T}$. In Chapter 4, we will show such an exponential blow-up on $\mathcal{T}_{\mathsf{f}}$ can be avoided by reversing only *relevant* finmod-cycles in $\mathcal{T}$.

Finally, we have shown that the standard reduction for unrestricted models to eliminate qualified arbitrary 'at-least' number restrictions from a Horn-$\mathcal{ALCQI}$ TBox also holds in finite models. Hence extending all the results for Horn-$\mathcal{ALCFI}$ to Horn-$\mathcal{ALCQI}$.

Chapter

4 Consequence Driven Finite Model Reasoning on Horn-$\mathcal{ALCFI}$

We have shown in Chapter 3 that the axiomatization provided by cycle-reversion yields a reduction of finite model reasoning in Horn-$\mathcal{ALCFI}$ to arbitrary model reasoning in Horn-$\mathcal{ALCFI}$. However, in the worst-case reversing *all* the cycles entailed blows up the TBox exponentially. The latter is a clear indication that from the algorithmic view point cycle-reversion (as defined in Section 3.3) is not well suited for a direct implementation.

In this section, we build on the results of Chapter 3 to devise a *consequence-driven* procedure for realizing finite model reasoning in Horn-$\mathcal{ALCFI}$. Consequence-driven procedures, such as CEL, CB, and ELK [14, 79, 80], underly modern and highly efficient reasoners for Horn-DLs. While traditional reasoning procedures in DLs based on tableaux algorithms aim to build a model of a given ontology that violates a subsumption test, consequence-driven ones derive subsumptions relations explicitly using inference rules. The main advantage of using such procedures is that subsumption relations are computed 'all at once' in a goal-directed fashion, instead of enumerating 'all possible' pairs of concepts and building counter-models. The latter approach for performing reasoning was first proposed for the description logic $\mathcal{EL}^{++}$ [13], and later extended to the Horn-$\mathcal{SHIQ}$ description logic [79], which extends Horn-$\mathcal{ALCQI}$ with *transitive roles* ($\mathcal{S}$) and *role hierarchies* ($\mathcal{H}$). The consequence-driven procedures developed for these Horn DLs work by 'saturating' (or completing) the input axioms by exhaustively applying a set of rules. These procedures are purely syntactic, and for that reason it is usually the case that the input axioms are considered to be in certain *normal form*.

We present first a set of rules for saturating a Horn-$\mathcal{ALCFI}$ TBox $\mathcal{T}$ and deriving all subsumptions holding in finite models of $\mathcal{T}$. In Section 4.3, we then expand them to finite ABox consistency and thus to finite instance checking.

R1 $\dfrac{}{K \sqcap A \sqsubseteq A}$ **R2** $\dfrac{}{K \sqsubseteq \top}$

R3 $\dfrac{K \sqsubseteq A_i \quad \bigsqcap A_i \sqsubseteq C}{K \sqsubseteq C}$ **R4** $\dfrac{K \sqsubseteq \exists r.K' \quad K' \sqsubseteq \forall r^-.A}{K \sqsubseteq A}$

R5 $\dfrac{K \sqsubseteq \exists r.K' \quad K \sqsubseteq \forall r.A}{K \sqsubseteq \exists r.(K' \sqcap A)}$ **R6** $\dfrac{K \sqsubseteq \exists r.K' \quad K' \sqsubseteq \bot}{K \sqsubseteq \bot}$

R7 $\dfrac{\begin{array}{lll} K \sqsubseteq \exists r.K_1 & K \sqsubseteq \exists r.K_2 & K_1 \sqsubseteq A \\ K \sqsubseteq (\leqslant 1\ r\ A) & & K_2 \sqsubseteq A \end{array}}{K \sqsubseteq \exists r.(K_1 \sqcap K_2)}$

R8 $\dfrac{\begin{array}{lll} K \sqsubseteq \exists r.K' & K' \sqsubseteq \exists r^-.K_1 & K \sqsubseteq A \\ K' \sqsubseteq (\leqslant 1\ r^-\ A) & & K_1 \sqsubseteq A \end{array}}{K \sqsubseteq A_1 \quad \text{for any } A_1 \in K_1}$

Table 4.1: Inference rules for Horn-$\mathcal{ALCFI}$

4.1 The Inference Rules

Our procedure starts with a given Horn-$\mathcal{ALCFI}$ TBox $\mathcal{T}$ which is in a slightly stricter normal form than the one introduced in Chapter 2, and then exhaustively applies a set of inference rules. More precisely, we consider axioms of the form

$$K \sqsubseteq \forall r.K' \qquad \text{and} \qquad K \sqsubseteq (\leqslant 1\ r\ K'), \tag{4.1}$$

where K' must be a concept name A. Recall that K stands for a conjunction of concept names, and that we regard such conjunctions as sets of concept names.

The inference rules displayed in Table 4.1 are minor variations of the corresponding rules in the calculus presented by Kazakov [79][1]. The main difference of our rules with respect to those by Kazakov is that our language does not include role hierarchies.

Observe that, rules **R1**-**R8** preserve the normal form in (4.1), and are applied in the sense that, if the concept inclusions in the precondition (above the line) are already present in the partially completed TBox, then those in the postcondition (below the line) are added to such TBox. Moreover, note that Rule **R1** is applied only if $K \sqcap A$ *occurs* in the current (partially completed) TBox, that is, there is a CI of the form $K \sqcap A \sqsubseteq C$ or $K' \sqsubseteq \exists r.(K \sqcap A)$. The same is true for rule **R2** with K in place of $K \sqcap A$.

We introduce an additional rule **R9** which performs a *syntactical cycle reversion*. Note that only the 'first edge' of each cycle is reversed, and that this is sufficient because the cycle can

[1] Indeed, Kazakov [79] presents a calculus for the more expressive DL Horn-$\mathcal{SHIQ}$

be rotated to make any edge the 'first' one. In the formulation of this rule, $\oplus_n$ means addition modulo n.

(syntactical cycle-reversion)

$$\textbf{R9}\quad \frac{\begin{array}{cc} K_i \sqsubseteq \exists r_i.K_{i\oplus_n 1} & \\ K_{i\oplus_n 1} \sqsubseteq (\leqslant 1\ r_i^-\ A_i) & K_i \sqsubseteq A_i \qquad 0 \geq i < n \end{array}}{K_1 \sqsubseteq \exists r_0^-.K_0 \qquad K_0 \sqsubseteq (\leqslant 1\ r_0\ A_1)}$$

Let us consider the following example to illustrate the application of rules **R1**–**R9** to a TBox.

Example 9. Consider the TBox $\mathcal{T}_1$ consisting of the following axioms:

$$A \sqsubseteq \exists r.(A \sqcap A_1 \sqcap \cdots \sqcap A_n), \tag{4.2}$$
$$A \sqsubseteq (\leqslant 1\ r^-\ A). \tag{4.3}$$

cycle-reversion from Section 3.3 reverses all of the exponentially many cycles K, r, K with $K \subseteq S := \{A, A_1, \ldots, A_n\}$ and $A \in K$, adding $K \sqsubseteq \exists r^-.K$ and $K \sqsubseteq (\leqslant 1\ r\ K)$ for all such K. In contrast, the calculus avoids introducing 'irrelevant' conjunctions K and instead jointly reverses all these cycles by generating $A \sqsubseteq \exists r^-.S$ and $A \sqsubseteq (\leqslant 1\ r\ A)$:

$S \sqsubseteq A$		from **R1**	(4.4)
$A \sqsubseteq A$		from **R1**	(4.5)
$S \sqsubseteq \exists r.S$		from (4.2), (4.4), **R3**	(4.6)
$S \sqsubseteq (\leqslant 1\ r^-\ A)$		from (4.3), (4.4), **R3**	(4.7)
$S \sqsubseteq \exists r^-.S$		and	(4.8)
$S \sqsubseteq (\leqslant 1\ r\ A)$		from (4.4), (4.6), (4.7), **R9**	(4.9)
$A \sqsubseteq A_i$		from (4.2), (4.4), (4.5), (4.7), (4.8), **R8**	(4.10)
$A \sqsubseteq \exists r^-.S$		from (4.8), (4.10), **R3**	(4.11)
$A \sqsubseteq (\leqslant 1\ r\ A)$		from (4.9), (4.10), **R3**	(4.12)

$\overline{\wedge}$

Note that avoiding to introduce 'irrelevant' conjunctions K, as illustrated by Example 9, is a main feature of consequence-based procedures which enables the excellent practical performance typically observed for this class of calculi.

The algorithm terminates after at most exponentially many rule applications since there are only exponentially many different concept inclusions that use the concept and role names of the original TBox. This of course because the TBox is in normal form, and modulo (simple) equivalence (recall that we regard conjunctions as sets of concept names). Each rule application

can be performed in polynomial time, which is easy to see for the rules **R1**–**R8**. For **R9**, the crucial observation is that it suffices to consider all conjunctions K_0, K_1 and to check whether they are involved in *any* cycle. The latter can easily be done by a variation of directed graph reachability, where the nodes of the graph are the conjunctions that occur in the current TBox and the edges come from inclusions $K \sqsubseteq \exists r.K'$.

4.2 Soundness and Completeness

In this section, we show that our calculus is sound and complete for concept satisfiability. The following theorem states this result more precisely.

Theorem 4.1. *Let $\mathcal{T}$ be a Horn-$\mathcal{ALCFI}$ TBox, $\widehat{\mathcal{T}}$ be obtained by exhaustively applying Rules* **R1**–**R9***, and let A_0 be a concept name. Then A_0 is finitely satisfiable w.r.t. $\mathcal{T}$ iff $A_0 \sqsubseteq \bot \notin \widehat{\mathcal{T}}$.*

Proof of Theorem 4.1. "⇒:" The statement follows from the soundness of the rules **R1**–**R9** w.r.t. finite model entailment. Soundness of each application of rules **R1**–**R8** is straightforward since each rule derives only logical consequences of the axioms in its premises. Further, soundness of **R9** can be shown using Lemma 3.13. Indeed, if the premisses of **R9** are satisfied, i.e., there are conjunctions $K_0, \dots K_{n-1}$ with $K_i \sqsubseteq \exists r_i.K_{i\oplus_n 1}$, $K_{i\oplus_n 1} \sqsubseteq (\leqslant 1\ r_i^-\ A_i)$, and $K_i \sqsubseteq A_i$ are axioms in the partially closed TBox $\mathcal{T}$, then, by definition $K_0, r_0, \dots, r_{n-1}, K_1$ is a finmod-cycle in $\mathcal{T}$, and the axioms added by **R9** are entailed in finite models of $\mathcal{T}$. □

The proof of the "⇐:" direction of Theorem 4.1 requires a more subtle analysis. We will proceed in two steps: Assume that $A_0 \sqsubseteq \bot \notin \widehat{\mathcal{T}}$.

Step 1: We construct a (possibly infinite) model $\widehat{\mathcal{I}}$ of $\widehat{\mathcal{T}}$ with $A_0^{\widehat{\mathcal{I}}} \neq \emptyset$.

Step 2: We show that $\widehat{\mathcal{I}}$ is actually a model of $\mathcal{T}_\mathsf{f}$.

Note that **Step 2** allows us to reach our goal since, by Theorem 3.24, A_0 satisfiable w.r.t. $\mathcal{T}_\mathsf{f}$ implies that A_0 is finitely satisfiable w.r.t. $\mathcal{T}$.

Notation

We next introduce some notation that will be relevant for the rest of this chapter.

- For a TBox $\mathcal{T}$, we write $\widehat{\mathcal{T}}$ to denote the saturation of $\mathcal{T}$ w.r.t. the rules **R1**–**R9**.
- We write $\mathsf{KON}(\widehat{\mathcal{T}})$ to denote the set of all conjunctions K such that K occurs in $\widehat{\mathcal{T}}$ and $K \sqsubseteq \bot \notin \widehat{\mathcal{T}}$.
- For two conjunctions K, K' and a concept name A, we will write $K \vdash_{\widehat{\mathcal{T}}} K'$ as a shortcut for $K \sqsubseteq A \in \widehat{\mathcal{T}}$ for all $A \in K'$.

With this notation at hand, we are ready to proceed with the first step of the proof of the "⇐:" direction of Theorem 4.1.

Proof of Theorem 4.1 "⇐:"

Step 1: construct a model of $\widehat{\mathcal{T}}$. We aim to construct a model $\widehat{\mathcal{I}}$ of $\widehat{\mathcal{T}}$. The domain $\Delta^{\widehat{\mathcal{I}}}$ consists of finite words $d = K_1 K_2 \cdots K_n \in \mathsf{KON}(\widehat{\mathcal{T}})^*$. For a word $d = K_1 K_2 \cdots K_n$ we use $\mathsf{tail}(d)$ to denote K_n. We start with an initial interpretation $\widehat{\mathcal{I}}_0$, such that starting with

$$\begin{aligned}\Delta^{\widehat{\mathcal{I}}_0} &= \mathsf{KON}(\widehat{\mathcal{T}})\\ A^{\widehat{\mathcal{I}}_0} &= \{K \in \mathsf{KON}(\widehat{\mathcal{T}}) \mid K \sqsubseteq A \in \widehat{\mathcal{T}}\}\\ r^{\widehat{\mathcal{I}}_0} &= \emptyset\end{aligned}$$

We assume w.l.o.g. that A_0 actually occurs in $\mathcal{T}$. Further, since $A_0 \sqsubseteq \bot \notin \widehat{\mathcal{T}}$, $\Delta^{\widehat{\mathcal{I}}}$ contains the conjunction $K = A_0$ and thus $A_0^{\widehat{\mathcal{I}}} \neq \emptyset$. The interpretation $\widehat{\mathcal{I}}$ will be the result of exhaustively applying the following rule to $\widehat{\mathcal{I}}_0$:

(†) if there is some $d \in \Delta^{\widehat{\mathcal{I}}}$ with $\mathsf{tail}(d) \sqsubseteq \exists r.K' \in \widehat{\mathcal{T}}$, K' maximal with this property, and $d \notin (\exists r.K')^{\widehat{\mathcal{I}}}$, then add a fresh element $e = dK'$ to $\Delta^{\widehat{\mathcal{I}}}$, add (d, K') to $r^{\widehat{\mathcal{I}}}$, and add dK' to $A^{\widehat{\mathcal{I}}}$ whenever $K' \sqsubseteq A \in \widehat{\mathcal{T}}$.

Now, to conclude **Step 1** it remains to show that $\widehat{\mathcal{I}}$ is a model of $\widehat{\mathcal{T}}$.

Lemma 4.2. *$\widehat{\mathcal{I}} \models \widehat{\mathcal{T}}$.*

Proof. The proof amounts to a case distinction over the forms of CIs that can be present in $\widehat{\mathcal{T}}$, in each case relying on the fact that $\widehat{\mathcal{T}}$ is closed under the rules of the calculus. Let $K \sqsubseteq C \in \widehat{\mathcal{T}}$. We distinguish the following cases.

- **case '$C = \bot$'.**

  If there were some $d \in K^{\widehat{\mathcal{I}}}$, then the construction of $\widehat{\mathcal{I}}$ would ensure that $\mathsf{tail}(d) \vdash_{\widehat{\mathcal{T}}} K$, and from $K \sqsubseteq \bot \in \widehat{\mathcal{T}}$ and **R3** we would get $\mathsf{tail}(d) \sqsubseteq \bot \in \widehat{\mathcal{T}}$, which is impossible, as the following inductive argument shows. If $d = K \in \mathsf{KON}(\widehat{\mathcal{T}})$, the claim follows from the construction in the initial step. If $|d| > 1$, then d was added due to some element d' with $\mathsf{tail}(d') \sqsubseteq \exists r.\mathsf{tail}(d) \in \widehat{\mathcal{T}}$. Now $\mathsf{tail}(d) \sqsubseteq \bot \in \widehat{\mathcal{T}}$ implies $\mathsf{tail}(d') \sqsubseteq \bot \in \widehat{\mathcal{T}}$ due to **R6**, contradicting the inductive hypothesis.

- **case '$C = A$'.**

  Let $d \in K^{\widehat{\mathcal{I}}}$. Then $\mathsf{tail}(d) \vdash_{\widehat{\mathcal{T}}} K$ by construction of $\widehat{\mathcal{I}}$. Together with $K \sqsubseteq A \in \widehat{\mathcal{T}}$, Rule **R3** yields $\mathsf{tail}(d) \sqsubseteq A \in \widehat{\mathcal{T}}$, hence $d \in A^{\widehat{\mathcal{I}}}$ by construction of $\widehat{\mathcal{I}}$.

- **case '$C = \exists r.K'$'.**

  Let $d \in K^{\widehat{\mathcal{I}}}$ and assume $\mathsf{tail}(d) = K''$. By construction of $\widehat{\mathcal{I}}$, we have $K'' \vdash_{\widehat{\mathcal{T}}} K$. Further, from $K \sqsubseteq \exists r.K' \in \widehat{\mathcal{T}}$, by **R3**, $K'' \sqsubseteq \exists r.K' \in \widehat{\mathcal{T}}$. Then the construction ensures that $d' \in (\exists r.K')^{\widehat{\mathcal{I}}}$ as required.

- **case '$C = \forall r.A$'.**

  Let $d \in K^{\widehat{\mathcal{I}}}$ and $(d, d') \in r^{\widehat{\mathcal{I}}}$. Further, let $\mathsf{tail}(d) = K_1$. Since $K \sqsubseteq \forall r.A \in \widehat{\mathcal{T}}$ and $K_1 \vdash_{\widehat{\mathcal{T}}} K$, we get by Rule **R3** that $K_1 \sqsubseteq \forall r.A \in \widehat{\mathcal{T}}$. We distinguish the following cases.

  (i) $d' = dK_2$ i.e., d' was added after d because of some $K_1 \sqsubseteq \exists r.K_2 \in \widehat{\mathcal{T}}$, with K_2 maximal with this property. Since $K_1 \sqsubseteq \forall r.A \in \widehat{\mathcal{T}}$ and $K_1 \sqsubseteq \exists r.K_2 \in \widehat{\mathcal{T}}$, we get by **R5** that $K_1 \sqsubseteq \exists r.(K_2 \sqcap A) \in \widehat{\mathcal{T}}$. Maximality of K_2 implies that $A \in K_2$. By **R1**, we have $K_2 \sqsubseteq A \in \widehat{\mathcal{T}}$ and thus $d' \in A^{\widehat{\mathcal{I}}}$ by construction of $\widehat{\mathcal{I}}$.

  (ii) $d = d'K_1$ i.e., d was added after d' because of some $K_2 \sqsubseteq \exists r^-.K_1 \in \widehat{\mathcal{T}}$ with $K_2 = \mathsf{tail}(d')$. Since $K_1 \sqsubseteq \forall r.A \in \widehat{\mathcal{T}}$ and $K_2 \sqsubseteq \exists r^-.K_1 \in \widehat{\mathcal{T}}$, we get by rule **R4** that $K_2 \sqsubseteq A \in \widehat{\mathcal{T}}$. Thus, $d' \in A^{\widehat{\mathcal{I}}}$ by construction of $\widehat{\mathcal{I}}$.

- **case '$C = (\leqslant 1\ r\ A)$'.**

  Let $d \in K^{\widehat{\mathcal{I}}}$ and let $K'' = \mathsf{tail}(d)$. Assume that there are e_1, e_2 with $(d, e_i) \in r^{\widehat{\mathcal{I}}}$ and $e_i \in A^{\widehat{\mathcal{I}}}$ for $i = 1, 2$. We have $K'' \vdash_{\widehat{\mathcal{T}}} K$ by construction of $\widehat{\mathcal{I}}$ and thus, by Rule **R3**, $K'' \sqsubseteq (\leqslant 1\ r\ A) \in \widehat{\mathcal{T}}$.

  Let $K_i = \mathsf{tail}(e_i)$; hence $K_i \sqsubseteq A \in \widehat{\mathcal{T}}$, $i = 1, 2$. We distinguish two cases according to the construction of $\widehat{\mathcal{I}}$.

  (i) Each e_i was added by $K'' \sqsubseteq \exists r.K_i$ and K_i is maximal with this property. Hence $e_i = dK_i$. Since $K'' \sqsubseteq (\leqslant 1\ r\ A) \in \widehat{\mathcal{T}}$ and $K_i \sqsubseteq A \in \widehat{\mathcal{T}}$, we have by **R7** that $K'' \sqsubseteq \exists r.(K_1 \sqcap K_2) \in \widehat{\mathcal{T}}$. The maximality conditions on both K_i imply $K_1 \subseteq K_2$ and $K_2 \subseteq K_1$. Hence, $e_1 = e_2$, and $d \in (\leqslant 1\ r\ A)^{\widehat{\mathcal{I}}}$ as required.

  (ii) $d = e_1K''$ and $e_2 = dK_2$. Hence d is added after e_1 due to some $K_1 \sqsubseteq \exists r^-.K'' \in \widehat{\mathcal{T}}$ with K'' maximal, and e_2 is added after d due to some $K'' \sqsubseteq \exists r.K_2 \in \widehat{\mathcal{T}}$ with K_2 maximal. Since $K'' \sqsubseteq (\leqslant 1\ r\ A) \in \widehat{\mathcal{T}}$ and $K_i \sqsubseteq A \in \widehat{\mathcal{T}}$, we get by rule **R8** that $K_1 \sqsubseteq A' \in \widehat{\mathcal{T}}$ for every $A' \in K_2$. Then, by construction of $\widehat{\mathcal{I}}$, we have $e_1 \in K_2^{\widehat{\mathcal{I}}}$ and thus e_2 cannot be added as an r-successor of d. Hence, $d \in (\leqslant 1\ r\ A)^{\widehat{\mathcal{I}}}$.

□

We now proceed with the second step of the proof of the "⇐:" direction of Theorem 4.1.

Step 2: $\widehat{\mathcal{T}}$ is satisfiable implies $\mathcal{T}_\mathsf{f}$ is satisfiable. We show that $\widehat{\mathcal{I}}$ is a model of $\mathcal{T}_\mathsf{f}$, which is significantly more difficult to prove than Lemma 4.2 due to the fact that $\mathcal{T}_\mathsf{f}$ is obtained by reversing all cycles in $\mathcal{T}$ whereas the calculus is more careful to reverse only the 'relevant' ones, as explained above. We start with the observation that, when constructing $\mathcal{T}_\mathsf{f}$, it suffices to

reverse only maximal cycles. More precisely, a cycle $K_1, r_1, K_2, \ldots, K_n$ in a TBox $\mathcal{T}$ is *maximal* if every K_{j+1} is (subset) maximal with $\mathcal{T} \models K_j \sqsubseteq \exists r_j.K_{j+1}$, for $1 \leq j < n$.

To illustrate the notion of maximal cycle, consider the TBox $\mathcal{T}_1$ from Example 9:

$$\begin{aligned} A &\sqsubseteq \exists r.(A \sqcap A_1 \sqcap \cdots \sqcap A_n), \\ A &\sqsubseteq (\leqslant 1\ r^-\ A). \end{aligned}$$

Then, for example,

$$\{A\}, r, \{A\} \qquad \text{and} \qquad \{A, A_1, \ldots, A_n\}, r, \{A\}, r, \{A, A_1, \ldots, A_n\}$$

are both cycles in $\mathcal{T}_1$, but not maximal cycles since $\mathcal{T}_1 \models A \sqsubseteq \exists r.(A \sqcap A_1 \sqcap \cdots \sqcap A_n)$ and $\mathcal{T}_1 \models A \sqcap A_1 \sqcap \cdots \sqcap A_n \sqsubseteq \exists r.(A \sqcap A_1 \sqcap \cdots \sqcap A_n)$. In fact, there are (at least) exponentially many cycles in $\mathcal{T}_1$: consider all those of the form K, r, K with $K \subseteq S := \{A, A_1, \ldots, A_n\}$; and the maximal cycle from those is S, r, S.

Let $\mathcal{T}_{\mathsf{f}}^{\max}$ be the variation of $\mathcal{T}_{\mathsf{f}}$ that is obtained by reversing only maximal cycles. We now show the following.

Lemma 4.3. *$\mathcal{T}_{\mathsf{f}}$ is equivalent to $\mathcal{T}_{\mathsf{f}}^{\max}$.*

Proof. It suffices to show that, for every cycle $\mathcal{C}$ in a TBox $\mathcal{S}$, there is a maximal cycle $\widehat{\mathcal{C}}$ in $\mathcal{S}$ whose reversal implies the reversal of $\mathcal{C}$. More precisely, let $\mathcal{C} = K_1, r_1, K_2, \ldots, K_n$ be a cycle in $\mathcal{S}$. We show that there is a maximal cycle $\widehat{\mathcal{C}} = \widehat{K}_1, r_1, \widehat{K}_2, \ldots, \widehat{K}_n$ whose reversal – that is, adding the axioms $\widehat{K}_{j+1} \sqsubseteq \exists r_j^-.\widehat{K}_j$ and $\widehat{K}_j \sqsubseteq (\leqslant 1\ r_j\ \widehat{K}_{j+1})$ to $\mathcal{S}$– will lead to $\mathcal{S}$ implying the reversal of $\mathcal{C}$. We proceed in three steps.

- First, we construct $\widehat{\mathcal{C}} = \widehat{K}_1, r_1, \widehat{K}_2, \ldots, \widehat{K}_n$ iteratively as follows. Initially, set $\widehat{K}_j = K_j$ for every $j = 1, \ldots, n$. Then exhaustively apply the following step.

  While there is some $\widehat{L}_{j+1} \supsetneq \widehat{K}_{j+1}$ maximal with $\mathcal{S} \models \widehat{K}_j \sqsubseteq \exists r_j.\widehat{L}_{j+1}$ for some $j = 1, \ldots, n-1$, set $\widehat{K}_{j+1} = \widehat{L}_{j+1}$.

  The iteration terminates because the supply of conjunctions is bounded and $\mathcal{C}$'s length is fixed.

- Second, we verify that $\widehat{\mathcal{C}}$ is indeed a cycle. It suffices to show that one application of the construction step does not destroy the cycle property, i.e., by replacing $\widehat{K}_{j+1}$ with the larger $\widehat{L}_{j+1}$, the four subsumptions involving $\widehat{K}_{j+1}$ now hold for $\widehat{L}_{j+1}$:
  - $\mathcal{S} \models \widehat{K}_j \sqsubseteq \exists r_j.\widehat{L}_{j+1}$ holds due to the step's precondition.
  - $\mathcal{S} \models \widehat{L}_{j+1} \sqsubseteq \exists r_j.\widehat{K}_{j+2}$ holds because $\mathcal{S} \models \widehat{L}_{j+1} \sqsubseteq \widehat{K}_{j+1} \sqsubseteq \exists r_j.\widehat{K}_{j+2}$.
  - $\mathcal{S} \models \widehat{L}_{j+1} \sqsubseteq (\leqslant 1\ r_j^-\ \widehat{K}_j)$ holds because $\mathcal{S} \models \widehat{L}_{j+1} \sqsubseteq \widehat{K}_{j+1} \sqsubseteq (\leqslant 1\ r_j^-\ \widehat{K}_j)$.
  - $\mathcal{S} \models \widehat{K}_{j+2} \sqsubseteq (\leqslant 1\ r_{j+1}^-\ \widehat{L}_{j+1})$ holds because $\mathcal{S} \models \widehat{K}_{j+2} \sqsubseteq (\leqslant 1\ r_{j+1}^-\ \widehat{K}_{j+1})$ and $\mathcal{S} \models \widehat{L}_{j+1} \sqsubseteq \widehat{K}_{j+1}$.

- Third, we show that the reversal of $\widehat{\mathcal{C}}$ implies the reversal of $\mathcal{C}$. Again, it suffices to show that this is the case when $\widehat{\mathcal{C}}$ is obtained from $\mathcal{C}$ applying one single construction step. Let $\mathcal{S}^+$ be the TBox obtained from $\mathcal{S}$ after reversing $\widehat{\mathcal{C}}$, that is, $\mathcal{S}^+$ equals $\mathcal{S}$ plus the following $2j$ axioms.

$$\begin{array}{llll} \ldots & \widehat{L}_{j+1} \sqsubseteq \exists r_j^-.\widehat{K}_j & \widehat{K}_{j+2} \sqsubseteq \exists r_{j+1}^-.\widehat{L}_{j+1} & (*) \ \ldots \\ \ldots & \widehat{K}_j \sqsubseteq (\leqslant 1\ r_j\ \widehat{L}_{j+1}) & \widehat{L}_{j+1} \sqsubseteq (\leqslant 1\ r_{j+1}\ \widehat{K}_{j+2}) & \ldots \end{array}$$

  To prove that all $2j$ axioms that would be added by reversing $\mathcal{C}$ are implied by $\mathcal{S}^+$, it suffices to show that $\mathcal{S}^+ \models \widehat{K}_{j+1} \sqsubseteq \widehat{L}_{j+1}$ (which implies $\mathcal{S}^+ \models \widehat{K}_{j+1} \equiv \widehat{L}_{j+1}$). Consider an arbitrary model $\widehat{\mathcal{I}} \models \mathcal{S}^+$ and an instance d of $\widehat{K}_{j+1}$ in $\widehat{\mathcal{I}}$. Since $\widehat{\mathcal{C}}$ is a cycle, there is some e with $(d,e) \in r_{j+1}^{\mathcal{I}}$ and $e \in \widehat{K}_{j+2}^{\mathcal{I}}$. Then, due to the above axiom $(*)$ in $\mathcal{S}^+$, there is some d' with $(d',e) \in r_{j+1}^{\mathcal{I}}$ and $d' \in \widehat{L}_{j+1}^{\mathcal{I}}$. Now, since $\mathcal{C}$ is a cycle in $\mathcal{S}$ – i.e., $\widehat{K}_{j+2} \sqsubseteq (\leqslant 1\ r_{j+1}^-\ \widehat{K}_{j+1}) \in \mathcal{S}$ – and because $\widehat{L}_{j+1} \supseteq \widehat{K}_{j+1}$, we obtain that $d' = d$. Hence d is an instance of $\widehat{L}_{j+1}$.

□

Using the previous Lemma, we can prove the desired result: let $\mathcal{T}_{\mathsf{f}}^0, \mathcal{T}_{\mathsf{f}}^1, \ldots$ be the sequence of TBoxes obtained by starting with $\mathcal{T}_{\mathsf{f}}^0 = \mathcal{T}$ and then exhaustively closing maximal cycles, that is, $\mathcal{T}_{\mathsf{f}}^{\max}$ is the limit of this sequence; we prove by induction on i that $\widehat{\mathcal{I}}$ is a model of each $\mathcal{T}_{\mathsf{f}}^i$, thus of $\mathcal{T}_{\mathsf{f}}^{\max}$.

However, we still have to deal with the complication that $\mathcal{T}_{\mathsf{f}}^{\max} \subseteq \widehat{\mathcal{T}}$ needs not hold. As illustrated by Example 9, this is actually a main feature of our calculus because we are avoiding to introduce conjunctions K that are 'irrelevant' for the reasoning task at hand.

We address this issue by showing that the *relevant consequences* of all concept inclusions in $\mathcal{T}_{\mathsf{f}}^{\max} \setminus \widehat{\mathcal{T}}$ are reflected in $\widehat{\mathcal{T}}$, even if the inclusions themselves are missing. To make this more precise, note that $\mathcal{T}_{\mathsf{f}}^{\max} \setminus \widehat{\mathcal{T}}$ only contains CIs of the form

(i) $K \sqsubseteq \exists r.K'$ and

(ii) $K \sqsubseteq (\leqslant 1\ r\ K')$.

Note that, while $\mathcal{T}$ and $\widehat{\mathcal{T}}$ are in the stricter normal form described in (4.1), cycle-reversion may have introduced CIs of the form (ii) with arbitrary conjunctions K'.

For CIs of the form (i), we observe that K, K' may be irrelevant: they may not occur in $\widehat{\mathcal{T}}$. We show that there is some conjunction $\widehat{K}'$ that satisfies $\widehat{K}' \vdash_{\widehat{\mathcal{T}}} K'$ and which intuitively replaces K' such that for all relevant conjunctions $\widehat{K}$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$, the inclusion $\widehat{K} \sqsubseteq \exists r.\widehat{K}'$ is contained in $\widehat{\mathcal{T}}$.

For CIs of the form (ii), we show analogously that there is a replacement A of K' such that for all relevant conjunctions $\widehat{K}$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$, $\widehat{\mathcal{T}}$ contains $\widehat{K} \sqsubseteq (\leqslant 1\ r\ A)$. However, in this case the fact that A is a replacement of K' has to be formalized even a bit more carefully. We cannot

require that $K' \vdash_{\widehat{\mathcal{T}}} A$, again because K' may be irrelevant. Instead, we need that $\widetilde{K}' \vdash_{\widehat{\mathcal{T}}} K'$ implies $\widetilde{K}' \sqsubseteq A \in \widehat{\mathcal{T}}$ for all relevant $\widetilde{K}'$.

What we have just discussed is Lemma 4.5 below. In order to show that the above concept inclusions $\widehat{K} \sqsubseteq \exists r.\widehat{K}'$ and $\widehat{K} \sqsubseteq (\leqslant 1\ r\ A)$ all are in $\widehat{\mathcal{T}}$, we consider the sequence of TBoxes $\mathcal{T}_\mathsf{f}^0, \mathcal{T}_\mathsf{f}^1, \ldots$ that are obtained by repeatedly reversing maximal cycles and whose limit is $\mathcal{T}_\mathsf{f}^{\max}$. Note that $\mathcal{T}_\mathsf{f}^{i+1}$ is produced from $\mathcal{T}_\mathsf{f}^i$ by reversing one cycle, and that cycles are defined in terms of *semantic entailment* of CIs of the form (i) and (ii) by $\mathcal{T}_\mathsf{f}^i$, rather than *syntactic containment*. We first establish an auxiliary lemma that helps in bridging this gap.

Since, by assumption, the original TBox $\mathcal{T}$ is in the stricter normal form introduced at the begin of the chapter (cf. Equation 4.1), all TBoxes $\mathcal{T}_\mathsf{f}^i$ contain $\forall$-restrictions only in the form $\forall r.A$. However, due to cycle reversion, functionality restrictions may occur in the form $(\leqslant 1\ r\ L')$ for arbitrary conjunctions L'. Every $\mathcal{T}_\mathsf{f}^i$ and every conjunction K that is satisfiable w.r.t. $\mathcal{T}_\mathsf{f}^i$ gives rise to a *TBox* $(\mathcal{T}_\mathsf{f}^i)_K$ as follows:

1. for all CIs $L \sqsubseteq C \in \mathcal{T}_\mathsf{f}^i$ with $\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq L$ and C of one of the forms $\exists r.L'$, $\forall r.A$, and $(\leqslant 1\ r\ L')$, include $K \sqsubseteq C$;

2. then exhaustively apply rules **R5** and **R7′**, where **R7′** is obtained from **R7** by replacing *containment* in $\widehat{\mathcal{T}}$ with *entailment* in $\mathcal{T}_\mathsf{f}^i$:

$$\textbf{R7′}\quad \frac{\begin{array}{lll} K \sqsubseteq \exists r.K_1 & K \sqsubseteq \exists r.K_2 & \mathcal{T}_\mathsf{f}^i \models K_1 \sqsubseteq K' \\ K \sqsubseteq (\leqslant 1\ r\ K') & & \mathcal{T}_\mathsf{f}^i \models K_2 \sqsubseteq K' \end{array}}{K \sqsubseteq \exists r.(K_1 \sqcap K_2)}$$

It is easy to see that $\mathcal{T}_\mathsf{f}^i \models (\mathcal{T}_\mathsf{f}^i)_K$. Note that Step 1 above addresses the fact that K need not occur syntactically in $\mathcal{T}_\mathsf{f}^i$.

The proof of the following lemma uses $(\mathcal{T}_\mathsf{f}^i)_K$ to introduce two variants of the canonical model for $\mathcal{T}_\mathsf{f}^i$ and to extract the required witnesses for entailments.

Lemma 4.4. *For every $i \geq 0$, the following hold.*

1. *If $\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq \exists r.K'$ and K is satisfiable w.r.t. $\mathcal{T}_\mathsf{f}^i$ then there is some conjunction L' with*
   (a) *$\mathcal{T}_\mathsf{f}^i \models L' \sqsubseteq K'$ and*
   (b) *$(\mathcal{T}_\mathsf{f}^i)_K \ni K \sqsubseteq \exists r.L'$.*
2. *If $\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq (\leqslant 1\ r\ K')$ and $\mathcal{T}_\mathsf{f}^i \models K' \sqsubseteq \exists r^-.K$ such that K is maximal with this property and K' is satisfiable w.r.t. $\mathcal{T}_\mathsf{f}^i$, then there are L, L' with*
   (a) *$\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq L$, and*
   (b) *$\mathcal{T}_\mathsf{f}^i \models K' \sqsubseteq L'$, and*
   (c) *$\mathcal{T}_\mathsf{f}^i \ni L \sqsubseteq (\leqslant 1\ r\ L')$.*

Proof. We begin by constructing a variant of the canonical model for $\mathcal{T}_\mathsf{f}^i$ that will be used in the proofs of both points of the lemma. Let K be a conjunction satisfiable w.r.t. $\mathcal{T}_\mathsf{f}^i$. The interpretation $\mathcal{I}_K$ is defined as follows. The domain $\Delta^{\mathcal{I}_K}$ consists of words over the alphabet built up of all conjunctions of concept names that occur in $\mathcal{T}_\mathsf{f}^i$ and are satisfiable w.r.t. $\mathcal{T}_\mathsf{f}^i$. Initially, $\Delta^{\mathcal{I}_K}$ is the singleton set $\{d_0\}$ for $d_0 = K$, and the concept and role names are interpreted such that

$$\begin{aligned}\mathsf{tp}_{\mathcal{I}_K}(d_0) &= \{A \mid \mathcal{T}_\mathsf{f}^i \models K \sqsubseteq A\}\\ r^{\mathcal{I}_K} &= \emptyset\end{aligned}$$

Then we add the required successors to the root node d_0. For every $K \sqsubseteq \exists r.L' \in (\mathcal{T}_\mathsf{f}^i)_K$ such that L' is maximal with this property,

- add a fresh element $e = KL'$ to $\Delta^{\mathcal{I}_K}$;
- add the pair (d_0, e) to $r^{\mathcal{I}_K}$;
- interpret concept names such that $\mathsf{tp}_{\mathcal{I}_K}(e) = \{A \mid \mathcal{T}_\mathsf{f}^i \models L' \sqsubseteq A\}$.

Finally, we exhaustively generate required successors of non-root elements. For every $d = wL \in \Delta^{\mathcal{I}_K}$ with $d \neq d_0$, and every inclusion $L \sqsubseteq \exists r.L'$ such that $\mathcal{T}_\mathsf{f}^i \models L \sqsubseteq \exists r.L'$, L' is maximal with this property, and $d \notin (\exists r.L')^{\widehat{\mathcal{I}}}$,

- add a fresh element $e = wLL'$ to $\Delta^{\mathcal{I}_K}$;
- add the pair (d, e) to $r^{\mathcal{I}_K}$;
- interpret concept names such that $\mathsf{tp}_{\mathcal{I}_K}(e) = \{A \mid \mathcal{T}_\mathsf{f}^i \models L' \sqsubseteq A\}$.

Note the difference between the treatment of the root node d_0 and all other nodes: for d_0, we consider inclusions $K \sqsubseteq \exists r.L'$ that are syntactically contained in $(\mathcal{T}_\mathsf{f}^i)_K$; while for all other nodes, we consider inclusions that semantically follow from $\mathcal{T}_\mathsf{f}^i$ (equivalently: from $(\mathcal{T}_\mathsf{f}^i)_K$). We make the following claim to finish the proof, and we provide a proof for it latter.

Claim 2. $\mathcal{I}_K \models \mathcal{T}_\mathsf{f}^i$.

To prove Point (1) in the statement of the Lemma, assume $\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq \exists r.K'$ with K satisfiable w.r.t. $\mathcal{T}_\mathsf{f}^i$. Since $\mathcal{I}_K \models \mathcal{T}_\mathsf{f}^i$ and due to Step 1 of the construction of $\mathcal{I}_K$, there is some L' with $K \sqsubseteq \exists r.L' \in (\mathcal{T}_\mathsf{f}^i)_K$ and $\mathcal{T}_\mathsf{f}^i \models L' \sqsubseteq K'$.

To prove Point (2), assume $\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq (\leqslant 1\ r\ K')$ and $\mathcal{T}_\mathsf{f}^i \models K' \sqsubseteq \exists r^-.K$ with K maximal and K' – and thus K – satisfiable w.r.t. $\mathcal{T}_\mathsf{f}^i$. We construct an interpretation $\mathcal{J}$ from the models $\mathcal{I}_K$ and $\mathcal{I}_{K'}$ as follows. Start with two copies of $\mathcal{I}_{K'}$ and one of $\mathcal{I}_K$, pairwise disjoint. Since $\mathcal{T}_\mathsf{f}^i \models K' \sqsubseteq \exists r^-.K$ with K maximal, the root d_i of each of the two copies of $\mathcal{I}_{K'}$ has an r^--successor e_i of type K. Delete the subtrees starting at e_i and replace the r-edges (e_i, d_i) with (d_0, d_i), where d_0 is the root of the copy of $\mathcal{I}_K$.

Now the proof of the previous claim that $\mathcal{I}_K$ and $\mathcal{I}_{K'}$ are models of $\mathcal{T}_\mathsf{f}^i$ can be easily refined to yield that

- all axioms $L \sqsubseteq C \in \mathcal{T}_\mathsf{f}^i$ where C is of the form $A, \bot, \exists s.L', \forall s.A$ are satisfied by $\mathcal{J}$;
- if an axiom $L \sqsubseteq (\leqslant 1\ s\ L') \in \mathcal{T}_\mathsf{f}^i$ is violated by $\mathcal{J}$, then it is violated by the root of the copy of $\mathcal{I}_K$, i.e., $s = r$ and $d_0 \in L^\mathcal{J}$ but $d_0 \notin (\leqslant 1\ r\ L')^\mathcal{J}$.

Since $\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq (\leqslant 1\ r\ K')$ but obviously $\mathcal{J} \not\models K \sqsubseteq (\leqslant 1\ r\ K')$, we have that $\mathcal{J} \not\models \mathcal{T}_\mathsf{f}^i$. Consequently, there is an axiom $L \sqsubseteq (\leqslant 1\ r\ L') \in \mathcal{T}_\mathsf{f}^i$ which is violated by d_0; that is, $d_0 \in L^\mathcal{J} \setminus (\leqslant 1\ r\ L')^\mathcal{J}$. This establishes (c) directly and implies (a) and (b): first, by construction of the $\mathcal{I}_{K'}$, we get $\mathcal{T}_\mathsf{f}^i \models K' \sqsubseteq L'$, which is (b). Second, with $L \sqsubseteq (\leqslant 1\ r\ L') \in \mathcal{T}_\mathsf{f}^i$ and (b), we obtain $\mathcal{T}_\mathsf{f}^i \models L \sqsubseteq (\leqslant 1\ r\ K')$. Since K is maximal with this property, we have $L \subseteq K$, which implies (a). □

It remains to proove the auxiliary Claim 2 used in the previous proof, which amounts to a case distinction on the form of the axioms in $\mathcal{T}_\mathsf{f}^i$.

Proof of Claim 2. Let $L \sqsubseteq C \in \mathcal{T}_\mathsf{f}^i$. We distinguish the following cases.

- $C = \bot$. Assume that L has an instance $d = w\widehat{L}$ in $\mathcal{I}_K$. Then $\mathcal{T}_\mathsf{f}^i \models \widehat{L} \sqsubseteq L$ due to the construction of $\mathcal{I}_K$; hence $\mathcal{T}_\mathsf{f}^i \models \widehat{L} \sqsubseteq \bot$, which is impossible, as the following inductive argument shows. If $d = K$, the claim follows from the assumption that K is satisfiable in $\mathcal{T}_\mathsf{f}^i$. If $|d| > 1$ then d was added due to some element $d' = wL$ with $\mathcal{T}_\mathsf{f}^i \models L \sqsubseteq \exists r.\widehat{L}$. Then $\mathcal{T}_\mathsf{f}^i \models \widehat{L} \sqsubseteq \bot$ implies $\mathcal{T}_\mathsf{f}^i \models L \sqsubseteq \bot \in \widehat{\mathcal{T}}$, contradicting the inductive hypothesis.

- $C = A$. Let $d \in L^{\mathcal{I}_K}$ with $d = w\widehat{L}$. Then $\mathcal{T}_\mathsf{f}^i \models \widehat{L} \sqsubseteq L$ by construction of $\mathcal{I}_K$. Since $L \sqsubseteq A \in \mathcal{T}_\mathsf{f}^i$, we obtain $\mathcal{T}_\mathsf{f}^i \models \widehat{L} \sqsubseteq A$; hence $d \in A^{\mathcal{I}_K}$.

- $C = \exists r.L'$. Let $d \in L^{\mathcal{I}_K}$.

  In case $d = d_0 = K$, we have that $\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq L$. Together with $L \sqsubseteq \exists r.L' \in \mathcal{T}_\mathsf{f}^i$, this implies that $K \sqsubseteq \exists r.L' \in (\mathcal{T}_\mathsf{f}^i)_K$ by Step 1 of the construction of $(\mathcal{T}_\mathsf{f}^i)_K$. Let K' be maximal with $K \sqsubseteq \exists r.K' \in (\mathcal{T}_\mathsf{f}^i)_K$ and $L' \subseteq K'$. In the construction of $\mathcal{I}_K$, we thus create an r-successor e of d with $\mathsf{tp}_{\mathcal{I}_K}(e) \supseteq L'$. Hence $d \in (\exists r.L')^{\mathcal{I}_K}$.

  In case $d \neq d_0$, let $d = wK'$. Then $\mathcal{T}_\mathsf{f}^i \models K' \sqsubseteq L$. Together with $L \sqsubseteq \exists r.L' \in \mathcal{T}_\mathsf{f}^i$, this implies that $\mathcal{T}_\mathsf{f}^i \models K' \sqsubseteq \exists r.L'$. Then the construction of $\mathcal{I}_K$ ensures that there is an r-successor e of d with $\mathsf{tp}_{\mathcal{I}_K}(e) \supseteq L'$. Hence $d \in (\exists r.L')^{\mathcal{I}_K}$.

- $C = \forall r.A$. Let $d \in L^{\mathcal{I}_K}$ and $(d, e) \in r^{\mathcal{I}_K}$.

  In case $d = d_0 = K$ and $e = KK'$, we have that $\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq L$; hence $K \sqsubseteq \forall r.A \in (\mathcal{T}_\mathsf{f}^i)_K$ as above. Now e was added for some $K \sqsubseteq \exists r.K' \in (\mathcal{T}_\mathsf{f}^i)_K$ with K' maximal. Since $(\mathcal{T}_\mathsf{f}^i)_K$ is closed under application of **R5**, we have that $K \sqsubseteq \exists r.(K' \sqcap A) \in (\mathcal{T}_\mathsf{f}^i)_K$. Maximality of K' implies that $A \in K'$. The construction of $\mathcal{I}_K$ then implies that $A \in \mathsf{tp}_{\mathcal{I}_K}(e)$; i.e., $e \in A^{\mathcal{I}_K}$.

In case $e = d_0 = K$ and $d = KK'$, we have that $\mathcal{T}_\mathsf{f}^i \models K' \sqsubseteq L$; hence $\mathcal{T}_\mathsf{f}^i \models K' \sqsubseteq \forall r.A$ (x). Now d was added for some $K \sqsubseteq \exists r^-.K' \in (\mathcal{T}_\mathsf{f}^i)_K$ with K'. Since $\mathcal{T}_\mathsf{f}^i \models (\mathcal{T}_\mathsf{f}^i)_K$, we obtain $\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq \exists r^-.K'$. Together with (x), a simple semantic argument implies $\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq A$; hence, $e \in A^{\mathcal{I}_K}$.

In case $d = wK_1 \neq d_0$ and $e = wK_1K_2$, we have that $\mathcal{T}_\mathsf{f}^i \models K_1 \sqsubseteq L$; hence $\mathcal{T}_\mathsf{f}^i \models K_1 \sqsubseteq \forall r.A$ as above. Now e was added because $\mathcal{T}_\mathsf{f}^i \models K_1 \sqsubseteq \exists r.K_2$ with K_2 maximal. A simple semantic argument implies $\mathcal{T}_\mathsf{f}^i \models K_1 \sqsubseteq \exists r.(K_2 \sqcap A)$, and maximality of K_2 again yields $A \in K_2$; i.e., $e \in A^{\mathcal{I}_K}$.

In case $e = wK_1 \neq d_0$ and $d = wK_1K_2$, we have that $\mathcal{T}_\mathsf{f}^i \models K_2 \sqsubseteq L$; hence $\mathcal{T}_\mathsf{f}^i \models K_2 \sqsubseteq \forall r.A$. Now d was added because $\mathcal{T}_\mathsf{f}^i \models K_1 \sqsubseteq \exists r^-.K_2$. A simple semantic argument implies $\mathcal{T}_\mathsf{f}^i \models K_1 \sqsubseteq A$, i.e., $e \in A^{\mathcal{I}_K}$.

- $C = (\leqslant 1\ r\ L')$. Let $d \in L^{\mathcal{I}_K}$, and let $(d, e_i) \in r^{\mathcal{I}_K}$ and $e_i \in (L')^{\mathcal{I}_K}$ for $i = 1, 2$.

  In case $d = d_0 = K$ and $e_i = KK_i$, we have that (i) $\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq L$ and (ii) $\mathcal{T}_\mathsf{f}^i \models K_i \sqsubseteq L'$ for $i = 1, 2$. By construction of $(\mathcal{T}_\mathsf{f}^i)_K$, (i) and the assumption imply (iii) $K \sqsubseteq (\leqslant 1\ r\ L') \in (\mathcal{T}_\mathsf{f}^i)_K$. Now each e_i was added for some (iv) $K \sqsubseteq \exists r.K_i \in (\mathcal{T}_\mathsf{f}^i)_K$ with K_i maximal. Applying **R7′** to (iv), (iii), (ii) yields $K \sqsubseteq \exists r.(K_1 \sqcap K_2) \in (\mathcal{T}_\mathsf{f}^i)_K$. Maximality of the K_i implies that $K_1 = K_2$; hence $e_1 = e_2$.

  In case $e_1 = d_0 = K$, $d = KK_1$, and $e_2 = KK_1K_2$, we have that (i) $\mathcal{T}_\mathsf{f}^i \models K_1 \sqsubseteq L$ plus (ii) $\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq L'$ and (iii) $\mathcal{T}_\mathsf{f}^i \models K_2 \sqsubseteq L'$. By construction of $(\mathcal{T}_\mathsf{f}^i)_K$, (i) and the assumption imply (iv) $\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq (\leqslant 1\ r\ L')$. Now d was added for some (v) $K \sqsubseteq \exists r^-.K_1 \in (\mathcal{T}_\mathsf{f}^i)_K$, and e_2 was added for some (vi) $K_1 \sqsubseteq \exists r.K_2 \in (\mathcal{T}_\mathsf{f}^i)_K$. A simple semantic argument applied to (v), (vi), (iv), (ii) and (iii) yields $\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq K_2$. This contradicts the assumption that e_2 was added for (vi).

  In case $d = wK' \neq d_0$ and $e_i = wK'K_i$, we argue as in the first case, but purely on a semantic basis, i.e., referring to entailment by $\mathcal{T}_\mathsf{f}^i$ instead of containment in $(\mathcal{T}_\mathsf{f}^i)_K$.

  In case $e_1wK' \neq d_0$, $d = wK'K_1$, and $e_2 = wK'K_1K_2$, we argue "semantically" as in the second case.

□

We say that a conjunction K of concept names is *active in* $\widehat{\mathcal{T}}$ if there is a $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$. To achieve our goal in this step, it just remains to prove the following.

Lemma 4.5. *For every $i \geq 0$, the following hold.*

1. *If $\mathcal{T}_\mathsf{f}^i \ni K \sqsubseteq \exists r.K'$ and K is active in $\widehat{\mathcal{T}}$, then there is a $\widehat{K}' \in \mathsf{KON}(\widehat{\mathcal{T}})$ such that*
   a) *$\widehat{K}' \vdash_{\widehat{\mathcal{T}}} K'$;*
   b) *for all $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$, we have $\widehat{\mathcal{T}} \ni \widehat{K} \sqsubseteq \exists r.\widehat{K}'$.*

2. *If* $\mathcal{T}_\mathsf{f}^i \ni K \sqsubseteq (\leqslant 1\ r\ K')$ *and* K *is active in* $\widehat{\mathcal{T}}$*, then there is a concept name* A *such that*
   a) *for all* $\widetilde{K}' \in \mathsf{KON}(\widehat{\mathcal{T}})$*: if* $\widetilde{K}' \vdash_{\widehat{\mathcal{T}}} K'$ *then* $\widetilde{K}' \sqsubseteq A \in \widehat{\mathcal{T}}$*.*
   b) *for all* $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ *with* $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$*, we have* $\widehat{\mathcal{T}} \ni \widehat{K} \sqsubseteq (\leqslant 1\ r\ A)$*.*
3. $\widehat{\mathcal{I}} \models \mathcal{T}_\mathsf{f}^i$
4. *Point 1 holds when "*$\mathcal{T}_\mathsf{f}^i \ni K \sqsubseteq \exists r.K'$*" is replaced with "*$\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq \exists r.K'$*".*
5. *Point 2 holds when "*$\mathcal{T}_\mathsf{f}^i \ni K \sqsubseteq (\leqslant 1\ r\ K')$*" is replaced with "*$\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq (\leqslant 1\ r\ K')$*,* $\mathcal{T}_\mathsf{f}^i \models K' \sqsubseteq \exists r^-.K$*, K is maximal with this property, and K' is active in* $\widehat{\mathcal{T}}$*".*

Proof of Lemma 4.5. We simultaneously prove Points 1–5 by induction on i, showing

- the straightforward base case for Points 1–3;
- that Points 1–3 imply Points 4 and 5 for *every* $i \geq 0$;
- the induction step for Points 1–2 simultaneously, and for Point 3.

For Point 1 of the base case, assume that $\mathcal{T}_\mathsf{f}^0 = \mathcal{T} \ni K \sqsubseteq \exists r.K'$ with K active in $\widehat{\mathcal{T}}$. Then $K \in \mathsf{KON}(\widehat{\mathcal{T}})$ because $\mathcal{T} \subseteq \widehat{\mathcal{T}}$, and K' is the required $\widehat{K}'$: (a) and (b) are due to Rules **R1** and **R3**, respectively.

For Point 2 of the base case, assume that $\mathcal{T}_\mathsf{f}^0 = \mathcal{T} \ni K \sqsubseteq (\leqslant 1\ r\ K')$. Then K' is in fact a concept name A because we are assuming $\mathcal{T}$ to be in the stricter normal form. This A is the required concept name: (a) holds trivially, and (b) is due to Rule **R3**.

For Point 3 of the base case follows from $\widehat{\mathcal{I}} \models \widehat{\mathcal{T}}$ (Lemma 4.2) and $\widehat{\mathcal{T}} \supseteq \mathcal{T} = \mathcal{T}_\mathsf{f}^0$.

For Point 4, we will show that, for *every* $i \geq 0$, Point 4 follows from Points 1–3. The following argument thus combines base case and induction step for Point 4.

Assume that $\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq \exists r.K'$ with K active in $\widehat{\mathcal{T}}$. We also have that K is satisfiable w.r.t. $\mathcal{T}_\mathsf{f}^i$ since $\widehat{\mathcal{I}} \models \mathcal{T}_\mathsf{f}^i$ by Point 3 and, by construction, $\widehat{\mathcal{I}}$ has an instance of K. Consider the TBox $(\mathcal{T}_\mathsf{f}^i)_K$. By Lemma 4.4 (1), there is some L' such that

(a′) $\mathcal{T}_\mathsf{f}^i \models L' \sqsubseteq K'$ and

(b′) $(\mathcal{T}_\mathsf{f}^i)_K \ni K \sqsubseteq \exists r.L'$.

We will show below that, for every $K \sqsubseteq \exists r.L'$ in $(\mathcal{T}_\mathsf{f}^i)_K$, there is some $\widehat{K}' \in \mathsf{KON}(\widehat{\mathcal{T}})$ with

(a″) $\widehat{K}' \vdash_{\widehat{\mathcal{T}}} L'$

(b″) for all $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$, we have $\widehat{\mathcal{T}} \ni \widehat{K} \sqsubseteq \exists r.\widehat{K}'$.

This implies (a) and (b). (b) is exactly (b‴), and (a) follows from (a′) and (a″) by inspecting the root element $\widehat{K}'$ of $\Delta^{\widehat{\mathcal{I}}}$: by construction of $\widehat{\mathcal{I}}$ and (a″), this element is an instance of L'; since $\widehat{\mathcal{I}} \models \widehat{\mathcal{T}}_{\mathsf{f}}^i$ (Point 3), it is an instance of K' too; by construction of $\widehat{\mathcal{I}}$, we get $\widehat{K}' \vdash_{\widehat{\mathcal{T}}} K'$.

To prove the above, we use induction on the number of rule applications used to construct $(\mathcal{T}_{\mathsf{f}}^i)_K$. The base case is that $K \sqsubseteq \exists r.L' \in (\mathcal{T}_{\mathsf{f}}^i)_K$ enters $(\mathcal{T}_{\mathsf{f}}^i)_K$ in Step 1 of the construction. Then there is some $L \sqsubseteq \exists r.L' \in \mathcal{T}_{\mathsf{f}}^i$ with $\mathcal{T}_{\mathsf{f}}^i \models K \sqsubseteq L$. Since K is active in $\widehat{\mathcal{T}}$, so is L: for some $\widetilde{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widetilde{K} \vdash_{\widehat{\mathcal{T}}} K$, the root element $\widetilde{K}$ of $\Delta^{\widehat{\mathcal{I}}}$ must make L true. By Point 1, there is a $\widehat{K}' \in \mathsf{KON}(\widehat{\mathcal{T}})$ with (a″) and (b″) as required.

In the induction step, $K \sqsubseteq \exists r.L'$ enters $(\mathcal{T}_{\mathsf{f}}^i)_K$ in Step 2 of the construction. In case this happens via an application of **R5**, we have that $L' = L'_1 \sqcap A$ and

(i) $K \sqsubseteq \exists r.L'_1 \in (\mathcal{T}_{\mathsf{f}}^i)_K$

(ii) $K \sqsubseteq \forall r.A \in (\mathcal{T}_{\mathsf{f}}^i)_K$

Applying the induction hypothesis to (i), we obtain

(i′) there is some $\widehat{K}'_1 \in \mathsf{KON}(\widehat{\mathcal{T}})$ with

(a‴) $\widehat{K}'_1 \vdash_{\widehat{\mathcal{T}}} L'_1$

(b‴) for all $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$, we have $\widehat{\mathcal{T}} \ni \widehat{K} \sqsubseteq \exists r.\widehat{K}'_1$.

From (ii), we obtain $L \sqsubseteq \forall r.A \in \mathcal{T}_{\mathsf{f}}^i$ for some L with $\mathcal{T}_{\mathsf{f}}^i \models K \sqsubseteq L$ because axioms with $\forall$-restrictions never enter $(\mathcal{T}_{\mathsf{f}}^i)_K$ in Step 2 of the construction. Since such axioms are not generated by closing cycles either, we even have $L \sqsubseteq \forall r.A \in \mathcal{T}$; hence

(ii′) $L \sqsubseteq \forall r.A \in \widehat{\mathcal{T}}$ with $\mathcal{T}_{\mathsf{f}}^i \models K \sqsubseteq L$.

We now observe that $\mathcal{T}_{\mathsf{f}}^i \models K \sqsubseteq L$ and $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$ imply $\widehat{K} \vdash_{\widehat{\mathcal{T}}} L$ (again by consulting the domain element of $\widehat{\mathcal{I}}$ created for $\widehat{K}$). Hence, application of **R3** to (ii′) yields

(ii″) $\widehat{K} \sqsubseteq \forall r.A \in \widehat{\mathcal{T}}$ for all $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$.

Now $\widehat{K}' := \widehat{K}'_1 \sqcap A$ is as required:

- $\widehat{K}' \in \mathsf{KON}(\widehat{\mathcal{T}})$.

  Since K is active in $\widehat{\mathcal{T}}$, there is some $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$. By (b‴), we thus have $\widehat{K} \sqsubseteq \exists r.\widehat{K}'_1 \in \widehat{\mathcal{T}}$. Applying **R5** to this and (ii′) yields $\widehat{K} \sqsubseteq \exists r.(\widehat{K}'_1 \sqcap A) \in \widehat{\mathcal{T}}$. Hence $\widehat{K}'_1 \sqcap A \in \mathsf{KON}(\widehat{\mathcal{T}})$.

- (a″) is satisfied, that is, $\widehat{K}'_1 \sqcap A \vdash_{\widehat{\mathcal{T}}} L'_1 \sqcap A$.

  For every $A' \in L'_1$, we have $\widehat{K}'_1 \sqcap A \sqsubseteq A' \in \widehat{\mathcal{T}}$ because of (a‴), **R1**, **R3**. Furthermore, $\widehat{K}'_1 \sqcap A \sqsubseteq A \in \widehat{\mathcal{T}}$ due to **R1**.

- (b'') is satisfied.

  Let $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ such that $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$. Then **R5** applied to (b''') and (ii'') implies $\widehat{K} \sqsubseteq \exists r.(\widehat{K}_1' \sqcap A) \in \widehat{\mathcal{T}}$.

In case $K \sqsubseteq \exists r.L'$ enters $(\mathcal{T}_\mathsf{f}^i)_K$ via an application of **R7′**, we have that $L' = L_1' \sqcap L_2'$ and

(i) $K \sqsubseteq \exists r.L_j' \in (\mathcal{T}_\mathsf{f}^i)_K$, $j = 1, 2$

(ii) $K \sqsubseteq (\leqslant 1\ r\ L_3') \in (\mathcal{T}_\mathsf{f}^i)_K$ for some L_3' with

(iii) $\mathcal{T}_\mathsf{f}^i \models L_j' \sqsubseteq L_3'$, $j = 1, 2$.

Applying the induction hypothesis to (i), we obtain

(i′) there is some $\widehat{K}_j' \in \mathsf{KON}(\widehat{\mathcal{T}})$, $j = 1, 2$, with

(a''') $\widehat{K}_j' \vdash_{\widehat{\mathcal{T}}} L_j'$

(b''') for all $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$, we have $\widehat{\mathcal{T}} \ni \widehat{K} \sqsubseteq \exists r.\widehat{K}_j'$.

Regarding (ii), we observe that axioms with functionality restrictions never enter $(\mathcal{T}_\mathsf{f}^i)_K$ in Step 2. Hence, there is some $L \sqsubseteq (\leqslant 1\ r\ L_3') \in \mathcal{T}_\mathsf{f}^i$ with $\mathcal{T}_\mathsf{f}^i \models K \sqsubseteq L$, and the same observation as in the previous case yields $\widehat{K} \vdash_{\widehat{\mathcal{T}}} L$ whenever $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$. We thus obtain from Point 2 that

(ii′) there is some A with

(a^4) for all $\widetilde{K}' \in \mathsf{KON}(\widehat{\mathcal{T}})$: if $\widetilde{K}' \vdash_{\widehat{\mathcal{T}}} L_3'$, then $\widetilde{K}' \sqsubseteq A \in \widehat{\mathcal{T}}$.

(b^4) for all $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$, we have $\widehat{\mathcal{T}} \ni \widehat{K} \sqsubseteq (\leqslant 1\ r\ A)$.

Furthermore (iii) and (a''') yield

(iii′) $\widehat{K}_j' \vdash_{\widehat{\mathcal{T}}} L_3'$ for $i = 1, 2$,

and with (a^4) we get

(iii″) $\widehat{K}_j' \sqsubseteq A \in \widehat{\mathcal{T}}$ for $i = 1, 2$.

Now $\widehat{K}' = \widehat{K}_1' \sqcap \widehat{K}_2'$ is as required:

- $\widehat{K}' \in \mathsf{KON}(\widehat{\mathcal{T}})$.

  Since K is active in $\widehat{\mathcal{T}}$, there is some $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$. By ($b'''$) and ($b^4$), we thus have $\widehat{K} \sqsubseteq \exists r.\widehat{K}_j' \in \widehat{\mathcal{T}}$ and $\widehat{K} \sqsubseteq (\leqslant 1\ r\ A) \in \widehat{\mathcal{T}}$. Applying **R7** to these and (iii″) yields $\widehat{K} \sqsubseteq \exists r.(\widehat{K}_1' \sqcap \widehat{K}_2') \in \widehat{\mathcal{T}}$. Hence $\widehat{K}_1' \sqcap \widehat{K}_2' \in \mathsf{KON}(\widehat{\mathcal{T}})$.

- (a'') is satisfied, that is, $\widehat{K}_1' \sqcap \widehat{K}_2' \vdash_{\widehat{\mathcal{T}}} L_1' \sqcap L_2'$.

  For every $i = 1, 2$ and $A' \in L_i'$, we have $\widehat{K}_1' \sqcap \widehat{K}_2' \sqsubseteq A \in \widehat{\mathcal{T}}$ because of (a'''), **R1**, and **R3**.

- (b″) is satisfied.

  Let $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ such that $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$. Then **R7** applied to (b‴), (b[4]), (iii″) implies $\widehat{K} \sqsubseteq \exists r.(\widehat{K}'_1 \sqcap \widehat{K}'_2) \in \widehat{\mathcal{T}}$.

For Point 5, we will show that, for *every* $i \geq 0$, Point 5 follows from Points 1–3.

Assume that $\mathcal{T}_{\mathsf{f}}^i \models K \sqsubseteq (\leqslant 1\ r\ K')$, $\mathcal{T}_{\mathsf{f}}^i \models K' \sqsubseteq \exists r^-.K$, K is maximal with this property, and K' is active in $\widehat{\mathcal{T}}$. We have to show that there is an A such that

(a) for all $\widetilde{K}' \in \mathsf{KON}(\widehat{\mathcal{T}})$: if $\widetilde{K}' \vdash_{\widehat{\mathcal{T}}} K'$, then $\widetilde{K}' \sqsubseteq A \in \widehat{\mathcal{T}}$.

(b) for all $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$, we have $\widehat{\mathcal{T}} \ni \widehat{K} \sqsubseteq (\leqslant 1\ r\ A)$.

Since K' is active in $\widehat{\mathcal{T}}$ and $\widehat{\mathcal{I}} \models \mathcal{T}_{\mathsf{f}}^i$ by Point 3, we also have that K' is satisfiable w.r.t. $\mathcal{T}_{\mathsf{f}}^i$. By Lemma 4.4 (2), there is some $L \sqsubseteq (\leqslant 1\ r\ L') \in \mathcal{T}_{\mathsf{f}}^i$ with

(i) $\mathcal{T}_{\mathsf{f}}^i \models K \sqsubseteq L$ and

(ii) $\mathcal{T}_{\mathsf{f}}^i \models K' \sqsubseteq L'$.

We apply Point 2 and conclude that there is some A with

(a′) for all $\widetilde{K}' \in \mathsf{KON}(\widehat{\mathcal{T}})$: if $\widetilde{K}' \vdash_{\widehat{\mathcal{T}}} L'$, then $\widetilde{K}' \sqsubseteq A \in \widehat{\mathcal{T}}$;

(b′) for all $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} L$, we have $\widehat{\mathcal{T}} \ni \widehat{K} \sqsubseteq (\leqslant 1\ r\ A)$.

It remains to show that A is the required conjunction. For (a), take some $\widetilde{K}' \in \mathsf{KON}(\widehat{\mathcal{T}})$ and assume that $\widetilde{K}' \vdash_{\widehat{\mathcal{T}}} K'$. Then (ii) and Point 3 ($\widehat{\mathcal{I}} \models \mathcal{T}_{\mathsf{f}}^i$) imply that $\widetilde{K}' \vdash_{\widehat{\mathcal{T}}} L'$. Now (a′) implies that $\widetilde{K}' \sqsubseteq A \in \widehat{\mathcal{T}}$.

For (b), take some $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K$. Then (i) and $\widehat{\mathcal{I}} \models \mathcal{T}_{\mathsf{f}}^i$ imply that $\widehat{K} \vdash_{\widehat{\mathcal{T}}} L$. Now (b′) implies that $\widehat{\mathcal{T}} \ni \widehat{K} \sqsubseteq (\leqslant 1\ r\ A)$.

For Points 1–2 of the induction step, we prove both points simultaneously. If any of the CIs $K \sqsubseteq \exists r.K'$ and $K \sqsubseteq (\leqslant 1\ r\ K')$ is in $\mathcal{T}_{\mathsf{f}}^{i-1}$, then we can use the induction hypothesis for it. Otherwise, the respective CI has been introduced by closing a cycle $K_1, r_1, K_2, \ldots, r_{n-1}, K_n$ in $\mathcal{T}_{\mathsf{f}}^{i-1}$ with $K = K_j$ for some $j \in \{1, \ldots, n-1\}$, and thus K_j is active in $\widehat{\mathcal{T}}$. Take some $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K_j$. Applying Point 4 of the induction hypothesis to $\mathcal{T}_{\mathsf{f}}^{i-1} \models K_j \sqsubseteq \exists r_j.K_{j+1}$, we find a $\widehat{K}_{j+1} \in \mathsf{KON}(\widehat{\mathcal{T}})$ such that

(i) $\widehat{K}_{j+1} \vdash_{\widehat{\mathcal{T}}} K_{j+1}$, and

(ii) for all $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K_j$, we have $\widehat{\mathcal{T}} \ni \widehat{K} \sqsubseteq \exists r_j.\widehat{K}_{j+1}$.

By Point (i), K_{j+1} is active in $\widehat{\mathcal{T}}$ and thus we can iterate the argument to find

$$\widehat{K}_{j+2}, \ldots, \widehat{K}_n = \widehat{K}_1, \widehat{K}_2, \ldots, \widehat{K}_j$$

with the following properties, for $1 \leq j < n$.

(iii) $\widehat{K}_j \vdash_{\widehat{\mathcal{T}}} K_j$ and $\widehat{K}_j \in \mathsf{KON}(\widehat{\mathcal{T}})$;

(iv) $\widehat{\mathcal{T}} \ni \widehat{K}_j \sqsubseteq \exists r_j.\widehat{K}_{j+1}$.

Let $1 < j \leq n$. Applying Point 5 of the induction hypothesis to $\mathcal{T}_{\mathsf{f}}^{i-1} \models K_{j+1} \sqsubseteq (\leqslant 1\ r_j^-\ K_j)$, we find an A_j such that

(v) for all $\widetilde{K}_j \in \mathsf{KON}(\widehat{\mathcal{T}})$: if $\widetilde{K}_j \vdash_{\widehat{\mathcal{T}}} K_j$, then $\widetilde{K}_j \sqsubseteq A_j \in \widehat{\mathcal{T}}$;

(vi) for all $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K_{j+1}$, we have $\widehat{\mathcal{T}} \ni \widehat{K} \sqsubseteq (\leqslant 1\ r_j^-\ A_j)$.

From (vi) and (i), in particular we obtain:

(vii) $\widehat{\mathcal{T}} \ni \widehat{K}_{j+1} \sqsubseteq (\leqslant 1\ r_j^-\ A_j)$.

From (iii) and (v), we obtain:

(viii) $\widehat{K}_j \sqsubseteq A_j \in \widehat{\mathcal{T}}$.

We can now apply the cycle rule **R9** to the CIs in (iv), (vii) and (viii), obtaining

(ix) $\widehat{\mathcal{T}} \ni \widehat{K}_{j+1} \sqsubseteq \exists r_j^-.\widehat{K}_j$

(x) $\widehat{\mathcal{T}} \ni \widehat{K}_j \sqsubseteq (\leqslant 1\ r_j\ A_{j+1})$

To establish both Points 1 and 2, we set $\widehat{K}' = \widehat{K}_j$ and $A = A_{j+1}$, and we have to show

(1a) $\widehat{K}_j \vdash_{\widehat{\mathcal{T}}} K'$;

(1b) for all $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K_{j+1}$, we have $\widehat{\mathcal{T}} \ni \widehat{K} \sqsubseteq \exists r_j^-.\widehat{K}_j$;

(2a) for all $\widetilde{K}' \in \mathsf{KON}(\widehat{\mathcal{T}})$: if $\widetilde{K}' \vdash_{\widehat{\mathcal{T}}} K_{j+1}$ then $\widetilde{K}' \sqsubseteq A_{j+1} \in \widehat{\mathcal{T}}$;

(2b) for all $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K_j$, we have $\widehat{\mathcal{T}} \ni \widehat{K} \sqsubseteq (\leqslant 1\ r_j\ A_{j+1})$.

Now (1a) and (2a) are just (iii) and (v); hence it remains to show (1b) and (2b). We first claim that

(xi) For every $j \geq 1$, we have $\widehat{K} \sqsubseteq A \in \widehat{\mathcal{T}}$ for all $A \in \widehat{K}_j$.

To show (xi), let $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K_j$. From (ii) and (v), we get that

(xii) $\widehat{K} \sqsubseteq \exists r_j.\widehat{K}_{j+1} \in \widehat{\mathcal{T}}$.

From (viii), and from (v) with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K_j$, we obtain:

(xiii) $\widehat{K}_j \sqsubseteq A,\ \widehat{K} \sqsubseteq A_j\ \in \widehat{\mathcal{T}}$.

Applying **R8** to (xii), (ix), (xiii), (vi) yields (xi).

For showing (1b), take a $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K_{j+1}$. With (xi), we get $\widehat{K} \sqsubseteq A \in \widehat{\mathcal{T}}$ for every $A \in \widehat{K}_{j+1}$. Hence, **R3** and (ix) give us that $\widehat{K} \sqsubseteq \exists r_j^-.\widehat{K}_j \in \widehat{\mathcal{T}}$. For showing (2b), take $\widehat{K} \in \mathsf{KON}(\widehat{\mathcal{T}})$ with $\widehat{K} \vdash_{\widehat{\mathcal{T}}} K_j$. Again with (xi), we get that $\widehat{K} \sqsubseteq A \in \widehat{\mathcal{T}}$ for every $A \in \widehat{K}_j$ which, by **R3** and (x), yields $\widehat{K} \sqsubseteq (\leqslant 1\ r_j\ A_{j+1}) \in \widehat{\mathcal{T}}$ as required.

For Point 3 of the induction step. For every $K \sqsubseteq C \in \mathcal{T}_{\mathsf{f}}^i$, if K is realized in $\widehat{\mathcal{I}}$, then in the form of a supertype from $\mathsf{KON}(\widehat{\mathcal{T}})$. Thus it is easy to show that $\widehat{\mathcal{I}}$ is a model of every $K \sqsubseteq C \in \mathcal{T}_{\mathsf{f}}^i$, using Points 1 and 2 for the cases $C = \exists r.K'$ and $C = (\leqslant 1\ r\ K')$, and deriving the remaining cases from $\widehat{\mathcal{I}} \models \widehat{\mathcal{T}}$.

□

Finally, Point 1 of Lemma 4.5 implies the required statement

Lemma 4.6. $\widehat{\mathcal{I}} \models \mathcal{T}_{\mathsf{f}}^{\max}$.

4.3 Finite Model Subsumption and ABox Consistency

The statement from Theorem 4.1 is formulated only for finite concept satisfiability. However, our consequence driven approach can also be used to decide finite subsumption via the usual reduction to finite satisfiability. Specifically, $\mathcal{T} \models_{\mathsf{fin}} A \sqsubseteq B$ iff A_0 is finitely unsatisfiable w.r.t. the TBox $\mathcal{T} \cup \{A_0 \sqsubseteq A, A_0 \sqcap B \sqsubseteq \bot\}$, where A_0 is a fresh concept name.

Let us consider the following Example 9.

Example 10. Consider the TBox $\mathcal{T}_1$ from Example 9 and, let $\mathcal{T}_2$ be the union of $\mathcal{T}_1$ and the following axioms:

$$A \sqsubseteq \exists r.(A \sqcap B_1), \tag{4.13}$$
$$A \sqsubseteq \exists r.(A \sqcap B_2), \tag{4.14}$$
$$B_1 \sqcap B_2 \sqsubseteq \bot \tag{4.15}$$

The calculus derives $A \sqsubseteq \bot$, thus A is finitely unsatisfiable w.r.t. $\mathcal{T}_2$: [2]

$$A \sqcap B_i \sqsubseteq A \qquad \text{from } \mathbf{R1} \tag{4.16}$$
$$A \sqsubseteq \exists r.(A \sqcap B_1 \sqcap B_2) \qquad \text{from } (4.12)\text{–}(4.14), (4.16), \mathbf{R7} \tag{4.17}$$
$$A \sqsubseteq \bot \qquad \text{from } (4.15), (4.17), \mathbf{R6} \tag{4.18}$$

$\overline{\wedge}$

[2] A is obviously satisfiable w.r.t. $\mathcal{T}'$ in unrestricted models.

$$\textbf{R10}\ \frac{K(a) \quad K \sqsubseteq A}{A(a)} \qquad \textbf{R11}\ \frac{K(a) \quad r(a,b) \quad K \sqsubseteq \forall r.K'}{K'(b)}$$

$$\textbf{R12}\ \frac{\begin{array}{c} K_1(a) \quad K_2(a) \quad r(a,b) \quad K(b) \quad K_1 \sqsubseteq (\leqslant 1\ r\ A) \\ K_2 \sqsubseteq \exists r.K' \quad K \sqsubseteq A \quad K' \sqsubseteq A \end{array}}{K'(b)}$$

Table 4.2: Inference rules for ABox reasoning

Actually, from Theorem 3.14, and Theorems 3.15 and 4.1 we can conclude the following.

Theorem 4.7. *Let $\mathcal{T}$ be a Horn-$\mathcal{ALCFI}$ TBox, $\widehat{\mathcal{T}}$ the result of saturating $\mathcal{T}$ using the rules* **R1**–**R9**, *and $K \sqsubseteq C$ a Horn-$\mathcal{ALCFI}$ axiom. The following are equivalent:*

1. $\mathcal{T} \models_{\mathsf{fin}} K \sqsubseteq C$
2. $\mathcal{T}_{\mathsf{f}} \models K \sqsubseteq C$
3. $\widehat{\mathcal{T}} \models K \sqsubseteq C$

In order to extend our algorithm to ABox consistency we propose the additional rules shown in Figure 4.2. Instead of starting with only a TBox $\mathcal{T}$, the algorithm now takes as input an ontology $(\mathcal{T}, \mathcal{A})$, where $\mathcal{A}$ is an ABox, and then exhaustively applies rules **R1** to **R12**. Essentially, **R10** to **R12** *saturate* the ABox $\mathcal{A}$ w.r.t. the inclusions in the (saturated) TBox $\mathcal{T}$. In rules **R10** to **R12**, $K(a)$ is an abbreviation for $A_1(a) \ \cdots \ A_k(a)$ when $K = \{A_1, \ldots, A_k\}$. Recall that rules **R1** and **R2** only apply when the conjunction in their precondition occurs in the partially completed TBox. For the extension with ABoxes, an additional way for K to *occur* is that, for some ABox individual a, $K = \{A \mid A(a)$ is in the partial completion$\}$. It is easy to see that rule application still terminates after exponentially many steps. Let $(\widehat{\mathcal{T}}, \widehat{\mathcal{A}})$ be the ontology generated after the saturation step. The algorithm is sound and complete in the following sense.

Lemma 4.8. *$\mathcal{A}$ is finitely consistent w.r.t. $\mathcal{T}$ iff for every $K(a) \in \widehat{\mathcal{A}}$, $K \sqsubseteq \bot \notin \widehat{\mathcal{T}}$.*

Proof. To prove this, one updates the construction of $\widehat{\mathcal{I}}$ by starting with an initial interpretation defined by setting $\Delta^{\widehat{\mathcal{I}}} = \mathsf{Ind}(\mathcal{A})$, $r^{\widehat{\mathcal{I}}} = \{(a,b) \mid r(a,b) \in \mathcal{A}\}$, and $A^{\widehat{\mathcal{I}}} = \{\ a \in \mathsf{Ind}(\mathcal{A}) \mid A(a) \in \widehat{\mathcal{A}}\ \}$. The rest of the construction of $\widehat{\mathcal{I}}$ is as before. It is not hard to adapt the proof of Lemma 4.2 to show that $\widehat{\mathcal{I}}$ satisfies all inclusions and assertions in $\widehat{\mathcal{T}}$ and $\widehat{\mathcal{A}}$, respectively. Finally, the proof to show that $\widehat{\mathcal{I}}$ is a model of $\mathcal{T}_{\mathsf{f}}$ goes through without modification. □

Apart from providing a basis for practical implementations, our algorithm also yields an ExpTime upper bound for finite model reasoning in Horn-$\mathcal{ALCFI}$. This result is known from

[94], where it is shown that ABox consistency in the non-Horn version of $\mathcal{ALCFI}$ is in ExpTime. A matching lower bound can be derived from [15] where an ExpTime lower bound is established for unrestricted subsumption in the $\mathcal{ELI}$ fragment of Horn-$\mathcal{ALCFI}$; the proof can easily be adapted to finite satisfiability.

Theorem 4.9. *Finite satisfiability and finite ABox consistency in Horn-$\mathcal{ALCFI}$ are* ExpTime*-complete.*

Conclusions

The procedure presented in this Chapter establishes a promising foundation for actual implementations of finite-model reasoning on Horn-$\mathcal{ALCQI}$ and, via the reduction in Section 3.5, in Horn-$\mathcal{ALCFI}$. Indeed, Proposition 3.29 enables the use of our consequence-based procedure for deciding finite satisfiability (and subsumption) in Horn-$\mathcal{ALCQI}$.

However, it is not immediately obvious how to extend the translation (3.24) in page 78 and Proposition 3.29 to ABox consistency and instance checking. We believe, though, that it is not too hard to modify the proof of Theorem 3.24 for Horn-$\mathcal{ALCQI}$, to adapt the consequence-based procedure to allow a direct treatment of Horn-$\mathcal{ALCQI}$ TBoxes without prior reduction to Horn-$\mathcal{ALCFI}$.

Chapter

5 Query Answering under the Finite Model Assumption

In this chapter, we study ontological query answering under the finite model assumption in the case where queries are positive existential queries (PEQs) and TBoxes are formulated in Horn-$\mathcal{ALCFI}$. The plan is as follows. In Section 5.1, we describe a general strategy for showing that finite PEQ answering can be reduced to unrestricted PEQ answering by reversing finmod-cycles in the TBox. This result enables to use algorithms for unrestricted PEQ answering also in the finite case. Since the proof that this reduction is as required turned out to be quite challenging, we devote Sections 5.2 and 5.3 to present the technical results that allow to show the correctness the reduction. Establishing the correctness of the reduction also allows us to provide complexity boundaries for the problem of ontological query answer under the finite model assumption. In particular, we show that finite PEQ answering w.r.t. Horn-$\mathcal{ALCFI}$ TBoxes is ExpTime-complete regarding combined complexity, and PTime-complete regarding data complexity, we present the discussion on complexity results in Section 5.4.

5.1 Reducing Finite OQA to Unrestricted OQA in Horn-$\mathcal{ALCFI}$

In this section, our main objective is to prove that using cycle reversion we can also reduce ontological query answering on finite models to ontological query answering on unrestricted models. More precisely, for a given ontology $(\mathcal{T}, \mathcal{A})$ and PEQ q, we show that deciding whether $(\mathcal{T}, \mathcal{A}) \models_{\mathsf{fin}} q$ is equivalent to decide whether $(\mathcal{T}_\mathsf{f}, \mathcal{A}) \models q$, where $\mathcal{T}_\mathsf{f} = \mathsf{finClosure}(\mathcal{T})$ is defined as in Section 3.3.

Note that this result is not a direct consequence of Theorem 3.15 (cf. Chapter 3) since PEQs are more expressive than Horn-$\mathcal{ALCFI}$ axioms. We aim then to prove the following theorem.

Theorem 5.1. *Let $\mathcal{T}$ be a Horn-$\mathcal{ALCFI}$ TBox and $\mathcal{A}$ an ABox that is finitely consistent w.r.t. $\mathcal{T}$. We have that, for any PEQ q, the following holds*

$$(\mathcal{T}, \mathcal{A}) \models_{\mathsf{fin}} q \quad \textit{iff} \quad (\mathcal{T}_\mathsf{f}, \mathcal{A}) \models q$$

We begin by proving the easy direction:"$\Leftarrow$".

Proof of Theorem 5.1 "$\Leftarrow$:" We actually prove the equivalent statement: $(\mathcal{T},\mathcal{A}) \not\models_{\mathsf{fin}} q$ implies $(\mathcal{T}_{\mathsf{f}},\mathcal{A}) \not\models q$. Indeed, assume $(\mathcal{T},\mathcal{A}) \not\models_{\mathsf{fin}} q$, then there is a finite model $\mathcal{I}$ of $(\mathcal{T},\mathcal{A})$ such that $\mathcal{I} \not\models q$. Since by Theorem 3.14, every finite model of $\mathcal{T}$ is also a model of $\mathcal{T}_{\mathsf{f}}$, then $\mathcal{I}$ is also a witness for $(\mathcal{T}_{\mathsf{f}},\mathcal{A}) \not\models q$. □

We now concentrate on the proof of the "$\Rightarrow$" direction. In particular, we use the so-called (possibly infinite) *canonical model of* $(\mathcal{T},\mathcal{A})$.

Definition 5.1. Let $(\mathcal{T},\mathcal{A})$ be a satisfiable Horn-$\mathcal{ALCFI}$ ontology. The *canonical initial interpretation* $\mathcal{U}_0 = (\Delta^{\mathcal{U}_0}, \cdot^{\mathcal{U}_0})$ *of* $(\mathcal{T},\mathcal{A})$ is defined a follows:

$$\begin{aligned}\Delta^{\mathcal{U}_0} &= \mathsf{Ind}(\mathcal{A})\\ A^{\mathcal{U}_0} &= \{a \in \mathsf{Ind}(\mathcal{A}) \mid (\mathcal{T},\mathcal{A}) \models A(a)\}\\ r^{\mathcal{U}_0} &= \{(a,b) \mid r(a,b) \in \mathcal{A}\}\end{aligned}$$

The *canonical model* $\mathcal{U}$ *of* $(\mathcal{T},\mathcal{A})$ is then the result of exhaustively applying the following completion rule starting from $\mathcal{U} = \mathcal{U}_0$:

(**can**) for all $d \in \Delta^{\mathcal{U}}$, and for each role $r \in \mathsf{role}(\mathcal{T})$ such that $\mathsf{tp}_{\mathcal{U}}(d) \rightarrow_r t'$ and $d \notin (\exists r.t')^{\mathcal{U}}$ proceed as follows.

1. Add a fresh element d' to $\Delta^{\mathcal{U}}$;
2. extend $\cdot^{\mathcal{U}}$ in such a way that $(d,d') \in r^{\mathcal{U}}$ and $d' \in A^{\mathcal{U}}$, for all concept names $A \in t'$.

△

The following properties of $\mathcal{U}$ are well-known, indeed these are the reason why $\mathcal{U}$ is called canonical [51, 88, 104].

Lemma 5.2. *Let $(\mathcal{T},\mathcal{A})$ be a satisfiable Horn-$\mathcal{ALCFI}$ ontology, and $\mathcal{U}$ the canonical model of $(\mathcal{T},\mathcal{A})$, then the following hold.*

1. *$\mathcal{U}$ is indeed a model of $(\mathcal{T},\mathcal{A})$;*
2. *for any PEQ q, we have that $(\mathcal{T},\mathcal{A}) \models q$ iff $\mathcal{U} \models^{\pi} q$.*

Before we proceed with our proof, we introduce the notions of *homomorphism* and *simulation* between interpretations.

Definition 5.2. Let $\mathcal{I}_1 = (\Delta^{\mathcal{I}_1}, \cdot^{\mathcal{I}_1})$, and $\mathcal{I}_2 = (\Delta^{\mathcal{I}_2}, \cdot^{\mathcal{I}_2})$ be interpretations over the same vocabulary.

A *homomorphism from* $\mathcal{I}_1$ *to* $\mathcal{I}_2$ is a function $h : \Delta^{\mathcal{I}_1} \to \Delta^{\mathcal{I}_2}$ such that

1. $h(a) = a$ for all $a \in \mathsf{N_I}$;
2. $d \in A^{\mathcal{I}_1}$ implies $h(d) \in A^{\mathcal{I}_2}$ for all concept names A;
3. $(d, e) \in r^{\mathcal{I}_1}$ implies $(h(d), h(e)) \in r^{\mathcal{I}_2}$ for all (possibly inverse) roles r.

A *simulation of* $\mathcal{I}_1$ *in* $\mathcal{I}_2$ is a relation $\rho \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$ such that for all $(d, e) \in \rho$ the following conditions are satisfied:

1. for each $a \in \mathsf{N_I} \cap \Delta^{\mathcal{I}_1}$, we have $(a, a) \in \rho$.
2. if $d \in A^{\mathcal{I}_1}$, then $e \in A^{\mathcal{I}_2}$;
3. if $(d, d') \in r^{\mathcal{I}_1}$ for some (possibly inverse) role r, then there is an $e' \in \Delta^{\mathcal{I}_2}$ with $(e, e') \in r^{\mathcal{I}_2}$ and $(d', e') \in \rho$.

We write $(\mathcal{I}_1, d) \preceq (\mathcal{I}_2, e)$ if there is a simulation ρ of $\mathcal{I}_1$ in $\mathcal{I}_2$ such that $(d, e) \in \rho$. △

We will use Point 2 of Lemma 5.2 to establish the "⇒" direction of Theorem 5.1. More precisely, we will prove that $(\mathcal{T}, \mathcal{A}) \models_{\mathsf{fin}} q$ implies $\mathcal{U} \models q$, where $\mathcal{U}$ is the canonical model of $(\mathcal{T}_\mathsf{f}, \mathcal{A})$. Consider the contraposition of the statement. Let us assume, for a fixed query q, that $\mathcal{U}$ does not have a match for q. We aim at the following.

(∗) We construct a finite model $\mathcal{J}'$ of $(\mathcal{T}, \mathcal{A})$ such that every substructure of $\mathcal{J}'$ that is sufficiently large to contain a match for the query q can be homomorphically embedded into $\mathcal{U}$.

Since PEQs are preserved under homomorphisms, we obtain that $\mathcal{J}'$ does not have a match for q, and hence we can conclude that $(\mathcal{T}, \mathcal{A}) \not\models_{\mathsf{fin}} q$ as required. We now work towards the construction of such finite model. We start by formalizing the notion of the substructure described above.

Definition 5.3. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation, and let $\Delta' \subseteq \Delta^{\mathcal{I}}$ be a subset of the domain of $\mathcal{I}$. The *sub-interpretation* $\mathcal{I}'$ *of* $\mathcal{I}$ *induced by* Δ' is the interpretation $\mathcal{I}' = (\Delta', \cdot^{\mathcal{I}'})$ such that $(\cdot^{\mathcal{I}'})$ is the restriction of $(\cdot^{\mathcal{I}})$ to Δ'.

For $n > 0$, an *n-substructure* of $\mathcal{I}$ is the sub-interpretation $\mathcal{I}'$ of $\mathcal{I}$ induced by a set $\Delta^{\mathcal{I}'} \subseteq \Delta^{\mathcal{I}}$ such that $\#\Delta^{\mathcal{I}'} \leq n$.

△

We actually show the following that in particular will provide the finite model from (∗).

Proposition 5.3. *For every $n > 0$, there is a finite model $\mathcal{J}'$ of $(\mathcal{T}, \mathcal{A})$ such that there is a homomorphism from any n-substructure $\mathcal{J}'_n$ of $\mathcal{J}'$ to the canonical model $\mathcal{U}$ of $(\mathcal{T}_\mathsf{f}, \mathcal{A})$.*

Recall that we aim to show that $(\mathcal{T}, \mathcal{A}) \models_\mathsf{fin} q$ implies $\mathcal{U} \models q$. Indeed, assume $(\mathcal{T}, \mathcal{A}) \models_\mathsf{fin} q$ and let n_0 be the number of variables in q. Now, by Proposition 5.3, there is a finite model $\mathcal{J}'$ such that there is a homomorphism from any n_0-substructure $\mathcal{J}'_{n_0}$ of $\mathcal{J}'$ to the canonical model $\mathcal{U}$ of $(\mathcal{T}_\mathsf{f}, \mathcal{A})$. By assumption, $\mathcal{J}' \models q$. In particular, the latter implies that there is a match π of q in an n_0-substructure $\mathcal{J}'_{n_0}$ of $\mathcal{J}'$. Thus a match of q in $\mathcal{U}$ can be found by composing π with the homomorphism h from $\mathcal{J}'_{n_0}$ to $\mathcal{U}$ granted by Proposition 5.3. This would conclude the proof of the "⇒" direction of Theorem 5.1.

From the discussion above, one can see that the proof of the "⇒" direction of Theorem 5.1 relies on the construction of the finite model $\mathcal{J}'$ from Proposition 5.3 for a particular n_0 that depends on q. Therefore, in what follows, we concentrate on the construction of such $\mathcal{J}'$. To this aim, we modify the construction of finite models in Definition 3.12 (summarized in Table 5.1 for convenience), which in turn is an extension of the construction detailed in Section 3.3.1.

Note that the finite model construction in Table 5.1 needs not satisfy the condition formulated for $\mathcal{J}'$ in Proposition 5.3. Let $\mathcal{I}$ be the finite model of $(\mathcal{T}_\mathsf{f}, \mathcal{A})$ constructed as in Table 5.1. We encounter then the following two problems:

Problem 1. $\mathcal{I}$ can contain paths of length $\leq n_0$ that do not exist in $\mathcal{U}$.

Problem 2. $\mathcal{I}$ can contain cycles that do not exclusively consist of ABox elements, while no such cycles are present in $\mathcal{U}$.

Let us start by discussing **Problem 1**. Indeed, if such a path exists, then one cannot find the homomorphism required in Proposition 5.3 from the substructure containing that path to $\mathcal{U}$. In other words, there is a query q that can *distinguish* between $\mathcal{I}$ and $\mathcal{U}$. There are two sources for such paths in $\mathcal{I}$: applications of rules (**c2**) and (**c3**) (cf. Table 5.1 above). Indeed, these completion rules can introduce paths of the form

$$d_1, r_1, \ldots, r_n, d, s_m^-, e_{m-1}, \ldots, s_1^-, e_1$$

with $n, m \geq 1$ and such that

$$\mathsf{tp}_\mathcal{I}(d_1) \to_{r_1} \mathsf{tp}_\mathcal{I}(d_2) \ldots \to_{r_n} \mathsf{tp}_\mathcal{I}(d); \qquad \text{and}$$
$$\mathsf{tp}_\mathcal{I}(e_1) \to_{s_1} \mathsf{tp}_\mathcal{I}(e_2) \ldots \to_{s_m} \mathsf{tp}_\mathcal{I}(d).$$

Recall that completion rule (**c2**) is responsible for handling ∃-requirements of the form $t \; {}^1{\leftrightarrow}{}^1_r$, by adding the necessary instances of the types in the class P of t (and t'), so as to keep them equal in number and respecting the functionality of roles; while (**c3**) ensures that ∃-requirements of the form $t \to_r t'$ are satisfied, by reusing the existing instance of type t' introduced at the beginning of the construction.

Now, let us consider an example to illustrate both **Problems 1** and **2**.

The *initial interpretation* $\mathcal{I}$ is defined as follows:

$$\begin{aligned} \Delta^{\mathcal{I}} &= \mathsf{Ind}(\mathcal{A}) \cup \{\ d_t \mid t \in \mathsf{TP}(\mathcal{T}_{\mathsf{f}})\ \} \\ A^{\mathcal{I}} &= \{\ a \in \mathsf{Ind}(\mathcal{A}) \mid A \in \mathsf{tp}_{\mathcal{A}}(a)\ \} \cup \{d_t \mid A \in t\ \} \\ r^{\mathcal{I}} &= \{\ (a,b) \mid r(a,b) \in \mathcal{A}\ \} \end{aligned} \tag{5.1}$$

The *finite completion of* $\mathcal{A}$ *w.r.t.* $\mathcal{T}_{\mathsf{f}}$ is the finite interpretation obtained by applying exhaustively the following rules to $\mathcal{I}$, and giving preference to applications of a rule (**ci**) over those of (**cj**), whenever $i < j$.

(**c1**) Choose an element $d \in \Delta^{\mathcal{I}}$ such that $\mathsf{tp}_{\mathcal{I}}(d) \rightarrow^1_r t$, $t \not\rightarrow^1_{r^-} \mathsf{tp}_{\mathcal{I}}(d)$, and $d \notin (\exists r.t)^{\mathcal{I}}$, then

- add a fresh element e to Δ, and
- modify the extension of concept and role names such that $\mathsf{tp}_{\mathcal{I}}(e) = t$ and $(d,e) \in r^{\mathcal{I}}$.

(**c2**) Choose a type class P that is minimal w.r.t. the order $\prec^+$, a $\lambda = t\ {}^1{\leftrightarrow}^1_r\ t'$ with $t \in P$, and an element $d \in \Delta^{\mathcal{I}}$ with $\exists$-requirement $t\ {}^1{\leftrightarrow}^1_r\ t'$; and let $n_{\max} = \max\{\#s^{\mathcal{I}} \mid s \in P\}$, for each $s \in P$, take a fresh set of domain elements

$$\Delta_s := \{d_{s,i} \mid \#s^{\mathcal{I}} < i \leq n_{\max}\}.$$

For each $\lambda = s\ {}^1{\leftrightarrow}^1_r\ s'$ with $s \in P$, let

$$X^{\mathcal{I}}_{\lambda,1} = s^{\mathcal{I}} \setminus (\exists r.s')^{\mathcal{I}} \qquad \text{and} \qquad X^{\mathcal{I}}_{\lambda,2} = s'^{\mathcal{I}} \setminus (\exists r^-.s)^{\mathcal{I}}.$$

and choose a bijection π_λ between $X^{\mathcal{I}}_{\lambda,1} \cup \Delta_s$ and $X^{\mathcal{I}}_{\lambda,2} \cup \Delta_{s'}$, then

- add the elements $\biguplus_{s \in P} \Delta_s$ to Δ;
- for each $\lambda = s\ {}^1{\leftrightarrow}^1_r\ s'$ with $s, s' \in P$, extend $r^{\mathcal{I}}$ with π_λ, and
- interpret concept names so that $\mathsf{tp}_{\mathcal{I}}(d) = s$ for each $d \in \Delta_s$, and $s \in P$.

(**c3**) Choose an element $d \in \Delta^{\mathcal{I}}$ such that $\mathsf{tp}_{\mathcal{I}}(d) \rightarrow_r t$, $\mathsf{tp}_{\mathcal{I}}(d) \not\rightarrow^1_r t$, and $d \notin (\exists r.t)^{\mathcal{I}}$. Let $d_t \in \Delta^{\mathcal{I}}$ be the element introduced for type t in the initialization step (5.1), then, add (d, d_t) to $r^{\mathcal{I}}$.

Table 5.1: Finite model construction for Horn-$\mathcal{ALCFI}$ ontologies from Definition 3.12

Example 11. Assume $\mathcal{A}$ is the following ABox:

$$\mathcal{A} = \{\ B_1(a), D_1(a), B_2(b), D_2(b)\ \};$$

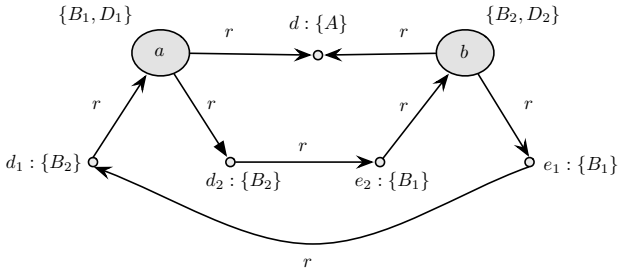
and that $\mathcal{T}$ is the TBox containing the following axioms:

$$\begin{aligned} &B_1 \sqsubseteq \exists r.A, && B_2 \sqsubseteq \exists r.A, \\ &B_1 \sqsubseteq \exists r.B_2, && B_2 \sqsubseteq (\leqslant 1\ r^-\ B_1), \\ &B_2 \sqsubseteq \exists r.B_1, && B_1 \sqsubseteq (\leqslant 1\ r^-\ B_2). \end{aligned}$$

Observe that $\mathsf{TP}(\mathcal{T}_\mathsf{f}) \supseteq \{\ \{B_1, D_1\}, \{B_2, D_2, \}, \{B_1\}, \{B_2\}, \{A\}\ \}$. Further, we have the following relations among the types for $\mathcal{T}_\mathsf{f}$:

$$\begin{aligned} &\{B_1, D_1\} \to_r \{A\}, \\ &\{B_2, D_2\} \to_r \{A\}, \\ &\quad \{B_i\} \to_r \{A\}, \quad \text{for } i \in \{1, 2\} \qquad \text{and} \\ &\quad \{B_1\}\ {}^1{\leftrightarrow}^1_r\ \{B_2\}. \end{aligned}$$

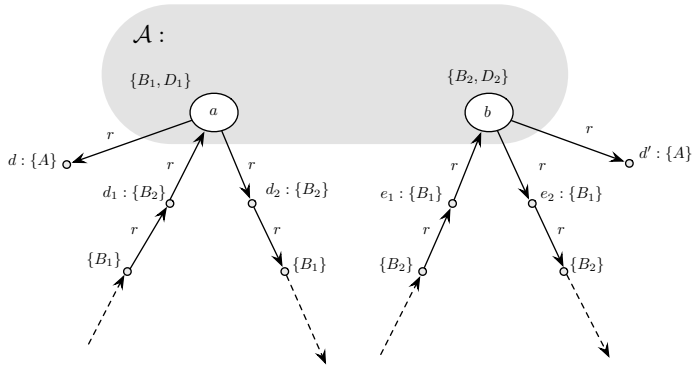
When constructing the finite model $\mathcal{I}$ of $\mathcal{T}, \mathcal{A}$ as in Table 5.1, the initial interpretation contains one object for each individual name in $\mathcal{A}$, and one object for each type for $\mathcal{T}_\mathsf{f}$. After some applications of the completion steps, we get the following:



Note that one problematic path in $\mathcal{I}$ is a, r, d, r^-, b, which is introduced by two (distinct) applications of (**c3**) to a and b respectively. Other problematic path is, for example, a, r, d_2, r, e_2, r, b, in this case the edge $(d_2, e_2) \in r^\mathcal{I}$ is introduced by an application of (**c2**).
Now, consider the following queries:

$$\begin{aligned} q_1 &= \exists x_1, x_2, x_3.r(x_1, x_2,) \wedge r(x_3, x_2) \qquad \text{and} \\ q_2 &= \exists y_1, y_2, y_3, y_4.D_1(y_1) \wedge r(y_1, y_2) \wedge r(y_2, y_3) \wedge r(y_3, y_4) \wedge D_2(y_4). \end{aligned}$$

We have that $\mathcal{I} \models q_1$ and $\mathcal{I} \models q_2$, while $\mathcal{U} \not\models q_1$ and $\mathcal{U} \not\models q_2$, where $\mathcal{U}$ is the canonical model of $\mathcal{T}_\mathsf{f}$ and $\mathcal{A}$ depicted below.



Furthermore, observe that $\mathcal{I}$ (as well as every other finite model of $\mathcal{T}$) contains cycles that cannot be avoided at all. Then for example the following cyclic query can also distinguish between $\mathcal{I}$ and $\mathcal{U}$

$$q_3 = \exists y_1, y_2, y_3, y_4, y_5, y_6 . r(y_1, y_2) \wedge r(y_2, y_3) \wedge r(y_3, y_4) \wedge r(y_4, y_5), r_(y_5, y_6), r(y_6, y_1)$$

$\overline{\wedge}$

We solve separately **Problems 1** and **2** to obtain the desired model $\mathcal{J}'$ from Proposition 5.3, as follows:

- **To solve Problem 1**. We will modify the definitions of the completions steps (**c2**) and (**c3**).

- **To solve Problem 2**. We eliminate 'small' cycles in the model obtained using these modified completion steps.

To characterize the presence of the 'problematic paths' related with **Problem 1**, (illustrated in the example above), we introduce the notion of *bounded simulations* which can be understood as a weakening of simulations.

Definition 5.4. Let $\mathcal{I}_1 = (\Delta^{\mathcal{I}_1}, \cdot^{\mathcal{I}_1})$ and $\mathcal{I}_2 = (\Delta^{\mathcal{I}_2}, \cdot^{\mathcal{I}_2})$ be interpretations. A *bounded simulation of* $\mathcal{I}_1$ *in* $\mathcal{I}_2$ is a relation $\rho \subseteq \Delta^{\mathcal{I}_1} \times \mathbb{N} \times \Delta^{\mathcal{I}_2}$ such that for all $(d, i, e) \in \rho$, the following conditions are satisfied:

1. if $d \in A^{\mathcal{I}_1}$, then $e \in A^{\mathcal{I}_2}$;
2. if $i > 0$ and $(d, d') \in r^{\mathcal{I}_1}$ for some (possibly inverse) role r, then there is an $e' \in \Delta^{\mathcal{I}_2}$ with $(e, e') \in r^{\mathcal{I}_2}$ and $(d', i-1, e') \in \rho$.

We write $(\mathcal{I}_1, d) \preceq_k (\mathcal{I}_2, e)$, for $d \in \Delta^{\mathcal{I}_1}$ and $e \in \Delta^{\mathcal{I}_2}$, if there is a bounded simulation of $\mathcal{I}_1$ in $\mathcal{I}_2$ such that $(d, k, e) \in \rho$ and for all $a \in \mathsf{N_I} \cap \Delta^{\mathcal{I}_1}$, we have $(a, k, a) \in \rho$. Then $\mathcal{I}_1 \preceq_k \mathcal{I}_2$ denotes that for every $d \in \Delta^{\mathcal{I}_1}$, there is an $e \in \Delta^{\mathcal{I}_2}$ with $(\mathcal{I}_1, d) \preceq_k (\mathcal{I}_2, e)$. We write $(\mathcal{I}_1, d) \sim_k (\mathcal{I}_2, e)$ if $(\mathcal{I}_1, d) \preceq_k (\mathcal{I}_2, e)$ and vice versa.

$\triangle$

Then, "eliminating the problematic paths" (thus solving **Problem 1**) means that we can show the following.

Proposition 5.4. *Let $(\mathcal{T}, \mathcal{A})$ be a finitely satisfiable ontology, and let $\mathcal{U}$ be the canonical model of $(\mathcal{T}_\mathsf{f}, \mathcal{A})$. For every $n_0 > 0$, there is a finite model $\mathcal{I}_{n_0}$ of $\mathcal{A}$ and $\mathcal{T}$ such that $\mathcal{I}_{n_0} \preceq_{n_0} \mathcal{U}$.*

To remove the undesired paths illustrated by query q_1 in Example 11 above, we will modify the construction of $\mathcal{I}$, as presented in Table 5.1, by replacing the elements d_t, $t \in \mathsf{TP}(\mathcal{T}_\mathsf{f})$ that are introduced at the beginning of the construction of $\mathcal{I}$ and used as 'targets' for role edges introduced by applications of (**c3**). In the modified construction, we instead introduce one (**c3**)-target for each n_0-*simulation type*. An n_0-*simulation type* is an equivalence class of $\sim_{n_0}$ on the set of all pointed interpretations $(\mathcal{I}_1, d)$. The effect of this modification in Example 11 is that the two $\exists$-requirements handled by (**c3**) would no longer be witnessed by the same d because the 1-simulation type of the witnesses are different (one has an r-predecessor in $B_1 \sqcap D_1$ and the other in $B_2 \sqcap D_2$). Since simulations need only to consider symbols that occur in the (fixed) ABox $\mathcal{A}$ and (fixed) TBox $\mathcal{T}$, there are only finitely many n_0-simulation types and thus finiteness of $\mathcal{I}$ is not compromised.

Further, as mentioned in the strategy above, we also need to modify rule (**c2**). In particular, we modify it so that the sequences

$$\mathsf{tp}_\mathcal{I}(e_0)\ {}^1{\leftrightarrow}^1_{r_1}\ \cdots\ {}^1{\leftrightarrow}^1_{r_{k-1}}\ \mathsf{tp}_\mathcal{I}(e_k) \tag{5.2}$$

are of length exceeding n_0 and thus the problem illustrated by query q_2 in Example 11, which involves both ends of the sequence, is not 'visible' in n_0-substructures. We also include an initial piece of the canonical model $\mathcal{U}$ of $(\mathcal{T}_\mathsf{f}, \mathcal{A})$ of depth n_0 in the initial version of $\mathcal{I}$ to avoid the undesired 'shortcuts' between ABox elements, illustrated also in Example 11. We will address the construction of finite models complying with Proposition 5.4 in Section 5.2.

To solve **Problem 2** and thus obtain the desired $\mathcal{J}'$ from Proposition 5.3, we have to eliminate all non-ABox-cycles of size at most n_0 in the model $\mathcal{I}_{n_0}$ delivered by Proposition 5.4, that is, the model solving **Problem 1**. Therefore, after such elimination, any PEQ q with at most n_0 variables cannot distinguish between $\mathcal{J}'$ and the canonical model $\mathcal{U}$ of $(\mathcal{T}_\mathsf{f}, \mathcal{A})$. We will obtain $\mathcal{J}'$ by taking the product of $\mathcal{I}_{n_0}$ with a suitable finite group of large girth, a technique championed by Otto [106]. We will explain in more detail in Section 5.3 how this technique is implemented, and provide the specific construction of finite models required to prove Proposition 5.3.

5.2 Constructing n-similar Finite Models

In this section, we describe the construction of a finite interpretation satisfying the conditions in Proposition 5.4, and thus solving **Problem 1**.

Let us start by fixing, for the remaining of this section, a finitely satisfiable TBox $\mathcal{T}$, and ABox $\mathcal{A}$ and $n_0 > 0$. We construct a finite model of $(\mathcal{T}, \mathcal{A})$ by modifying the construction from

Section 3.3 (cf. Table 5.1 above). In particular, as previously discussed, we modify the initial interpretation and the completion rules (**c2**) and (**c3**). In more detail, we proceed as follows:

(I) We eliminate from the initial interpretation the set of elements $\{d_t \mid t \in \mathsf{TP}(\mathcal{T}_\mathsf{f})\}$; and instead, we include the initial portion of the canonical model $\mathcal{U}$ of $(\mathcal{T}_\mathsf{f}, \mathcal{A})$ corresponding to the truncation of $\mathcal{U}$ at depth n_0.

(II) We then apply only the completion rules (**c1**) and (**c2′**) described below.

(III) To provide targets for applications of the new completion rule (**c3′**), we determine all relevant n_0-simulation types and add them to the interpretation constructed so far.

(IV) Steps (II) and (III) are iterated until no new n_0-simulation types are added.

(V) Lastly, we apply the completion rule (**c3′**) that reuses the latter elements to satisfy the remaining $\exists$-requirements that were not satisfied after the applications of (**c1**) and (**c2′**).

We start by defining a finite interpretation $\mathcal{I}_0$ as follows:

- First, consider the interpretation obtained from the canonical interpretation $\mathcal{U}$ of $\mathcal{T}_\mathsf{f}$ and $\mathcal{A}$ after applying the rule (**can**) to elements of $\mathcal{U}$ up to depth n_0-1. Let us denote with $\mathcal{U}_{n_0}$ such an initial interpretation.
- $\mathcal{I}_0$ is then the result of exhaustively applying the completion rules (**c1**) and the new (**c2′**) starting from $\mathcal{U}_{n_0}$, and giving more preference to applications of rule (**c1**) over those of (**c2′**) —as in Section 3.3.1, see Table 5.1.

In the construction of $\mathcal{I}_0$ above, note that the ABox $\mathcal{A}$ is a substructure of $\mathcal{U}_{n_0}$, and that $\mathcal{U}_{n_0}$ contains all those elements from $\Delta^\mathcal{U}$ that can be reached from an ABox individual by traveling at most n_0 role edges.

The new completion rule (**c2′**) handles an $\exists$-requirement $t\ {}^1{\leftrightarrow}^1_r\ t'$ with P the type class of t in two steps:

1. in the first stage, $\mathcal{I}_0$ is extended by a *partial unraveling* following only the $\exists$-requirements $\lambda = s\ {}^1{\leftrightarrow}^1_r\ s'$, where $s, s' \in P$.
2. Then, in the second stage the necessary objects are added as to keep the instances of types from P equal in number and respect the functionality of roles.

Note that the second stage amounts to an application of the completion rule (**c2**) from Definition 3.10 to the partially unravelled interpretation $\mathcal{I}_0$.

We formally introduce rule (**c2′**) in Table 5.2. Moreover, rule (**c2′**) guarantees two desirable properties: We call an element d in $\mathcal{I}_0$ *old* if it existed before the application of (**c2′**) and *new* otherwise. A *path in* $\mathcal{I}_0$ is a sequence $d_1 r_1 d_2 \cdots d_k r_k d_{k+1}$, with $d_1, \ldots, d_{k+1} \in \Delta^{\mathcal{I}_0}$, $r_1, \ldots, r_k$

(**c2′**) Choose a type class P that is minimal w.r.t. the order $\prec^+$ and such that there is a $\lambda = s\ {}^1\!\leftrightarrow^1_r\ s'$ with $s \in P$, and an element $d \in \Delta^{\mathcal{I}_0}$ with $\mathsf{tp}_{\mathcal{I}_0}(d) = s$ and $d \notin (\exists r.s')^{\mathcal{I}_0}$.

First, extend $\mathcal{I}_0$ by applying the following rule at most n_0 times to each element in $\mathcal{I}_0$:

(**c2′.1**) For every element $e \in \Delta^{\mathcal{I}_0}$ and $\lambda = s\ {}^1\!\leftrightarrow^1_r\ s'$ with $s, s' \in P$ and r such that $\mathsf{tp}_{\mathcal{I}_0}(d) = s$ and $e \notin (\exists r.s')^{\mathcal{I}_0}$ add a new element e'; and extend $(\cdot^{\mathcal{I}_0})$ such that $\mathsf{tp}_{\mathcal{I}_0}(e') = s'$, and $(d, e') \in r^{\mathcal{I}_0}$.

Second, apply the following rule to the resulting interpretation:

(**c2′.2**) Let $n_{\max} = \max\{\#(s^{\mathcal{I}_0}) \mid s \in P\}$, for each $s \in P$, take a fresh set of domain elements

$$\Delta_s := \{d_{s,i} \mid \#(s^{\mathcal{I}_0}) < i \leq n_{\max}\}.$$

For each $\lambda = s\ {}^1\!\leftrightarrow^1_r\ s'$ with $s \in P$, let

$$X^{\mathcal{I}}_{\lambda,1} = s^{\mathcal{I}_0} \setminus (\exists r.s')^{\mathcal{I}_0} \qquad \text{and} \qquad X^{\mathcal{I}}_{\lambda,2} = s'^{\mathcal{I}_0} \setminus (\exists r^-.s)^{\mathcal{I}_0}.$$

and choose a bijection π_λ between $X^{\mathcal{I}}_{\lambda,1} \cup \Delta_s$ and $X^{\mathcal{I}}_{\lambda,2} \cup \Delta_{s'}$, then

- extend $\Delta^{\mathcal{I}_0}$ by adding the elements in $\biguplus_{s \in P} \Delta_s$;
- for each $\lambda = s\ {}^1\!\leftrightarrow^1_r\ s'$ with $s, s' \in P$, extend $r^{\mathcal{I}_0}$ with π_λ, and
- extend $(\cdot^{\mathcal{I}_0})$ in such a way that $\mathsf{tp}_{\mathcal{I}_0}(d) = s$ for each $d \in \Delta_s$.

Table 5.2: Completion rule (**c2′**)

(potentially inverse) roles and $(d_i, d_{i+1}) \in r_i^{\mathcal{I}_0}$ for $1 \leq i \leq k$. A path is *simple* if there are no multiple occurrences of the same object d_i. The definition of (**c2′**) then ensures that:

(**edge**) no edge $(d_1, d_2) \in r^{\mathcal{I}_0}$ is introduced with both d_1, d_2 old;

(**path**) for each new element $d_1 \in \biguplus_s \Delta_s$, there is at most one simple path

$$d_1 r_1 d_2 \cdots d_k r_k d_{k+1}$$

of length at most n_0 such that $d_1, \ldots, d_k$ are new and d_{k+1} is old.

We have now all the required ingredients to define rule (**c2′**) (see Table 5.2). Indeed, (**edge**) and (**path**) follow from the observation that the application of (**c2′.1**) results in tree-shaped substructures of depth n_0 attached to old elements of $\mathcal{I}_0$, and that all edges $(d, d') \in r^{\mathcal{I}_0}$ for some role r added in this step, contain at most one old element. Further, all the edges added by (**c2′.2**) contain only new elements from the sets Δ_s or new elements at level n_0 (i.e., leaves) in the trees introduced by (**c2′.1**).

We now argue that applications of these rules preserve the invariants (**i1**), (**i2′**), and (**i3**) from Section 3.3.1 (see Table 5.3).

(**i1**)	$\mathsf{tp}_{\mathcal{I}}(d) \in \mathsf{TP}(\mathcal{T}_{\mathsf{f}})$ for all $d \in \Delta^{\mathcal{I}}$;
(**i2′**)	if $(d, d') \in r^{\mathcal{I}} \setminus (\mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}))$, then we have $\mathsf{tp}_{\mathcal{I}}(d) \rightarrow_r \mathsf{tp}_{\mathcal{I}}(d')$ or $\mathsf{tp}_{\mathcal{I}}(d') \rightarrow_{r^-} \mathsf{tp}_{\mathcal{I}}(d)$;
(**i3**)	if $\mathcal{T}_{\mathsf{f}} \models K \sqsubseteq (\leqslant 1\ r\ K')$, then $\mathcal{I} \models K \sqsubseteq (\leqslant 1\ r\ K')$.

Table 5.3: Invariants satisfied by the finite model construction

- The initial interpretation $\mathcal{U}_{n_0}$ satisfies the invariants: (**i1**), (**i2′**) are satisfied by construction, and (**i3**) holds because $\mathcal{U} \models \mathcal{T}_{\mathsf{f}}$ (Lemma 5.2).
- To show that all the invariants are preserved by applications of (**c1**) and (**c2′**) the same arguments as for Lemma 3.17 go through.

Furthermore, Proposition 3.21 also holds for $\mathcal{I}_0$, therefore we can prove that the construction of $\mathcal{I}_0$ terminates, and $\Delta^{\mathcal{I}_0}$ is finite.

In the following three lemmas, we will show that bounded simulations of $\mathcal{I}_0$ (and extensions thereof) in $\mathcal{U}$ exist, and therefore provide the arguments for Proposition 5.4 (solving then **Problem 1**). In fact, as we will see, it suffices to consider sub-interpretations of $\mathcal{I}_0$ induced by n_0-*neighborhoods*.

Definition 5.5. Let $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ be an interpretation, and let $d \in \Delta^{\mathcal{J}}$. The n_0-*neighborhood of* d *in* $\mathcal{J}$ is the subset $N^{\mathcal{J}}_{n_0}(d) \subseteq \Delta^{\mathcal{J}}$ such that

$$N^{\mathcal{J}}_{n_0}(d) := \{d' \in \Delta^{\mathcal{J}} \mid \text{there is a path from } d \text{ to } d' \text{in } \mathcal{J} \text{ of length at most } n_0\}.$$

We denote by $\mathcal{J}|^{n_0}_d$ the sub-interpretation of $\mathcal{J}$ induced by $N^{\mathcal{J}}_{n_0}(d)$.

△

We show now the announced result.

Lemma 5.5. $\mathcal{I}_0 \preceq_{n_0} \mathcal{U}$.

Proof. Let $d^* \in \Delta^{\mathcal{I}_0}$. We show that $(\mathcal{I}_0|^{n_0}_{d^*}, d^*) \preceq (\mathcal{U}, e)$ for some $e \in \Delta^{\mathcal{U}}$. Let Δ denote the domain of $\mathcal{I}_0|^{n_0}_{d^*}$. An element $d \in \Delta$ is an *initial element* if $d \in \Delta^{\mathcal{U}_{n_0}}$. A *forward path* is a sequence $d_0 r_0 d_1 \cdots d_{k-1} r_{k-1} d_k$ such that

(a) $(d_i, d_{i+1}) \in r_i^{\mathcal{I}_0|^{n_0}_{d^*}}$ for all $i < k$.

(b) $\mathsf{tp}_{\mathcal{I}_0}(d_i) \rightarrow_{r_i} \mathsf{tp}_{\mathcal{I}_0}(d_{i+1})$ for all $i < k$.

(c) $(r_i^-, d_{i+1}) \neq (r_{i-1}, d_{i-1})$ for $0 < i < k$;

(d) $d_1, \ldots, d_k$ are not initial.

An element $d \in \Delta$ is a *root* if the following two conditions are satisfied:

($\mathbf{r_1}$) if $(d, e) \in r^{\mathcal{I}_0|^{n_0}_{d^*}}$, then both d and e are initial or $\mathsf{tp}_{\mathcal{I}_0}(d) \to_r \mathsf{tp}_{\mathcal{I}_0}(e)$;

($\mathbf{r_2}$) for every forward path $d = d_0 r_0 d_1 \cdots r_{k-1} d_k$ and each $(d_k, e) \in r^{\mathcal{I}_0|^{n_0}_{d^*}}$, we have that $\mathsf{tp}_{\mathcal{I}_0}(d_k) \to_r \mathsf{tp}_{\mathcal{I}_0}(e)$ or $e = d_{k-1}$ and $r = r_{k-1}^-$.

Let us define an order over the elements in Δ. We write $d < e$ if the rule application that created d happened before the one that created e; and we write $d \leqq e$ if (i) $d < e$ or (ii) d and e are initial elements or (iii) d and e have been created in the same application (thus of ($\mathbf{c2'}$)).

Take some element $d_r \in \Delta$ that is minimal w.r.t. $\leqq$, i.e., whenever $d \leqq d_r$, we also have $d_r \leqq d$. We will now show that:

(1) d_r is a root, and

(2) d^* is reachable from d_r on a forward path.

We proceed to show (1) and (2).

- To prove (1), we have to show that d_r satisfies Conditions ($\mathbf{r_1}$) and ($\mathbf{r_2}$) of roots. Let us start with ($\mathbf{r_1}$). Take some arbitrary element e with $(d_r, e) \in r^{\mathcal{I}_0|^{n_0}_{d^*}}$. The choice of d_r yields three cases:

  case 1: both d_r, e are initial. Then ($\mathbf{r_1}$) is satisfied trivially.

  case 2: d_r was created by an application of a completion rule prior to the creation of e. By the construction of $\mathcal{I}_0$, we get that $\mathsf{tp}_{\mathcal{I}_0}(d_r) \to_r \mathsf{tp}_{\mathcal{I}_0}(e)$ since $(d_r, e) \in r^{\mathcal{I}_0|^{n_0}_{d^*}}$.

  case 3: d_r, e were introduced by the same application of a completion rule. By the construction of $\mathcal{I}_0$, we have that d_r, e were introduced by an application of ($\mathbf{c2'}$)
  Hence, we have $\mathsf{tp}_{\mathcal{I}_0}(d_r) \; {}^1{\leftrightarrow}^1_r \; \mathsf{tp}_{\mathcal{I}_0}(e)$.

  To show that d_r satisfies ($\mathbf{r_2}$), let $d_r = d_0 r_0 d_1 \cdots r_{k-1} d_k$ be a forward path and $(d_k, e) \in r^{\mathcal{I}_0|^{n_0}_{d^*}}$. In case the edge $(d_k, e) \in r^{\mathcal{I}_0|^{n_0}_{d^*}}$ was created in an application of ($\mathbf{c2'}$), we have $\mathsf{tp}_{\mathcal{I}_0}(d_k) \; {}^1{\leftrightarrow}^1_r \; \mathsf{tp}_{\mathcal{I}_0}(e)$ and we are done. Otherwise, this edge was created in an application of ($\mathbf{c1}$), and either $\mathsf{tp}_{\mathcal{I}_0}(d_k) \to_r \mathsf{tp}_{\mathcal{I}_0}(e)$ (in which case we are done) or $\mathsf{tp}_{\mathcal{I}_0}(e) \to_{r^-} \mathsf{tp}_{\mathcal{I}_0}(d_k)$. Then the edge $(d_{k-1}, d_k) \in r_{k-1}^{\mathcal{I}_0|^{n_0}_{d^*}}$ was created in an application of ($\mathbf{c1}$) (in which case we must have $e = d_{k-1}$ and $r = r_{k-1}^-$ and are done) or of ($\mathbf{c2'}$).

  In the latter case, we trace that ($\mathbf{c2'}$) application backwards on the path. If all elements $d_r = d_0, \ldots, d_k$ have been created by the same application, we have that $e < d_0$, which contradicts d_r being $\leqq$-minimal. Otherwise, d_0 is an old element of that application, and we consider a simple path $d_0 = d'_0 r'_0 d'_1 \cdots r'_{\ell-1} d'_\ell = d_k$ of length $\leq 2n_0$ from $d_r = d_0 = d'_0$

to $d_k = d'_\ell$. Due to its construction, $\mathcal{I}_0|^{n_0}_{d^*}$ has to contain such a path. Then some edge on this new path must have been created in an application of (**c2′**): otherwise, we would have $d'_0 < d'_1 < \cdots < d'_j > \cdots > d'_{\ell-1} > d'_\ell$, that is, d'_j would have been created in two different (**c1**) rule applications.

Consider the latest such (**c2′**) application in the construction of $\mathcal{I}_0$ and observe that $d'_\ell = d_k$ is old for it (we reuse the argument from above). If $d'_0 = d_0$ is old for it as well, then we get a contradiction as follows. Take the maximal index f such that $d'_0, \ldots, d'_f$ are all old and the minimal index g such that $d'_g, \ldots, d'_\ell$ are all old. Since $f < g$ due to Condition (**edge**) of (**c2′**), there is a middle element $d'_{j'}$ with $j' = \lfloor \frac{f+g}{2} \rfloor$, which has simple paths of length $\leq n_0$ to both old elements d'_f and d'_g. These paths coincide due to Condition (**path**) of (**c2′**), which contradicts the assumption that $d'_0 r'_0 d'_1 \cdots r'_{\ell-1} d'_\ell$ is simple.

Otherwise, if d'_0 had been created in the same (**c2′**) application, then we would get $d_k < d'_0$ which contradicts d_r being $\leqq$-minimal. Finally, if d'_0 had been created after that (**c2′**) application, we would get $d'_1 < d'_0$, again contradicting $\leqq$-minimality of d_r.

- To prove (2), we observe that, by construction of $\mathcal{I}_0|^{n_0}_{d^*}$, there is a simple path

$$d_r = d_0 r_0 d_1 \ldots r_{k-1} d_k = d^*$$

in $\mathcal{I}_0|^{n_0}_{d^*}$ with $k \leq n_0$ such that no element other than possibly d_0 is initial. We pick such a path, and our choice ensures Conditions (a), (c) and (d) of forward paths. This leaves us with showing Condition (b).

Since $(d_i, d_{i+1}) \in r_i^{\mathcal{I}_0|^{n_0}_{d^*}}$ for all $i < k$, we have that either $\mathsf{tp}_{\mathcal{I}_0}(d_i) \rightarrow_{r_i} \mathsf{tp}_{\mathcal{I}_0}(d_{i+1})$ or $\mathsf{tp}_{\mathcal{I}_0}(d_{i+1}) \rightarrow_{r_i^-} \mathsf{tp}_{\mathcal{I}_0}(d_i)$ for all $i < k$. Assume that there is some i with $\mathsf{tp}_{\mathcal{I}_0}(d_{i+1}) \rightarrow_{r_i^-} \mathsf{tp}_{\mathcal{I}_0}(d_i)$ but $\mathsf{tp}_{\mathcal{I}_0}(d_i) \not\rightarrow_{r_i} \mathsf{tp}_{\mathcal{I}_0}(d_{i+1})$, and take the smallest such i. Then the edge $(d_i, d_{i+1}) \in r_i^{\mathcal{I}_0}$ has been introduced in some application of (**c1**), and the previous edge $(d_{i-1}, d_i) \in r_{i-1}^{\mathcal{I}_0}$ has been introduced in some application of (**c2′**) (otherwise, we would have $d_{i+1} = d_{i-1}$ and $r_i = r_{i-1}^-$, contradicting Condition (d) of forward paths). We can now reuse the above argument, tracing back that application of (**c2′**), and derive a contradiction.

Since every initial element is $\leqq$-minimal, (1) and (2) above establish the following:

Claim 3.

- every initial element is a root;
- Δ contains at least one root d_r such that d^* is reachable from d_r on a forward path.

Then, we will define the required (bounded) simulation starting from the roots in Δ. Observe that, by definition (in general) there is no *unique* root in Δ. We use the following notion to ease the definition of the required simulation.

We say that a relation $\rho \subseteq \Delta \times \Delta^{\mathcal{U}}$ is *rooted* if for every $(d, e) \in \rho$, there is a forward path $d_0 r_0 d_1 \cdots r_{k-1} d_k$ and elements $e_0, \ldots, e_k \in \Delta^{\mathcal{U}}$ such that d_0 is a root, $d_k = d$, $e_k = e$, $(d_i, e_i) \in \rho$ for all $i \leq k$ and $(e_i, e_{i+1}) \in r_i^{\mathcal{U}}$ for all $i < k$.

We define a sequence of relations

$$\rho_0 \subseteq \rho_1 \subseteq \cdots \subseteq \Delta \times \Delta^{\mathcal{U}}$$

such that

(†) if $(d, e) \in \rho_i$, then $\mathsf{tp}_{\mathcal{I}_0}(d) = \mathsf{tp}_{\mathcal{U}}(e)$;

(‡) ρ_i is rooted.

Choose a root d_r such that d^* is reachable from d_r by a forward path, whose existence is guaranteed by Point 2 above. Also choose an $e_r \in \Delta^{\mathcal{U}}$ with $\mathsf{tp}_{\mathcal{U}}(e_r) = \mathsf{tp}_{\mathcal{I}_0}(d_r)$, which exists by invariant (**i1**). Then

- set
$$\rho_0 \;=\; \{(d, d) \mid d \in \Delta \text{ is initial}\} \;\cup\; \{(d_r, e_r)\}$$
Note that (†) and (‡) are trivially satisfied since by definition all initial elements belong to $\Delta^{\mathcal{U}_{n_0}} \subseteq \Delta^{\mathcal{U}}$.

- ρ_{i+1} is obtained from ρ_i by doing the following for each $(d, e) \in \rho_i$ and $(d, d') \in r^{\mathcal{I}_0|_{d^*}^{n_0}}$. By (‡), there is a forward path $d_0 r_0 d_1 \cdots r_{k-1} d_k$ and elements $e_0, \ldots, e_k \in \Delta^{\mathcal{U}}$ such that d_0 is a root, $d_k = d$, $e_k = e$, $(d_i, e_i) \in \rho$ for all $i \leq k$ and $(e_i, e_{i+1}) \in r_i^{\mathcal{U}}$ for all $i < k$. By Point 2 above, we can distinguish two cases:

  - $\mathsf{tp}_{\mathcal{I}_0}(d_k) \to_r \mathsf{tp}_{\mathcal{I}_0}(d')$ and it is not true that $d' = d_{k-1}$ and $r = r_{k-1}^-$.
    Then $\mathsf{tp}_{\mathcal{I}_0}(d) \to_r \mathsf{tp}_{\mathcal{I}_0}(d')$. By (†), we have $\mathsf{tp}_{\mathcal{U}}(e) \to_r \mathsf{tp}_{\mathcal{I}_0}(d')$ and thus we find an $e' \in \Delta^{\mathcal{U}}$ with $(e, e') \in r^{\mathcal{U}}$ and $\mathsf{tp}_{\mathcal{U}}(e') = \mathsf{tp}_{\mathcal{I}_0}(d')$. Include (d', e') in ρ_{i+1}. Clearly, (†) is still satisfied. Using that it is not true that $d' = d_{k-1}$ and $r = r_{k-1}^-$, it is also straightforward to show that (‡) is still satisfied.
  - $d' = d_{k-1}$ and $r = r_{k-1}^-$.
    Then we do not need to add an extra tuple to ρ_i since there already is a $(d', e') \in \rho_i$ such that $(e, e') \in r^{\mathcal{U}}$. To see this, recall that e_{k-1} and e_k are such that $(d_{k-1}, e_{k-1}) \in \rho_i$, $(d_k, e_k) \in \rho_i$, and $(e_{k-1}, e_k) \in r_{k-1}^{\mathcal{U}}$. Since $d' = d_{k-1}$ and $r = r_{k-1}^-$, e_{k-1} can serve as the required e'.

Set $\rho = \bigcup_{i \geq 0} \rho_i$. By construction, ρ is a simulation. Since there is a forward path from d_r to d^*, by construction of ρ, there must be some $(d^*, e) \in \rho$. Thus we have shown $(\mathcal{I}_0|_{d^*}^{n_0}, d^*) \preceq (\mathcal{U}, e)$. Note that $\mathsf{tp}_{\mathcal{I}_0}(d^*) = \mathsf{tp}_{\mathcal{U}}(e)$ as required. This finalizes the proof of Lemma 5.5. □

However, $\mathcal{I}_0$ is not yet a model of $(\mathcal{T}, \mathcal{A})$ since some $\exists$-requirements might not be satisfied because completion step (**c3′**) (defined below) has not been applied. Our next step for completing the interpretation $\mathcal{I}_0$ as to obtain the required finite model of $\mathcal{T}$ and $\mathcal{A}$, is to introduce the objects that will be used by (**c3′**) to satisfy the remaining $\exists$-requirements in $\mathcal{I}_0$.

Generating Witnesses for (c3′). We need to introduce fresh elements to the interpretation $\mathcal{I}_0$ that can be used by (**c3′**) to satisfy $\exists$-requirements $t \to_r t'$ for some role r, and such that it does not hold that $t \to_r^1 t'$, which means that we can safely reuse such witnesses without violating the functionality of r^-. Further, recall that we also need to ensure that no 'problematic path' is introduced after the application of (**c3′**). The latter, means that if $\mathcal{I}$ is the obtained interpretation after every possible application of (**c3′**) to $\mathcal{I}_0$, then it holds that $\mathcal{I} \preceq_{n_0} \mathcal{U}$.

As discussed at the beginning of this Section, we first need to identify the relevant n_0-simulation types that need to be realized by the witnesses. In order to identify those types, we extend the finite model $\mathcal{I}_0$ constructed so far to an infinite interpretation $\mathcal{I}_0^+$. While $\mathcal{I}_0^+$ will of course not be part of the finite model that we aim to construct, it will guide the further construction.

We obtain $\mathcal{I}_0^+$ from $\mathcal{I}_0$ by starting with $\mathcal{I}_0^+ = \mathcal{I}_0$, and then exhaustively applying the completion rule (**can**) from Definition 5.1, repeated and rephrased here for convenience:

($*$) for all $d \in \Delta^{\mathcal{I}_0^+}$ and role r such that $\mathsf{tp}_{\mathcal{I}_0^+}(d) \to_{t'}$, w.r.t. $\mathcal{T}_\mathsf{f}$, and $d \notin (\exists r.t')^{\mathcal{I}_0^+}$, add a fresh element d' to $\Delta^{\mathcal{I}_0^+}$, the edge (d, d') to $r^{\mathcal{I}_0^+}$, and d' to the interpretation $A^{\mathcal{I}_0^+}$ of all concept names $A \in t'$.

The desired result presented next follows from Lemma 5.5 and from the construction of $\mathcal{I}_0^+$.

Lemma 5.6. $\mathcal{I}_0^+ \preceq_{n_0} \mathcal{U}$.

Proof. Let $d^* \in \Delta^{\mathcal{I}_0^+}$. We show that $(\mathcal{I}_0^+|_{d^*}^{n_0}, d^*) \preceq (\mathcal{U}, e)$ for some $e \in \Delta^{\mathcal{U}}$. For brevity, we use Δ to denote the domain of $\mathcal{I}_0^+|_{d^*}^{n_0}$. We distinguish three cases:

case 1: d^* is from $\Delta^{\mathcal{I}_0}$.

Then Lemma 5.5 gives us an n_0-bounded simulation ρ of $(\mathcal{I}_0, d^*)$ in $(\mathcal{U}, e)$ for some e. We can extend ρ to the desired n_0-bounded simulation of $(\mathcal{I}_0^+, d^*)$ in $(\mathcal{U}, e)$ by following the applications of the completion rule applied to construct $\mathcal{I}_0^+$ from $\mathcal{I}_0$, and exploiting that $\mathcal{U}$ is constructed by applying the same rule.

case 2: d^* is not from $\Delta^{\mathcal{I}_0}$ and Δ contains elements from $\Delta^{\mathcal{I}_0}$.

Let d_0 be the unique element from Δ that is in $\Delta^{\mathcal{I}_0}$ and can be reached from d^* in $\mathcal{I}_0^+|_{d^*}^{n_0}$ on a path of minimal length.[1] Start with a n_0-bounded simulation ρ of $(\mathcal{I}_0, d_0)$ in $(\mathcal{U}, e)$ for some e (given by Lemma 5.5), restricted to the elements of Δ. Then proceed as in **case 1**.

[1] This element d_0 is unique since $\mathcal{I}_0^+$ extends $\mathcal{I}_0$ by attaching tree-shaped structures to existing elements.

case 3: Δ contains no elements from $\Delta^{\mathcal{I}_0}$.

Exploiting Invariant (**i1**), it is easy to show by induction on the number of rule applications used to construct $\mathcal{I}_0^+$ that for every $d \in \Delta^{\mathcal{I}_0^+}$, there is an $e \in \Delta^{\mathcal{U}}$ with $\mathsf{tp}_{\mathcal{I}_0^+}(d) = \mathsf{tp}_{\mathcal{U}}(e)$. For $d, d' \in \Delta$, we write $d \prec d'$ if d' was created by a later rule appliction than d during the construction of $\mathcal{I}_0^+$ from $\mathcal{I}_0$. Let d_0 be the unique element of Δ that is minimal w.r.t. $\prec$. We start with the initial bounded simulation $\rho = \{(d_0, n_0, e)\}$ for some e with $\mathsf{tp}_{\mathcal{I}_0^+}(d_0) = \mathsf{tp}_{\mathcal{U}}(e)$ and then proceed as in **case 1** above.

□

We now choose one representative $(\mathcal{J}, d) \in S$ of each n_0-simulation type S realized in $\mathcal{I}_0^+$, i.e., such that there is some $d \in \Delta^{\mathcal{I}_0^+}$ with $(\mathcal{I}_0^+, d) \in S$. Then extend $\mathcal{I}_0$ with pairwise disjoint copies of all the chosen representatives. By Lemma 5.6, the resulting interpretation $\mathcal{I}_1$ satisfies $\mathcal{I}_1 \preceq_{n_0} \mathcal{U}$. We treat $\mathcal{I}_1$ as an initial interpretation in the same way as we have treated $\mathcal{U}_0$ as an initial intepretation for constructing $\mathcal{I}_0$ and repeat the application of (**c1**) and (**c2′**) as described above, which results in a completed version of the interpretation $\mathcal{I}_1$. Lemmas 5.5 and 5.6 apply also to $\mathcal{I}_1$ in place of $\mathcal{I}_0$, with the proofs going through without modification. The same is true for the invariants (**i1**) to (**i3**). Since $\mathcal{I}_1^+$ might realize n_0-simulation types that are not realized in $\mathcal{I}_0^+$, we then add copies of the new simulation types. Repeating this process leads to a sequence of finite interpretations $\mathcal{I}_0, \mathcal{I}_1, \ldots$. Since there are only finitely many n_0-simulation types and since the simulation type of added representatives does not change by applying the rules (**c1**) and (**c2′**), this process eventually stabilizes. Call the resulting finite interpretation $\mathcal{I}_\omega$. By what was said above, we have the following.

Lemma 5.7. *$\mathcal{I}_\omega \preceq_{n_0} \mathcal{U}$ and $\mathcal{I}_\omega^+ \preceq_{n_0} \mathcal{U}$.*

The disjoint copies just added will serve as the desired 'targets' for applying (a modified version) of the completion rule (**c3**). Indeed, to construct the desired finite interpretation $\mathcal{I}$, it remains to start with the infinite model $\mathcal{I} = \mathcal{I}_\omega^+$ and exhaustively apply a modified version (**c3′**) below of the completion rule (**c3**).

(**c3′**) If $d \in \Delta^{\mathcal{I}}$, $\mathsf{tp}_{\mathcal{I}}(d) \to_r t$, and $d \notin (\exists r.t)^{\mathcal{I}}$, then by construction of $\mathcal{I}_\omega^+$, we find an element $e \in \Delta^{\mathcal{I}_\omega^+}$ such that $(d, e) \in r^{\mathcal{I}_\omega^+}$ and $\mathsf{tp}_{\mathcal{I}_\omega^+}(e) = t$. By construction of $\mathcal{I}_\omega$, there is an element $e' \in \Delta^{\mathcal{I}_\omega}$ such that $(\mathcal{I}_\omega^+|_e^{n_0}, e)$ and $(\mathcal{I}_\omega|_{e'}^{n_0}, e')$ have the same simulation type. Include in $r^{\mathcal{I}}$ the edge (d, e').

Note that the modified version (**c3′**) of the old (**c3**) preserves all invariants because the same arguments as for the latter rule go through in Lemma 3.17.

Lemma 5.8. *$\mathcal{I}$ is a model of $(\mathcal{T}_{\mathsf{f}}, \mathcal{A})$.*

Proof. It is easy to see that the proof of Proposition 3.19 goes through also for the modified version of $\mathcal{I}$: the essential ingredients of that proof are the invariants (**i1**)–(**i3**), which hold for $\mathcal{I}$ as argued above, plus the argument after the case distinction (1)–(3) for axioms of the form $K \sqsubseteq \exists r.K'$, which is unaffected by our modification of the rules. □

We can now establish the main property that is satisfied by the finite model $\mathcal{I}$ just constructed.

Lemma 5.9. *For every n_0-substructure $\mathcal{I}'$ of $\mathcal{I}$, $\mathcal{I}' \preceq \mathcal{U}$.*

Proof. By Lemma 5.7, it suffices to show that $\mathcal{I} \preceq_{n_0} \mathcal{I}_\omega^+$. We call an edge $(d, e) \in r^\mathcal{I}$ *special* if it was added in the construction of $\mathcal{I}$ from $\mathcal{I}_\omega$, that is, by applying (**c3′**). The *source* of the special edge $(d, e) \in r^\mathcal{I}$ is the element from $\{d, e\}$ that plays the role of d in the formulation of (**c3′**).

Let $d^* \in \Delta^\mathcal{I}$. In the following, we construct an n_0-bounded simulation ρ of $(\mathcal{I}, d^*)$ in $(\mathcal{I}_\omega^+, d^*)$. To assist with the construction of ρ, we associate with every tuple (d, i, e) in the partially constructed ρ an i-bounded simulation $\rho_{d,i,e}$ of $(\mathcal{I}_\omega^+, d)$ in $(\mathcal{I}_\omega^+, e)$ whose purpose is to guide the further construction.

We start with setting $\rho = \{(d^*, n_0, d^*)\}$. As the required n_0-bounded simulation ρ_{d^*,n_0,d^*} of $(\mathcal{I}_\omega^+, d^*)$ in $(\mathcal{I}_\omega^+, d^*)$, we use the identity, that is, the set of all triples (d, i, d) with $d \in \Delta^{\mathcal{I}_\omega^+}$ and $i \leq n_0$. To extend the initial ρ just defined, we distingush three cases.

Assume that $(d, i, e) \in \rho$ with $i > 0$ and $(d, d') \in r^\mathcal{I}$ is non-special. Then $(d, d') \in r^{\mathcal{I}_\omega} \subseteq r^{\mathcal{I}_\omega^+}$ and thus we find a triple $(d', i-1, e') \in \rho_{d,i,e}$ with $(e, e') \in r^{\mathcal{I}_\omega^+}$. Add $(d', i-1, e')$ to ρ and set $\rho_{d',i-1,e'} = \rho_{d,i,e}$.

Now assume that $(d, i, e) \in \rho$ with $i > 0$, $(d, d') \in r^\mathcal{I}$ is special, and d is the source of this edge. Then there is a $d'' \in \Delta^{\mathcal{I}_\omega^+}$ such that $(d, d'') \in r^{\mathcal{I}_\omega^+}$ and $(\mathcal{I}_\omega^+, d'')$ has the same n_0-simulation type as $(\mathcal{I}_\omega, d')$. Then $(\mathcal{I}_\omega^+, d'')$ must also have the same n_0-simulation type as $(\mathcal{I}_\omega^+, d')$. We can thus find an $(i-1)$-bounded simulation ν of $(\mathcal{I}_\omega^+, d')$ in $(\mathcal{I}_\omega^+, d'')$. Since $(d, i, e) \in \rho_{d,i,e}$, there must be an $e'' \in \Delta^{\mathcal{I}_\omega^+}$ with $(d'', i-1, e'') \in \rho_{d,i,e}$ and $(e, e'') \in r^{\mathcal{I}_\omega^+}$. We add $(d', i-1, e'')$ to ρ. The required $(i-1)$-bounded simulation $\rho_{d',i-1,e''}$ of $(\mathcal{I}_\omega^+, d')$ in $(\mathcal{I}_\omega^+, e'')$ is obtained by composing ν with $\rho_{d,i,e}$.

Finally assume that $(d, i, e) \in \rho$ with $i > 0$, $(d, d') \in r^\mathcal{I}$ is special, and e is the source of this edge. Then there is a $\widehat{d} \in \Delta^{\mathcal{I}_\omega^+}$ such that $(\widehat{d}, d') \in r^{\mathcal{I}_\omega^+}$ and $(\mathcal{I}_\omega^+, \widehat{d})$ has the same n_0-simulation type as $(\mathcal{I}_\omega, d)$. Then $(\mathcal{I}_\omega^+, \widehat{d})$ must also have the same n_0-simulation type as $(\mathcal{I}_\omega^+, d)$. We can thus find an i-bounded simulation ν of $(\mathcal{I}_\omega^+, \widehat{d})$ in $(\mathcal{I}_\omega^+, d)$. Composing ν with $\rho_{d,i,e}$, we find an i-bounded simulation η of $(\mathcal{I}_\omega^+, \widehat{d})$ in $(\mathcal{I}_\omega^+, e)$. Since $(\widehat{d}, d') \in r^{\mathcal{I}_\omega^+}$, there must be some $\widehat{e} \in \Delta^{\mathcal{I}_\omega^+}$ such that $(d', i-1, \widehat{e}) \in \eta$ and $(e, \widehat{e}) \in r^{\mathcal{I}_\omega^+}$. Add $(d', i-1, \widehat{e})$ to ρ. The required $(i-1)$-bounded simulation $\rho_{d',i-1,\widehat{e}}$ of $(\mathcal{I}_\omega^+, d')$ in $(\mathcal{I}_\omega^+, \widehat{e})$ is provided by η. □

Finally, we have reached our goal, the construction of the interpretation $\mathcal{I}$ described through this section and Lemmata 5.8 and 5.9 provide the arguments to show Proposition 5.4. We have thus solved **Problem 1**.

5.3 Constructing Locally Acyclic Finite Models

In this section, we focus on solving **Problem 2**, and thus reach our final goal, that is, to construct a finite model of a satisfiable $(\mathcal{T}_\mathsf{f}, \mathcal{A})$ such that every n_0-substructure of this model homomorphically embeds into $\mathcal{U}$. Note that the model $\mathcal{I}$ from the previous section (solving **Problem 1**) is still not as required since it may contain (non-ABox) cycles that are not present in $\mathcal{U}$, and thus the existence of **Problem 2**. In particular, we will see that such cycles cannot be completely avoided, but they can be made large enough so that they are not 'visible' in n_0-substructures.

In order to construct the desired finite model, we will use a technique introduced by Otto [105] to obtain locally acyclic *bisimilar covers* of *finite labelled transition systems*, which can also be seen as Kripke structures or relational structures with unary and binary predicates. A homomorphism $\rho : \widehat{\mathfrak{I}} \to \mathfrak{I}$ is *bisimilar cover* of $\mathfrak{I}$ by $\widehat{\mathfrak{I}}$, if $\{(\widehat{d}, d) \mid d = \rho(\widehat{d})\}$ is a bisimulation between $\widehat{\mathfrak{I}}$ and $\mathfrak{I}$. Very informally, a bisimulation, is a relation $\sigma : \Delta^{\widehat{\mathfrak{I}}} \times \Delta^{\mathfrak{I}}$ that satisfies "both directions" of conditions 1, 2, and 3 of simulations (see Definition 5.2).

It is well known that the *tree unravelling* of a (finite) relational structure provides a bisimilar cover. However, note that the unravelling is infinite if the original relational structure does have cycles. Otto [105] shows how to obtain a locally acyclic bisimilar cover of a finite relational structure $\mathfrak{I}$ from the product of $\mathfrak{I}$ with a *finite group* of *high girth*. A more detailed explanation will follow, but let us first introduce some basic notions of group theory.

Definition 5.6. A *group* is a set, G, together with an operation $\circ$ that combines any two elements g_1 and g_2 to form another element $g_1 \circ g_2$. $(G, \circ)$ must satisfy the following *group axioms*:

Closure. For all $g_1, g_2 \in G$, $g_1 \circ g_2 \in G$.

Associativity. For all $g_1, g_2, g_3 \in G$, $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$.

Identity element. There exists a unique element $h \in G$, such that for every element $g \in G$, the equation $h \circ g = g \circ h = g$ holds. h is called the *identity element of* G, and is denoted by 1_G.

Inverse element. For each $g \in G$, there exists exactly one element $g' \in G$ such that $g \circ g' = g' \circ g = 1_G$. g' is called the *inverse* of g and is usually denoted by g^{-1}.

A group g element is called *involutive* if $g \circ g = 1_G$, i.e., $g = g^{-1}$. A *finite group* is a group $(G, \circ)$ such that G is a finite set. A *generating set* of a group $(G, \circ)$ is a subset $H \subseteq G$ such that every element $g \in G$ can be expressed as some finite combination $h_1 \circ \cdots \circ h_n$ where each h_i is such that either $h_i \in H$ or $h_i = h^{-1}$ for some $h \in H$. For an arbitrary set $H \subseteq G$, we write $\langle H \rangle$ to denote the group generated by H (which might not coincide with $(G, \circ)$). A nonempty subset H of a group G is said to be *symmetric* if $h^{-1} \in H$ whenever $h \in H$.

△

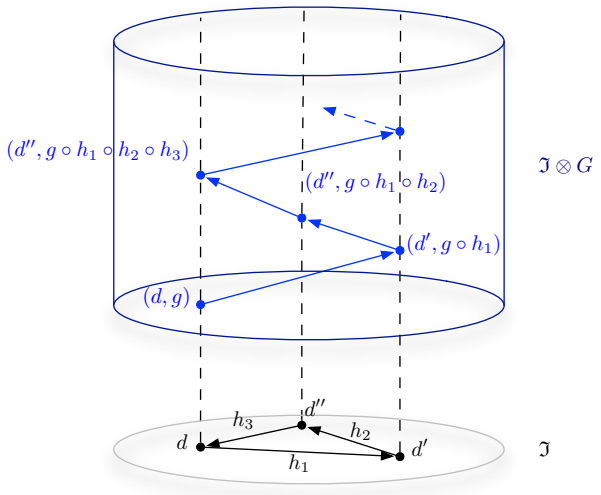


Figure 5.1: Product Construction

Notably, the structure of a group is made explicit by its so-called *Cayley Graph*.

Definition 5.7. Let $(G, \circ)$ be a finite group generated by a symmetric set H. The *Cayley graph of G w.r.t. H* is the undirected graph $\Gamma = (G, H)$ with set of vertices G and the edges defined as follows: $\{g, g'\}$ is an edge if $g' = g \circ h$, for some $h \in H$. The *girth* of a graph is the length of a shortest cycle contained in that graph. The *girth of a group G* is then the girth of its Cayley graph.

△

The construction conceived by Otto [105] uses a product $\mathfrak{I} \otimes G$ of a given structure $\mathfrak{I}$ with a finite group G generated by a symmetric set H that has an element g_e for every R-edge e in $\mathfrak{I}$, with R a binary predicate. Intuitively, in the product $\mathfrak{I} \otimes G$ there is then an R- edge from (d, g) to (d', g') if $(d, d') \in r^{\mathfrak{I}}$ and $g' = g \circ h_1$, where h_1 is the group element assigned to the R-edge (d, d'); e.g., see Figure 5.1 for a graphical reference. In this fashion, any cycle in the product projects to a cycle in the graph of G w.r.t. H, and hence its length is bounded from below by the girth of G. This means that, if the girth of G is sufficiently large one avoids the occurrence of "short cycles" in $\mathfrak{I} \otimes G$. Note also that the out-degree of every node in the required Cayley graph should be exactly k, with k the number of edges in $\mathfrak{I}$. That means that the graph (G, H) should be *k-regular*.

For any k and m, explicit constructions of k-regular graphs with girth greater than m have been studied in the literature. The following is a known result, c.f. [4] for a full discussion of the construction.

Theorem 5.10.
For every $k, m > 0$ there exists a finite group G which is generated by a set of k involutive generators, and whose Cayley graph has regular degree *k and girth at least m.*

From the observation that the relational structures considered by Otto correspond also to interpretations in DLs, the described product construction seems a good candidate to solve **Problem 2**. However, there are a number of technical challenges disallowing a direct application of this construction to our interpretations:

- **Challenge 1**. The presence of distinguished elements in our interpretations, that is, the ABox individuals. Indeed, by applying the previous construction to a model $\mathcal{I}$ of an ontology $(\mathcal{T}, \mathcal{A})$ such that there is cycle in $\mathcal{A}$, we would obtain an interpretation $\mathcal{I} \otimes G$ (think of $\mathcal{I}$ as a structure) that no longer satisfies $\mathcal{A}$.

- **Challenge 2**. The construction described above assumes that the relational structures are *simple*. Translated to our case an interpretation $\mathcal{I}$ is *simple* if for every pair of roles r, s, $r^{\mathcal{I}} \cap s^{\mathcal{I}} = \emptyset$; and there is no role r with $r(d, d) \in \mathcal{I}$. Both of these conditions cannot be guaranteed for any of our constructions of finite models described so far.

Before we proceed to solve **Challenge 1** and **2**, we formally define the product interpretation. Let $\mathcal{J}$ be an interpretation. We use $E_{\mathcal{J}}$ to denote the set of all edges of $\mathcal{J}$, that is, all sets $\{d, e\}$ such that $(d, e) \in r^{\mathcal{J}}$ for some role r. Let $(G, \circ)$ be a finite group generated by a set $H = \{g_S \mid S \in E_{\mathcal{J}}\}$: that is, the set of edges $E_{\mathcal{J}}$ can be embedded via an injection into H. The existence of such a group G is granted by Theorem 5.10. We use $\mathcal{J} \otimes G$ to denote the interpretation with domain $\Delta^{\mathcal{J}} \times G$ defined as follows:

$$A^{\mathcal{J} \otimes G} = \{\langle d, h\rangle \in \Delta^{\mathcal{J}} \times G \mid d \in A^{\mathcal{J}}\}$$
$$r^{\mathcal{J} \otimes G} = \{(\langle d, h\rangle, \langle d', h \circ g_{\{d,d'\}}\rangle) \mid (d, d') \in r^{\mathcal{J}}\}.$$

We are now ready to start solving **Challenge 1** and **2**. We proceed in the opposite order: we start solving **Challenge 2**. More precisely, we show that given an interpretation $\mathcal{I}$ we can transform it into a *simple* interpretation, i.e., an interpretation without self loops or elements related by more that one role outside the ABox.

5.3.1 Reduction to Simple Interpretations

As a preliminary, we first transform $\mathcal{I}$ to rule out cycles of length 1 (reflexive loops) or 2. We make more precise this notion in the following.

Definition 5.8. An interpretation $\mathcal{J}$ is called *simple* if it satisfies the following conditions for all objects $d, e \in \Delta^{\mathcal{J}}$ and (possibly inverse) roles r, s:

1. If $d \notin \mathsf{Ind}(\mathcal{A})$, then $(d, d) \notin r^{\mathcal{J}}$.

2. If $(d, e) \notin \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A})$ and $(d, e) \in r^{\mathcal{J}} \cap s^{\mathcal{J}}$, then $r = s$.

△

The following construction shows how to transform $\mathcal{I}$ into a simple interpretation $\mathcal{I}'$. Let $r_1, \dots, r_R$ be the role names occurring in $\mathcal{A}$ and $\mathcal{T}_\mathsf{f}$. We take $2R+2$ disjoint copies of $\Delta^\mathcal{I}$, and interpret ABox elements in the last copy and concept names in every copy the same way as in $\Delta^\mathcal{I}$. In the model to be constructed, r_k-edges between non-ABox elements jump over k copies (modulo $2R+2$), and r_k-edges between ABox elements remain in the last copy if they originate there, or otherwise jump over k copies (modulo $2R+1$ this time). This way, we leave the ABox structure intact in the last copy, and break up all other cycles of length 1 and 2. More precisely,

$$
\begin{aligned}
\Delta^{\mathcal{I}'} &= \Delta^\mathcal{I} \times \{0, \dots, 2R+1\} \\
A^{\mathcal{I}'} &= \{\langle d, i\rangle \mid d \in A^\mathcal{I},\ 0 \le i \le 2R+1\} \\
r_k^{\mathcal{I}'} &= \{(\langle d, i\rangle, \langle e, i \oplus_{2R+2} k\rangle) \mid (d,e) \in r_k^\mathcal{I} \setminus \mathsf{Ind}(\mathcal{A})^2\} \\
&\quad \cup \{(\langle d, i\rangle, \langle e, i \oplus_{2R+1} k\rangle) \mid (d,e) \in r_k^\mathcal{I} \cap \mathsf{Ind}(\mathcal{A})^2, i \le 2R\} \\
&\quad \cup \{(\langle d, 2R+1\rangle, \langle e, 2R+1\rangle) \mid (d,e) \in r_k^\mathcal{I} \cap \mathsf{Ind}(\mathcal{A})^2\}
\end{aligned}
$$

ABox individuals are interpreted in the first copy, that is, we identify a with $\langle a, 2R+1\rangle$. Note that the last line preserves the structure of the ABox, and the preceding lines ensure simpleness (but do not generally rule out cycles of length 3).

We next show that $\mathcal{I}'$ is indeed simple:

1. Whenever $(\langle d, i\rangle, \langle d, i\rangle) \in r_k^{\mathcal{I}'}$, the construction ensures that $i = 2R+1$ and d is an ABox element.

2. Let $(\langle d, i\rangle, \langle e, j\rangle) \in (r^{\mathcal{I}'} \cap s^{\mathcal{I}'}) \setminus \mathsf{Ind}(\mathcal{A})^2$. We distinguish three cases.

   - **case1:** *Both r, s are role names: $r = r_k$, $s = r_\ell$.* Then the above pair has been added in the first or second line of the constructions of both $r_k^{\mathcal{I}'}$ and $r_\ell^{\mathcal{I}'}$. If it was added in the second line, then we have $j = i \oplus_{2R+1} k = i \oplus_{2R+1} \ell$ which, due to $0 < k, \ell \le R$, implies $k = \ell$ and hence $r_k = r_\ell$. The case for the first line is analogous.
   - **case 2:** *One of r, s is a role name; the other is not: $r = r_k$, $s = r_\ell^-$.* As in the previous case, the above pair must have been added in the first or second line of the constructions of both role interpretations. If it was added in the second line, then we have $j = i \oplus_{2R+1} k$ and $i = j \oplus_{2R+1} \ell$. Inserting the first equation into the second, we get $i = i \oplus_{2R+1} k \oplus_{2R+1} \ell$, which is impossible because $0 < k + \ell \le 2R$. The case for the first line is analogous.
   - **case 3:** *None of r, s are role names: $r = r_k^-$, $s = r_\ell^-$.* This case reduces to the first case if we swap $\langle d, i\rangle$ and $\langle e, j\rangle$.

It is an easy exercise to show that $\mathcal{I}'$ is a model of $\mathcal{A}$ and $\mathcal{T}_\mathsf{f}$, and that $\mathcal{I}' \preceq \mathcal{I}$. From Lemma 5.9, we thus get $\mathcal{I}' \preceq_{n_0} \mathcal{U}$.

In what follows we concentrate on the solution of **Challenge 1**, and thus finally solving **Problem 2**.

5.3.2 Products of Simple Interpretations.

Now, consider the finite model $\mathcal{I}$ of $(\mathcal{T}_\mathsf{f}, \mathcal{A})$ constructed in the Section 5.2, that is, the model constructed to solve **Problem 1**. By the reduction above, we may assume that $\mathcal{I}$ is simple. Moreover, by Theorem 5.10, we can take a finite group G generated by an involutive set $H = \{g_S \mid S \in E_\mathcal{I}\}$ of cardinality $k = |E_\mathcal{I}|$, such that the girth of G is greater than n_0 and its Cayley graph w.r.t. H is k-regular.

To obtain an interpretation that satisfies the ABox (and then solving **Challenge 1**), we consider the interpretation $\widehat{\mathcal{J}}$ that can be obtained as follows:

- start with $\mathcal{I}^- \otimes G$, where $\mathcal{I}^-$ is obtained from $\mathcal{I}$ by removing, for each $r(a,b) \in \mathcal{A}$, the pair (a,b) from $r^\mathcal{I}$;
- then take an arbitrary but fixed $h_\mathcal{A} \in G$, for every $a \in \mathsf{Ind}(\mathcal{A})$ and identify each ABox element a with $(a, h_\mathcal{A})$;
- finally, for each $r(a,b) \in \mathcal{A}$, add $(\langle a,h\rangle, \langle b,h\rangle)$ to $r^{\widehat{\mathcal{J}}}$, for every pair $\langle a,h\rangle, \langle b,h\rangle \in \Delta^{\widehat{\mathcal{J}}}$.

Note that all copies of the ABox in $\widehat{\mathcal{J}}$, not just the 'main' one identified by $h_\mathcal{A}$, inherit the relational structure of the ABox. We first observe that $\widehat{\mathcal{J}}$ is still a (finite!) model of $\mathcal{T}_\mathsf{f}$ and $\mathcal{A}$. This essentially follows from the observations by Otto [105].

Lemma 5.11. *$\widehat{\mathcal{J}}$ is a model of $\mathcal{T}_\mathsf{f}$ and $\mathcal{A}$.*

Proof. To show that $\widehat{\mathcal{J}}$ is a model of $\mathcal{T}_\mathsf{f}$, we use the fact that $\mathcal{I}$ is a model of $\mathcal{T}_\mathsf{f}$ and the construction of $\widehat{\mathcal{J}}$. CIs of the form $K \sqsubseteq A$, $K \sqsubseteq \bot$, $K \sqsubseteq \exists r.K'$, and $K \sqsubseteq \forall r.K'$ are easy to deal with. We thus concentrate on CIs $K \sqsubseteq (\leqslant 1\ r\ K')$. Assume that $\langle d,h\rangle \in K^{\widehat{\mathcal{J}}}$. By construction of $\widehat{\mathcal{J}}$, this means $d \in K^\mathcal{I}$. Assume to the contrary of what is to be shown that there are $(\langle d,h\rangle, \langle e_i,h_i\rangle) \in r^{\widehat{\mathcal{J}}}$ for $i = 1,2$ such that $\langle e_i,h_i\rangle \in K'^{\widehat{\mathcal{J}}}$ and $\langle e_1,h_1\rangle \neq \langle e_2,h_2\rangle$. Then $(d,e_i) \in r^\mathcal{I}$ and since $\mathcal{I}$ is a model of $\mathcal{T}_\mathsf{f}$, we obtain $e_1 = e_2 =: e$. Now the construction of $\widehat{\mathcal{J}}$ yields that, if both d, e interpret ABox elements in $\mathcal{I}$, then $h_1 = h_2 = h$, and that otherwise $h_i = h \circ g_{\{d,e\}}$ for both $i = 1,2$. Finally, using the construction of $\widehat{\mathcal{J}}$, it is easy to observe that $\widehat{\mathcal{J}}$ is a model of $\mathcal{A}$. □

After solving **Challenge 1** and **2**, we can show that indeed $\widehat{\mathcal{J}}$ does not contain 'small' cycles, and hence solving **Problem 2**. Before we proceed with the proof, we need some auxiliary notions. A *cycle in* $\widehat{\mathcal{J}}$ (of length n) is a path $p_1, r_1, \ldots, r_n, p_{n+1}$, where $n > 2$, $p_i \in \Delta^{\widehat{\mathcal{J}}}$, each r_i is a (possibly inverse) role such that $(p_i, p_{i+1}) \in r_i^{\widehat{\mathcal{J}}}$, and $p_1 = p_{n+1}$. Further, a cycle is *simple* if, for $1 \leq i < j \leq n$, we have $p_i \neq p_j$. An element $p = \langle d,h\rangle \in \Delta^{\widehat{\mathcal{J}}}$ is an *ABox element* if $d \in \mathsf{Ind}(\mathcal{A})$. Note that this definition includes all "copies" of ABox elements from $\mathcal{I}$, not just those that interpret ABox individuals. We say that $\widehat{\mathcal{J}}$ is *n-acyclic relative to $\mathcal{A}$* if every simple cycle in $\widehat{\mathcal{J}}$ of length at most n contains exclusively ABox individuals.

Lemma 5.12. *$\widehat{\mathcal{J}}$ is n_0-acyclic relative to $\mathcal{A}$.*

Proof. We start with the following observation:

Observation: If $(p_1, p_2) \in r^{\widehat{\mathcal{J}}}$ with $p_i = \langle d_i, h_i \rangle$, $i \in \{1, 2\}$, and at least one of d, e is not an ABox element, then $h_2 = h_1 \circ g_{\{d_1,d_2\}}$.

In fact, this is immediate if r is a role name. If $r = s^-$, then $h_1 = h_2 \circ g_{\{d_1,d_2\}}$, which by multiplication with $g_{\{d_1,d_2\}}$ and due to the generators being involutive yields $h_1 \circ g_{\{d_1,d_2\}} = h_2$.

Let $\alpha = p_1, r_1, \ldots, r_n, p_{n+1}$ be a simple cycle in $\widehat{\mathcal{J}}$, with $p_i = \langle d_i, h_i \rangle$ for $1 \leq i \leq n$, such that for some i, p_i is not an ABox element. Assume to the contrary of what is to be shown that $n \leq n_0$. We show that h_{i-1}, h_i, and h_{i+1} are all different. Consequently, α gives rise to a cycle of length between three and n_0 in the Cayley graph of G (even if some of the other elements on α should coincide), which contradicts the non-existence of such cycles. We have $h_{i-1} \neq h_i$ since otherwise $h_{i-1} = h_{i-1} \circ g_{i-1}$, which is not possible due to n_0-regularity of the Cayley graph of G; for the same reason, $h_i \neq h_{i+1}$. Finally, assume to the contrary of what we want to show that $h_{i-1} = h_{i+1}$. Then $h_{i-1} \circ g_{i-1} \circ g_i = h_{i-1}$. Combining with g_i yields $h_{i-1} \circ g_{i-1} = h_{i-1} \circ g_i$, which gives $g_{i-1} = g_i$ by k-regularity of G. Since $g_{i-1} = g_{\{d_{i-1},d_i\}}$ and $g_i = g_{\{d_i,d_{i+1}\}}$, this yields $d_{i-1} = d_{i+1}$, thus $p_{i-1} = p_i$ in contrast to α being simple. □

Finally, we show that indeed the constructed $\widehat{\mathcal{J}}$ avoids the **Problems 1** and **2**, and therefore it can be used as the $\mathcal{J}'$ of Proposition 5.3. Recall that with Proposition 5.3 we can straightforwardly proof the " $\Rightarrow$: " of our main result: Theorem 5.1.

Lemma 5.13. *Every n_0-substructure $\mathcal{J}'_{n_0}$ of $\widehat{\mathcal{J}}$ homomorphically embeds into the canonical model $\mathcal{U}$ of $(\mathcal{T}_\mathsf{f}, \mathcal{A})$.*

Proof. We may assume w.l.o.g. that $\mathcal{J}'_{n_0}$ is connected. We start with making a useful observation.

Claim 4. For each $p_1 \in \Delta^{\widehat{\mathcal{J}}}$ that is not an ABox element, there is at most one simple path $p_1 r_1 p_2 \cdots p_k r_k p_{k+1}$ in $\mathcal{J}'_{n_0}$ such that $p_1, \ldots, p_k$ are not ABox elements and p_{k+1} is an ABox element.

Assume there is a $p_1 \in \Delta^{\widehat{\mathcal{J}}}$ that is not an ABox element and such that there are two simple paths in $\mathcal{J}'_{n_0}$ of the described form. Each such path $p_1 r_1 p_2 \cdots p_k r_k p_{k+1}$ gives rise to a corresponding path $d_1 r_1 d_2 \cdots d_k r_k d_{k+1}$ in $\mathcal{I}$ such that $d_1, \ldots, d_k$ are not ABox individuals, but d_{k+1} is. Note that the initial version of the modified finite interpretation $\mathcal{I}$ contains $\mathcal{U}_0$, which takes the form of the ABox $\mathcal{A}$ extended with a tree of depth n_0 below each ABox individual, and that later steps in the construction of $\mathcal{I}$ only add successors to leaves in these trees. Therefore, and since the length of all mentioned paths is clearly bounded by n_0, both paths in $\mathcal{I}$ must be inside the same tree of $\mathcal{U}_0$. But then they must be identical since they start and end at the same element and are simple.

Choose an arbitrary $p_0 = (d_0, h_0) \in \Delta^{\mathcal{J}'_{n_0}}$. By Lemma 5.9, we know that $(\mathcal{I}, d_0) \preceq_{n_0} \mathcal{U}$, witnessed by an n_0-bounded simulation ρ. We use ρ to construct the desired homomorphism η from $\mathcal{J}'_{n_0}$ to $\mathcal{U}$.

In what follows, let π be the projection on the first component of elements in $\Delta^{\widehat{\mathcal{J}}}$. We define a sequence of partial homomorphisms η_i, $i \geq 0$, that is, partial functions $\eta_i : \Delta^{\mathcal{J}'_{n_0}} \times \Delta^{\mathcal{U}}$ that satisfy Conditions 1 to 3 of homomorphisms. The desired homomorphism η is then obtained in the limit. We will make sure that all η_i satisfy the following properties:

(a) $(\pi(p), n_0 - i, \eta_i(p)) \in \rho$ for all $p \in \Delta^{\mathcal{J}'_{n_0}}$;

(b) if $\mathcal{J}'_{n_0}$ contains a path of length $\leq i$ from an initial element to p, then $\eta_i(p)$ is defined, where an element q is *initial* if $\eta_0(q)$ is defined.

We start by defining η_0 as follows.

- If $\Delta^{\mathcal{J}'_{n_0}}$ contains ABox elements, then set $\eta_0(p) = a$ for all $p = \langle a, h\rangle \in \Delta^{\mathcal{J}'_{n_0}}$ with $a \in \mathsf{Ind}(\mathcal{A})$.
- If $\Delta^{\mathcal{J}'_{n_0}}$ does not contain ABox elements, then choose an $e \in \Delta^{\mathcal{U}}$ with $(\pi(p_0), n_0, e) \in \rho$ and set $\eta_0(p_0) = e$.

Clearly, η_0 satisfies (a) and (b) and Condition 1 of homomorphisms. Satisfaction of Condition 2 follows from $(*)$. Finally, satisfaction of Condition 3 follows from the existence of ρ and the fact that, by definition of bounded simulations, ρ preserves all edges between ABox elements.

In the induction step, η_{i+1} is obtained from η_i by defining a value for all $p_2 \in \Delta^{\mathcal{J}'_{n_0}}$ such that there is some edge $(p_1, p_2) \in r^{\widehat{\mathcal{J}}}$ with $\eta_i(p_1)$ defined and $\eta_i(p_2)$ undefined. To define $\eta_{i+1}(p_2)$, we observe that $(\pi(p_1), \pi(p_2)) \in r^{\mathcal{I}}$ follows from $(p_1, p_2) \in r^{\widehat{\mathcal{J}}}$ and $(\pi(p_1), n_0 - i, \eta_i(p_1)) \in \rho$ holds by (a). Moreover, we have $i < n_0$ by (b) and since any two elements in $\mathcal{J}'_{n_0}$ reach each other by a path of length $\leq n_0$, thus $i = n_0$ contradicts $\eta_i(p_2)$ being undefined. Consequently there must be some e such that $(\pi(p_2), n_i - i - 1, e) \in \rho$ and $(\eta_i(p_1), e) \in r^{\mathcal{U}}$. Set $\eta_{i+1}(p_2) = e$.

We next show that η_{i+1} is well-defined, that is, if $(p_1, p) \in r^{\widehat{\mathcal{J}}}$ and $(p_2, p) \in s^{\widehat{\mathcal{J}}}$ with $\eta_i(p_1)$ and $\eta_i(p_2)$ defined and $\eta_i(p)$ undefined, then $(p_1, r) = (p_2, s)$. Assume to the contrary that this is not the case. We distinguish three cases:

- $p \in \{p_1, p_2\}$. Then $(p_1, p_1) \in r^{\widehat{\mathcal{J}}}$ or $(p_2, p) \in s^{\widehat{\mathcal{J}}}$. We address the fomer case, the latter is analogous. Let $p_1 = \langle d_1, h_1\rangle$. Then $(p_1, p_1) \in r^{\widehat{\mathcal{J}}}$ yields $(d, d) \in r^{\mathcal{I}}$. Since $\mathcal{I}$ is simple, $d \in \mathsf{Ind}(\mathcal{A})$, thus p is an an ABox element. This is a contradiction to $\eta_i(p)$ being undefined
- $p \notin \{p_1, p_2\}, p_1 = p_2$, and $r \neq s$. Let $p_1 = \langle d_1, h_1\rangle$ and $p = \langle d, h\rangle$. Then we have $(d_1, d) \in r^{\mathcal{I}}$ and $(d_1, d) \in s^{\mathcal{I}}$. Since $\mathcal{I}$ is simple and $d \notin \mathsf{Ind}(\mathcal{A})$ (because p cannot be an ABox element), this yields $r = s$ as required.
- $p \notin \{p_1, p_2\}$ and $p_1 \neq p_2$. Since $\eta_i(p_1)$ and $\eta_i(p_2)$ are defined and $\eta_{i+1}(p)$ is not, p_j is reachable from some initial element $\widehat{p}_j$ on a path $\mathcal{P}_j$ of length i and this is the shortest path from any initial element to p_j, for $j \in \{1, 2\}$. If $\Delta^{\mathcal{J}'_{n_0}}$ contains no ABox elements, then $\widehat{p}_1 = \widehat{p}_2 = p_0$. Otherwise, $\widehat{p}_1$ and $\widehat{p}_2$ are ABox elements and we obtain from Claim 1 and the fact that both $\widehat{p}_1$ and $\widehat{p}_2$ are reachable from p that $\widehat{p}_1 = \widehat{p}_2$. For readability, we from now on use $\widehat{p}$ to denote $\widehat{p}_1 (= \widehat{p}_2)$.

Let p' be the element on the path $\mathcal{P}_1$ that also occurs on the path $\mathcal{P}_2$ and is furthest away from $\widehat{p}$ (such an element always exists since $\widehat{p}$ is on both paths). Consider the following cycle in $\widehat{\mathcal{J}}$:

1. from p' to p_1 along $\mathcal{P}_1$;
2. from p_1 to p along r;
3. from p to p_2 along s^-;
4. from p_2 to p' backwards along $\mathcal{P}_2$.

Since $p \notin \{p_1, p_2\}$ and $p_1 \neq p_2$, this cycle has length > 2 as required. Moreover, the cycle is simple: by choice, the two travelled subpaths of $\mathcal{P}_1$ and $\mathcal{P}_2$ do not share any elements, including p_1 and p_2. Moreover, p does not occur on these subpaths because η_i must be defined for all elements on the subpaths whereas it is not defined for p. Since p occurs on a simple cycle, p must be an ABox element. This yields a contradiction to $\eta_i(p)$ being undefined.

To finish the proof, we note that it is clear that η_{i+1} satisfies (a), (b), and all three conditions of homomorphisms. □

5.4 Complexity Boundaries

Apart from enabling the use of algorithms for unrestricted PEQ answering to decide finite model PEQ entailment, the result shown in Theorem 5.1 yields tight complexity bounds for finite model PEQ entailment. Note that decidability of PEQ entailment in Horn-$\mathcal{ALCFI}$ was expected given a result by Pratt-Hartmann [112]. That result states that finite CQ answering for the two-variable guarded fragment of first-order logic extended with counting quantifiers $\mathcal{GC}^2$ – which is a proper superset of $\mathcal{ALCQI}$– is decidable. We assume that his proof can be extended to unions of conjunctive queries (UCQs), and thus to PEQs. The complexity results obtained by Pratt-Hartmann concern only data complexity of finite CQ entailment in $\mathcal{GC}^2$, which he finds to be coNP-complete.

Theorem 5.14. *Finite PEQ entailment in Horn-$\mathcal{ALCFI}$ is decidable,* ExpTime*-complete in combined complexity, and* PTime*-complete in data complexity.*

Proof. For the unrestricted case, an ExpTime lower bound is in Baader et al. [15] and a PTime one in [39]. Both results can easily be adapted to the finite case. The upper bounds can be proved using the following straightforward algorithm for PEQ entailment, which resembles existing algorithms such as those presented in [28, 51, 88, 104]. Assume that an input ABox $\mathcal{A}$, TBox $\mathcal{T}$, and PEQ q are given, and let n_0 be the number of variables in q. As a consequence of Theorem 3.24, finite satisfiability w.r.t. $\mathcal{T}$ coincides with unrestricted satisfiability w.r.t. $\mathcal{T}_\mathsf{f}$. Using our algorithm for computing finite satisfiability in Horn-$\mathcal{ALCFI}$ in ExpTime, we can thus compute the set $\mathsf{TP}(\mathcal{T}_\mathsf{f})$ of types for $\mathcal{T}_\mathsf{f}$ without computing $\mathcal{T}_\mathsf{f}$ or explicitly reasoning w.r.t.

this exponentially large TBox. Let $\mathcal{A}'$ be the extension of $\mathcal{A}$ with assertions $\{A(a_t) \mid A \in t\}$ for each $t \in \mathsf{TP}(\mathcal{A})$. Now compute an initial piece $\mathcal{U}'$ of the canonical model $\mathcal{U}$ of $\mathcal{A}'$ and $\mathcal{T}_\mathsf{f}$, namely its restriction to depth n_0. Similar to the computation of $\mathsf{TP}(\mathcal{T}_\mathsf{f})$ above, we can do this by using finite subsumption w.r.t. $\mathcal{T}$ instead of unrestricted subsumption w.r.t. $\mathcal{T}_\mathsf{f}$. It is not difficult to prove that $\mathcal{U}' \models q$ iff $\mathcal{U} \models q$. To check whether $\mathcal{U}' \models q$ within the desired time bounds, we can simply enumerate all possible maps of variables in q to elements of $\mathcal{U}'$ and check whether any such map is a match. □

Theorem 5.14 states that PEQ entailment in Horn-$\mathcal{ALCFI}$ has the same complexity in finite and in unrestricted models. For the unrestricted case, PTime-completeness in data complexity follows from the results in [72], and ExpTime-completeness in combined complexity is proved in [51] for UCQs. We assume that the techniques in that paper extend to PEQs.

Chapter

6 Queries with Negation and Inequality over Horn Ontologies

In classical database theory, conjunctive queries (CQs) have played a key role due to their desirable theoretical properties. In the light of this precedent, a vast amount of research on answering CQs in the context of OBDA has been conducted in the last decade, so that now we have a fairly clear landscape of the computational complexity of answering CQs over Horn and expressive DL languages.

Conjunctive queries belong to the positive existential fragment of first-order logic, and therefore lack means to express '*complementation*' or '*difference*'. Unfortunately, some natural queries require to express either of these; for example, to retrieve *'all members of the staff that do* `not` *belong to a union'* or *'all students whose month of birth is* `(different from) not` *September'*. A well-known fact from database theory is that answering CQs with negation is harder than answering CQs. Rosati [116] showed that in the case of ontological query answering in *DL-Lite* the increase in the complexity is dramatic: in striking contrast to the highly tractable AC^0 bound for data complexity in the case of *unions* of CQs, the problems of answering unions of CQs with *inequalities* ($CQs^{\neq}$) and unions of CQs with *safe negation* ($CQs^{\neg s}$) were shown to be undecidable over the simplest language of *DL-Lite*. Further, for $\mathcal{EL}$ the situation is similar for safe negation: answering $UCQs^{\neg s}$ is undecidable. Remarkably, Klenke [82] showed that answering *single* $CQs^{\neq}$ is also undecidable.

Extending CQs and UCQs with negated atoms has an effect not witnessed before in ontological query answering over Horn ontologies. In particular, there might be a difference between considering *union* of CQs or a *single* CQ: by the fact that answering $UCQs^{\neg s}$ and $UCQs^{\neq}$ is undecidable we cannot straightforwardly conclude that this is the case for $CQs^{\neg s}$ and $CQs^{\neq}$. A reason to draw that conclusion is that the proofs used so far to show the results reported above rely on reductions of undecidable problems (e.g., $\mathbb{N}\times\mathbb{N}$-tiling problem) to answering $UCQs^{\neq}$ ($UCQs^{\neg s}$) over Horn DLs where each disjunct of the $UCQ^{\neq}$ takes care of properly encoding a 'restriction' present in the reduced problem (e.g., coloring condition, matching condition, etc), and how to obtain a similar behavior with a single query is far from obvious.

The addition of negated atoms to CQs brings not only an increase in the computational complexity, but it also introduces further technical difficulties for the development of algorithmic approaches since negated atoms are not preserved under homomorphisms [49]. As a consequence, to potentially devise algorithms for answering CQs$^{\neq}$ and CQs$^{\neg s}$ over Horn DLs we cannot directly use techniques based on the construction of the canonical model of the ontology. Due to these difficulties, up to now, the only known results are coNP-hardness for answering CQs$^{\neq}$ and CQs$^{\neg s}$ over *DL-Lite*$^{\mathcal{H}}_{core}$ [116], and undecidability of answering CQs$^{\neq}$ in $\mathcal{EL}$ [82].

In this Chapter, we aim to sharpen the panorama of answering CQs with safe negation and inequalities over Horn ontologies. We investigate different syntactic restrictions to CQs$^{\neg s}$ or CQs$^{\neq}$ proposed in the literature. A notorious way of attaining decidability for undecidable logics is to allow only *guarded* quantifications; see e.g., [5, 18, 61]. In particular, it has been recently shown that open world answering of first-order queries with *guarded negation* is decidable [19]. Taking into account the latter, we investigate the impact of this restriction on answering CQs with negated atoms over Horn DLs. Another prominent syntactic restriction is to constraint the *number* of negated atoms allowed to occur in a query [6, 19, 54, 83]. Throughout this chapter, we study the influence of this parameter on the complexity of answering CQs with negated atoms over Horn DLs.

6.1 Answering CQs with Safe and Guarded Negation

In this section, we analyze answering queries with safe negations. It is known [116] that, answering *union* of CQs with safe negation (UCQ$^{\neg s}$) over *DL-Lite*$_{core}$ and $\mathcal{EL}$ is undecidable. These results regard both the ontology (TBox and ABox) and the query as part of the problem input. We begin this section (Theorem 6.2) by a transparent reduction of the undecidable halting problem to answering a *single fixed* Boolean CQ$^{\neg s}$ over $\mathcal{ELI}_{\bot}$ (that is $\mathcal{ELI}$ extended with $\bot$) ontologies with a *fixed* TBox. Then (in Lemma 6.3) we establish a close correspondence between $\mathcal{ELI}$ TBoxes and *unions* of CQ$^{\neg s}$s over *DL-Lite*$_{core}$ TBoxes, which automatically implies undecidability of answering UCQ$^{\neg s}$ over *DL-Lite*$_{core}$ when the TBox and query are fixed (Corollary 6.4).

We then proceed to show (in Lemma 6.5) that the union of tree-shaped CQ$^{\neg s}$s from the proof of Corollary 6.4 can be replaced by a *single* CQ$^{\neg s}$ and a number of role inclusions. Thus, we extend the undecidability result to the problem of answering CQs with safe negation in *DL-Lite*$^{\mathcal{H}}_{core}$. We point out that the transformation of Lemma 6.5 is general (in particular, it is applicable to plain CQs and CQs with inequalities) and may be of wider interest.

In Theorem 6.7, we explore the limits of undecidability and prove that answering unions of *three* CQ$^{\neg s}$s over *DL-Lite*$_{core}$, which does not contain role inclusions, is undecidable. The problem for unions with one or two disjuncts remains open.

Finally, we turn to CQs with guarded negation (GNCQs), which are known [19] to be decidable over Horn DLs and establish matching lower bounds for data complexity: coNP for GNCQs and P-complete for GNCQs with a single negated atom.

To ease the presentation and the proofs in what follows, we will consider a restricted form of $\mathcal{ELI}$ TBoxes. A *primitive $\mathcal{ELI}$ TBox* is a finite set of axioms of the form:

$$B \sqsubseteq C,$$

where C is an $\mathcal{ELI}$ concept, and B is either a concept name or a concept of the form $\exists r$, i.e., a basic *DL-Lite* concept. Notably, primitive $\mathcal{ELI}$ TBoxes have the same expressive power as *DL-Lite*$^{\mathcal{H}}_{core}$ TBoxes. This is justified by the following observation.

Proposition 6.1. *For every concept inclusion α of the form*

$$A \sqsubseteq C,$$

where A is a concept name and C an $\mathcal{ELI}$ concept, we can construct a DL-Lite$^{\mathcal{H}}_{core}$ TBox $\mathcal{T}$ which is a model conservative extension of α: every model of $\mathcal{T}$ is a model of α and, conversely, every model of α can be extended to a model of $\mathcal{T}$ by giving an interpretation to the fresh symbols of $\mathcal{T}$.

Proof. Indeed, a concept inclusion of the form $B \sqsubseteq C_1 \sqcap C_2$ is equivalent to two concept inclusions $B \sqsubseteq C_i$, for $i = 1, 2$; and a concept inclusion of the form $B \sqsubseteq \exists r.C$ can be replaced by two concept inclusions $B \sqsubseteq \exists r_C$, $\exists r_C^- \sqsubseteq C$ and a role inclusion $r_C \sqsubseteq r$; for more details see, e.g., [9]. ❑

However, such a shortcut is not available in *DL-Lite*$_{core}$ because it contains no role inclusions.

For simplicity, we will consider Boolean (U)CQs with safe negation, and the problem of query entailment w.r.t. an ontology of such queries. Recall that deciding whether a satisfiable ontology $\mathcal{O}$ entails a query q is equivalent to decide whether $\mathcal{O} \wedge \neg q$ is satisfiable, i.e., whether there is a counter model of q. With that equivalence in mind, in the following, we will often prefer to represent Boolean (U)CQs with safe negation in their *negated form*. Given a Boolean query

$$q = \exists \vec{y} \varphi(y) \wedge \psi(\vec{y})$$

where $\psi(\vec{y})$ is the conjunction of all the negated atoms in q, the *negated form of q* is the sentence:

$$\forall \vec{y}.\varphi(\vec{y}) \rightarrow \psi(\vec{y}),$$

which is logically equivalent to $\neg q$. To simplify notation, we will usually omit the universal quantification.

6.1.1 Safe Negation: Undecidability over $\mathcal{ELI}_\bot$

Our proofs of the undecidability results in this section are by reduction of the halting problem for deterministic Turing machines. We note at this point that the choice of our reduction differs from the usual reduction of the unbounded tiling problem (see e.g., [116]). Indeed, the latter reduction

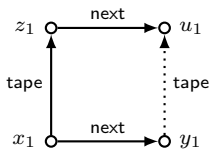


Figure 6.1: Completing the square with the Boolean CQ$^{\neg s}$ (6.1).

is usually easier to describe, however it requires the use of disjunction (or a UCQ), which is not available in our setting. A key observation to achieve our reduction is that a configuration of a *Turing machine* (that is, the contents of the tape and the current state with the position of the head at a particular step of the computation) can be written down on a sequence of domain elements with a role, tape, pointing to the representation of the next cell of the tape. Then a computation of the Turing machine can be thought of as a two-dimensional grid, where another role, next, points to the representation of each cell in the successive configuration.

In order to establish the required two-dimensional grid, we are going to use the following Boolean CQ$^{\neg s}$ $\boldsymbol{q}_1$:

$$\exists x_1, y_1, z_1, u_1 \left(\mathsf{next}(x_1, y_1) \land \mathsf{tape}(x_1, z_1) \land \mathsf{next}(z_1, u_1) \land \neg \mathsf{tape}(y_1, u_1)\right). \tag{6.1}$$

It can be readily seen that in any counter model $\mathcal{I}$ of $\boldsymbol{q}_1$, for every four elements forming the three sides of a square, there is a T-edge that completes the square, as shown in Fig. 6.1. Or more precisely, $\mathcal{I}$ satisfies the negated form of $\boldsymbol{q}_1$:

$$\mathsf{next}(x_1, y_1) \land \mathsf{tape}(x_1, z_1) \land \mathsf{next}(z_1, u_1) \to \mathsf{tape}(y_1, u_1) \tag{6.2}$$

Once the grid has been established, we can use $\mathcal{ELI}_\bot$ axioms to ensure that the elements of the grid encode successive configurations of a given deterministic Turing machine. This observation leads us to our first undecidability result.

Theorem 6.2. *There are a Boolean CQ$^{\neg s}$ and an $\mathcal{ELI}_\bot$ TBox such that query answering is undecidable.*

Proof. The proof is by reduction of the halting problem for deterministic Turing machines (see e.g., [107]). In particular, given a Turing machine M, we construct a TBox $\mathcal{T}$ and a query $\boldsymbol{q}$ such that M does not accept an input $\vec{w}$ encoded as an ABox $\mathcal{A}_{\vec{w}}$ iff $(\mathcal{T}, \mathcal{A}_{\vec{w}}) \not\models \boldsymbol{q}$ ($\mathcal{T}$ and $\boldsymbol{q}$ depend on M but not on $\vec{w}$). Applying this construction to a *fixed* deterministic *universal* Turing machine, i.e., a machine that accepts its input $\vec{w}$ iff the Turing machine encoded by $\vec{w}$ accepts the empty input, we obtain the required undecidability result.

Let $M = (\Gamma, Q, q_0, q_1, \delta)$ be a deterministic Turing machine, where Γ is an alphabet (containing the blank symbol $\sqcup$), Q a set of states, $q_0 \in Q$ and $q_1 \in Q$ are an initial and an accepting state, respectively, and $\delta \colon Q \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$ is a transition function. Computations of M can be thought of as sequences of configurations, with each configuration determined by the

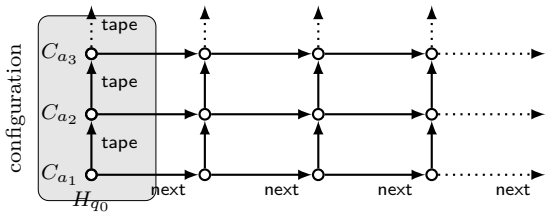


Figure 6.2: Encoding computations of a Turing machine as a grid.

contents of all (infinitely many) cells of the tape, the state and the head position. We encode a computation by domain elements arranged, roughly speaking, into a two-dimensional grid: one dimension is the tape and the other is time. See Fig. 6.2, where the nodes are domain elements and the grey rectangle illustrates an initial configuration, in which the tape contains the input $a_1a_2a_3\ldots$ and the head is positioned over the first cell in state q_0.

More precisely, we use a role tape to point to the representation of the next cell on the tape (within the same configuration) and a role next to point to the representation of the same cell in a successive configuration. Concepts C_a, for $a \in \Gamma$, encode the contents of cells in the sense that a domain element belongs to the interpretation of C_a if the respective cell contains symbol a. We use concepts H_q, for $q \in Q$, to indicate both the position of the head and the current state: a domain element belongs to the interpretation of H_q if the respective cell is under the head and the machine is in state q. We also use a concept E to mark all other cells on the tape (that is, cells that are not under the head of the machine) and concepts D^q_σ and D_σ, for $q \in Q$ and $\sigma \in \{-1, +1\}$, to propagate the head and no-head markers backwards and forwards along the tape, respectively. Finally, concept I is required to ensure that the tape is initially blank beyond the input word.

Let $\boldsymbol{q}$ be a Boolean $\mathrm{CQ}^{\neg s}$ with the following negated form:

$$\mathsf{next}(x, y) \land \mathsf{tape}(x, z) \land \mathsf{next}(z, u) \to \mathsf{tape}(y, u). \tag{6.3}$$

Let $\mathcal{T}_M$ be an $\mathcal{ELI}_\bot$ TBox containing the following concept inclusions:

$$H_q \sqcap C_a \sqsubseteq \exists \mathsf{next}.(C_{a'} \sqcap D^{q'}_\sigma), \qquad \text{for } \delta(q, a) = (q', a', \sigma), \tag{6.4}$$

$$E \sqcap C_a \sqsubseteq \exists \mathsf{next}.C_a, \qquad \text{for } a \in \Gamma, \tag{6.5}$$

$$H_q \sqsubseteq D_{-1} \sqcap D_{+1}, \qquad \text{for } q \in Q, \tag{6.6}$$

$$\exists \mathsf{tape}.D^q_{-1} \sqsubseteq H_q, \qquad \text{for } q \in Q, \tag{6.7}$$

$$\exists \mathsf{tape}.D_{-1} \sqsubseteq E \sqcap D_{-1}, \tag{6.8}$$

$$\exists \mathsf{tape}^{-}.D^q_{+1} \sqsubseteq H_q, \qquad \text{for } q \in Q, \tag{6.9}$$

$$\exists \mathsf{tape}^{-}.D_{+1} \sqsubseteq E \sqcap D_{+1}, \tag{6.10}$$

$$I \sqsubseteq \exists \mathsf{tape}.(I \sqcap C_{\sqcup}), \tag{6.11}$$

$$H_{q_1} \sqsubseteq \bot. \tag{6.12}$$

Let us explain briefly the intuition behind the CIs above.

- CIs described in (6.4) encode each transition $\delta(q, a) = (q', a', \sigma)$: if the symbol under the head is a and the current state is q ($H_q \sqcap C_a$), then the symbol under the head is changed to a' (using $C_{a'}$) and the new head position and state are recorded in the next step (i.e., in its next-succesor) using $D^{q'}_{\sigma}$, that information is then propagated by the CIs described in (6.7) and (6.9).
- On the other hand, CIs described in (6.5) are used to process the symbols in cells not under the head (i.e., those marked with the concept E) each such symbol is copied to the next configuration (using its next-successor).
- CIs described in (6.6), (6.10), (6.8) mark all the cells that are not under the head (using concept E). Recall that the cell under the head and the current state are marked using H_q, then by (6.6) such cell is also marked with D_{+1} (and D_{-1}), then all the cells after (and before) the one under the head will be marked with E by (6.10) (and by (6.8)).
- The CI in (6.11) (together with the ABox below $\mathcal{A}_{\vec{w}}$) is used for encoding that the tape initially contains blanks ($C_{\sqcup}$) in all the cells beyond the input.
- Finally, the CI in (6.12) serves to encode that the accepting state is never reached.

For every input $\vec{w} = a_1 \dots a_n \in \Gamma^*$,we take the following ABox $\mathcal{A}_{\vec{w}}$ with individual names $c_1, \dots, c_n$:

$$H_{q_0}(c_1), \quad C_{a_i}(c_i) \text{ and } \mathsf{tape}(c_i, c_{i+1}), \text{ for } 1 \leq i < n, \quad I(c_n).$$

It remains to show the following: $(\mathcal{T}_M, \mathcal{A}_{\vec{w}}) \not\models \boldsymbol{q}$ iff M does not accept $\vec{w}$.

($\Rightarrow$) Consider a model $\mathcal{I}$ of $(\mathcal{T}_M, \mathcal{A}_{\vec{w}})$ with $\mathcal{I} \not\models \boldsymbol{q}$. Then, by the definition of the ABox and (6.11), there exists an infinite sequence of (not necessarily distinct) domain elements $d_1, d_2, \dots$ that encode the initial configuration in a sense that $(d_i, d_{i+1}) \in \mathsf{tape}^{\mathcal{I}}$ for all $i \geq 1$, $d_1 \in H^{\mathcal{I}}_{q_0}$, $d_i \in H^{\mathcal{I}}_{\emptyset}$ for all $i > 1$, $d_i \in C^{\mathcal{I}}_{a_i}$, for each $1 \leq i \leq n$, and $d_i \in C^{\mathcal{I}}_{\sqcup}$ for all $i > n$. By (6.4) and (6.5), there exist elements $d'_1, d'_2, \dots$ such that $(d_i, d'_i) \in \mathsf{next}^{\mathcal{I}}$.

By (6.3), they form another tape-connected sequence, that is, $(d'_i, d'_{i+1}) \in \mathsf{tape}^{\mathcal{I}}$ for all i, which represents the second configuration of M. Indeed, by (6.4), the symbol in the cell under the head is changed according to the transition function δ of M, and the new head position and state are recorded in the concept $D^{q'}_{\sigma}$. By (6.7) and (6.9), the recorded head position and the state are passed onto the correct cell. Then, by (6.6), the domain element representing the head, say, d'_k, belongs to $D^{\mathcal{I}}_{+1}$, whence, by (6.10), all d'_i with $i > k$ belong to $D^{\mathcal{I}}_{+1}$ and $E^{\mathcal{I}}$. Similarly, by (6.6) and (6.8), $d'_i \in E^{\mathcal{I}}$, for all $i < k$. Therefore, all cells but the one under the head belong to $E^{\mathcal{I}}$, whence, by (6.5), the symbols they contain are preserved by the transition. By the same reasoning, there exists a respective sequence of elements for each configuration of the computation of M. Finally, (6.12) guarantees that the accepting state never occurs in the computation, i.e., M does not accept $\vec{w}$.

($\Leftarrow$) If M has a non-accepting computation on $\vec{w}$ then we can encode it by an infinite two-dimensional grid interpretation satisfying $(\mathcal{T}_M, \mathcal{A}_{\vec{w}})$ but not $\boldsymbol{q}$.

Since the problem of deciding whether a given deterministic machine accepts a given input is undecidable, we obtain the claim of the Theorem ❑

The use of $\bot$ seems to be crucial to achieve the result in 6.2. Indeed, CQ$^{\neg s}$ over $\mathcal{ELI}$ ontologies is PTIME-complete in data complexity (see saturated models by [116]).

We now aim to show that even when considering the simple description language *DL-Lite*$_{core}$ answering UCQ$^{\neg s}$ becomes undecidable. Let us illustrate the main idea for the proof by means of the following two examples. Consider first the following Boolean CQ$^{\neg s}$ $\boldsymbol{q}_2$ (in its negated form):

$$\mathsf{tape}(x_2, y_2) \rightarrow r(y_2, x_2). \tag{6.13}$$

It can be easily seen that $\mathcal{I} \not\models \boldsymbol{q}_2$ if and only if $\mathcal{I} \models \mathsf{tape}^{-} \sqsubseteq r$, for any interpretation $\mathcal{I}$. Thus, one can think of a role inclusion as the negation of a CQ$^{\neg s}$. It the follows from Proposition 6.1 that we can encode any $\mathcal{ELI}_{\bot}$ CI of the form $A \sqsubseteq C$, where A is a concept name, as a *DL-Lite*$_{core}$ TBox and a Boolean UCQ$^{\neg s}$ $\boldsymbol{q}$, such that a set of role inclusions encoded by $\boldsymbol{q}$ is satisfied by an interpretation $\mathcal{I}$ if and only if one of the corresponding queries has a positive answer in $\mathcal{I}$.

We can give the same treatment to $\mathcal{ELI}$ CIs of the form $B_1 \sqcap \exists r.B_2 \sqsubseteq A$. It is not difficult to see that this CI is satisfied in an interpretation $\mathcal{I}$ if an only the following Boolean CQ$^{\neg s}$ has a negative answer in $\mathcal{I}$:

$$\exists x, y \, \big(B_1(x) \wedge r(x, y) \wedge B_2(y) \wedge \neg A(x)\big).$$

Then basically, we can think of concept inclusions of the form $C \sqcap A$, where A is a concept name and C an $\mathcal{ELI}$ concept, simply as queries with *one* safe negation. Summing up, any $\mathcal{ELI}_{\bot}$ concept inclusion can be encoded as a *DL-Lite*$_{core}$ TBox and a Boolean UCQ$^{\neg s}$. We then can show the following lemma.

Lemma 6.3. *For any $\mathcal{ELI}_\bot$ TBox $\mathcal{T}$, one can construct a DL-Lite$_{core}$ TBox $\mathcal{T}'$ and a Boolean UCQ$^{\neg s}$ $\boldsymbol{q}'$ such that*

- *every model $\mathcal{I}$ of $\mathcal{T}'$ with $\mathcal{I} \not\models \boldsymbol{q}'$ is also a model of $\mathcal{T}$ and*
- *every model of $\mathcal{T}$ can be extended to a model $\mathcal{I}$ of $\mathcal{T}'$ with $\mathcal{I} \not\models \boldsymbol{q}'$ by interpreting fresh symbols of $\mathcal{T}'$.*

As a corollary of Theorem 6.2 and Lemma 6.3 we immediately obtain undecidability of answering *unions* of CQ$^{\neg s}$s over *DL-Lite$_{core}$* ontologies.

Corollary 6.4. *There is a Boolean UCQ$^{\neg s}$, such that each disjunct has one safe negation, and a DL-Lite$_{core}$ TBox such that answering UCQs$^{\neg s}$ is undecidable.*

A similar result was shown in [116, Theorem 15]. However, we observe that the result we present here is somewhat stronger in the sense that each disjunct of the UCQ$^{\neg s}$ has *one* safe negation, in contrast to the reduction in [116], where the number of negations depend on the tiling problem instance encoded.

6.1.2 From UCQs to CQs: the Case of *DL-Lite*$^{\mathcal{H}}_{core}$

We now proceed to show that under rather mild restrictions on the TBox, any union of tree-shaped Boolean CQ$^{\neg s}$s can be transformed into a single Boolean CQ$^{\neg s}$ that has the same answers over ontologies with TBoxes *extended* by a number of concept and role inclusions. This will allow us to obtain undecidability of answering a *single* CQ$^{\neg s}$ over *DL-Lite*$^{\mathcal{H}}_{core}$ using a similar reduction that the one used in Section 6.1.1. Observe that in contrast to what we showed in Corollary 6.4 we consider a single CQ$^{\neg}$, but we require role inclusions in the encoding.

We illustrate the transformation by considering a Boolean UCQ$^{\neg s}$ $\boldsymbol{q}$ comprised of the two queries from Section 6.1.1 $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ given in negated form in (6.2) and (6.13):

$$\boldsymbol{q}_1 = \exists x_1, y_1, z_1, u_1 \left(\mathsf{next}(x_1, y_1) \land \mathsf{tape}(x_1, z_1) \land \mathsf{next}(z_1, u_1) \land \neg\mathsf{tape}(y_1, u_1)\right),$$
$$\boldsymbol{q}_2 = \exists x_2, y_2 \left(\mathsf{tape}(x_2, y_2) \land \neg r(y_2, x_2)\right)$$

Using a fresh variable x and fresh roles g_1 and g_2, we can merge these two queries into a single Boolean CQ$^{\neg s}$ $\boldsymbol{q}'$, that consists of all the atoms of $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ together with the atoms $\mathsf{g}_1(x, x_1)$ and $\mathsf{g}_2(x, x_2)$ (see the right side of Figure 6.3). Note that the sets of variables in $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ are disjoint, and hence no connection between the primal graphs of the constituents is introduced with the merge.

The resulting CQ$^{\neg s}$ $\boldsymbol{q}'$ in general is not equivalent to $\boldsymbol{q}$. However, we can guarantee that for any TBox $\mathcal{T}$ the union $\boldsymbol{q}$ has the same answers over $(\mathcal{T}, \mathcal{A})$ as $\boldsymbol{q}'$ over an extended ontology $(\mathcal{T} \cup \mathcal{T}', \mathcal{A})$, for some suitable TBox $\mathcal{T}'$. This TBox is constructed in such a way that every model $\mathcal{J}$ of $(\mathcal{T} \cup \mathcal{T}', \mathcal{A})$ is obtained from a model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$ and satisfies the following properties:

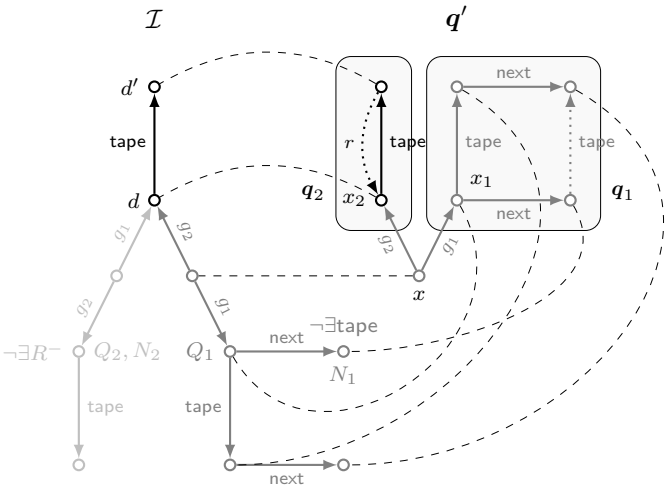
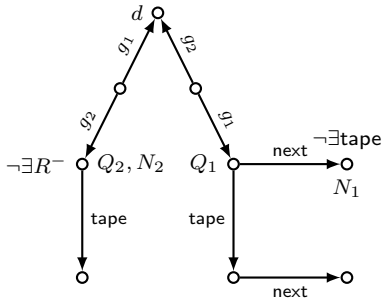


Figure 6.3: Matching $\text{CQ}^{\neg s}$ $\boldsymbol{q}'$ obtained from $\boldsymbol{q}_1 \vee \boldsymbol{q}_2$ in the extended model.

1. every domain element in $\mathcal{I}$ is in the extension of a special concept name D introduced in $\mathcal{T}'$, which can be achieved by adding the following CIs, for every concept name A and role name p in $\mathcal{T}$:

$$A \sqsubseteq D \qquad \exists P \sqsubseteq D \qquad \exists P^- \sqsubseteq D \tag{6.14}$$

2. every element d in the extension of D has attached the following tree:



which can be achieved by adding the following CIs to $\mathcal{T}'$:

$$\begin{aligned} D &\sqsubseteq \exists \mathsf{g}_2^-.\exists \mathsf{g}_1.Q_1, \\ Q_1 &\sqsubseteq \exists \mathsf{tape}.\exists \mathsf{next} \sqcap \exists \mathsf{next}.N_1, \\ D &\sqsubseteq \exists \mathsf{g}_1^-.\exists \mathsf{g}_2.Q_2, \\ Q_2 &\sqsubseteq \exists \mathsf{tape} \sqcap N_2, \\ N_1 \sqcap \exists \mathsf{tape} &\sqsubseteq \bot \qquad (6.15) \\ N_2 \sqcap \exists r^- &\sqsubseteq \bot. \qquad (6.16) \end{aligned}$$

where Q_1, N_1, Q_2 and N_2 are fresh concept names and g_1, g_2 fresh role names. Intuitively, each branch of the tree above encodes one query $\boldsymbol{q}_i$, $i \in \{1, 2\}$.

Observe that point 2 above ensures that in every model (satisfying D) of $\mathcal{T}'$ there is a match of *all* the atoms in $\boldsymbol{q}'$ *except* for those in $\boldsymbol{q}_2$ on the left brach of the tree above; similarly, there is a match of *all* the atoms in $\boldsymbol{q}'$ *except* for those in $\boldsymbol{q}_1$ on the right branch. The CIs in (6.15) and (6.16) ensure that the negative atoms of $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ can be matched by the respective parts of the model of $\mathcal{T}'$.

To see why the TBox $\mathcal{T}'$ above serves our purpose, assume that there is a model $\mathcal{I}$ of $\mathcal{T}$ with a single tape-edge between domain elements d and d' (see the left side of Figure 6.3). Then in a extension of $\mathcal{I}$ to a model of $\mathcal{T} \cup \mathcal{T}'$, d would be an instance of concept D and we we would also have that d has a dark-grey fragment attached to it to match all atoms of $\boldsymbol{q}'$ but $\boldsymbol{q}_2$ and a light-grey fragment to match all atoms of $\boldsymbol{q}'$ but $\boldsymbol{q}_1$. (actually, d' should also be in the interpretation of D and, hence, have similar fragments in the extended interpretation, but they are not depicted to reduce clutter). Observe that in the extended model, there is a match for $\boldsymbol{q}'$. Indeed, the atoms in $\boldsymbol{q}_2$ are matched in the original $\mathcal{I}$ and the rest of $\boldsymbol{q}'$ is matched to the dark-gray fragment in the extension of $\mathcal{I}$.

Following the latter observation, it should not be difficult to see that there is a match of $\boldsymbol{q}$ in a model $\mathcal{I}$ of $\mathcal{T}$ if and only if there is a match of $\boldsymbol{q}'$ in the interpretation $\mathcal{J}$ obtained by extending $\mathcal{I}$ to a model of $\mathcal{T} \cup \mathcal{T}'$. Indeed, if there is a match $\boldsymbol{q}$ in $\mathcal{I}$, then there is a match of $\boldsymbol{q}_i$, for some $i \in \{1, 2\}$. Then, (i) there is a match of $\boldsymbol{q}_1$ over the interpretation of D in $\mathcal{J}$ and the rest of $\boldsymbol{q}'$ is matched by the dark-grey fragment *or* (ii) there is a match of $\boldsymbol{q}_2$ over the interpretation of D and the rest of $\boldsymbol{q}'$ is matched by the dark-grey fragment. But then, there is a match of $\boldsymbol{q}'$ in $\mathcal{J}$.

We now generalize the construction above, and show that we can apply transform a $\text{UCQ}^{\neg s}$ with an arbitrary number of tree-shaped disjuncts into a $\text{CQ}^{\neq s}$ and a TBox.

Lemma 6.5. *Let $\mathcal{T}$ be a DL-Lite$^{\mathcal{H}}_{\text{core}}$ TBox and $\boldsymbol{q}$ a Boolean UCQ$^{\neg s}$ such that each disjunct $\boldsymbol{q}_i$ of $\boldsymbol{q}$ is tree-shaped. Then there exist a DL-Lite$^{\mathcal{H}}_{\text{core}}$ TBox $\mathcal{T}'$ and a CQ$^{\neg s}$ $\boldsymbol{q}'$ such that*

$$(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q} \quad \textit{iff} \quad (\mathcal{T} \cup \mathcal{T}', \mathcal{A}) \models \boldsymbol{q}', \quad \textit{for every ABox } \mathcal{A}.$$

Proof. We assume that each negative atom in each $\boldsymbol{q}_i$ contains a *loose* variable, that is a variable z such that $\mathcal{T} \not\models B_1 \sqsubseteq B_2$, for each positive atom $B_1(z)$ and each negative atom $\neg B_2(z)$ in $\boldsymbol{q}_i$ (to simplify notation, B_i refer here to basic concepts and we assume that $\boldsymbol{q}$ contains unary 'atoms' $\exists r(z_1)$ and $\exists r^-(z_2)$ if it contains a binary atom $r(z_1, z_2)$ and similarly for negative binary atoms). This assumption can be made without loss of generality, because every $\boldsymbol{q}_i$ with a negative atom without a loose variable always gives a negative answer on any interpretations satisfying $\mathcal{T}$ and hence, can be removed from $\boldsymbol{q}$ without affecting its answers.

Let $\boldsymbol{q}_i$ be of the form $\exists \vec{y}_i \, \varphi_i(\vec{y}_i)$, for $1 \leq i \leq n$. Since tree-shaped queries contain no individuals, each $\vec{y}_i$ is non-empty and we can fix a variable, say, y_{i1}, in each $\vec{y}_i$. Let y be a fresh

variable and, for each $1 \leq i \leq n$, let g_i be a fresh role name. Define $\varphi'_i(y, \vec{y}_i) = g_i(y, y_{i1}) \wedge \hat{\varphi}_i(\vec{y}_i)$, where $\hat{\varphi}_i$ is the result of replacing each concept name A with a fresh $\hat{A}$ and each role name p with a fresh $\hat{p}$ in φ_i. Consider

$$\boldsymbol{q}' \quad = \quad \exists y\, \vec{y}_1 \dots \vec{y}_n \bigwedge_{1 \leq i \leq n} \varphi'_i(y, \vec{y}_i).$$

Let D be a fresh concept name. Let $\mathcal{T}_D$ consist of $A \sqsubseteq \hat{A}$ and $A \sqsubseteq D$, for each concept name A occurring in $\mathcal{T}$ or $\boldsymbol{q}$, and $p \sqsubseteq \hat{p}$, $\exists p \sqsubseteq D$ and $\exists p^- \sqsubseteq D$, for each role name P occurring in $\mathcal{T}$ or $\boldsymbol{q}$. So, in any model of $\mathcal{T}_D$, the interpretation of D contains the interpretations of all concepts of $\mathcal{T}$ and $\boldsymbol{q}$ including domains and ranges of its roles.

Since the positive part of each $\varphi'_i(y, \vec{y}_i)$ is tree-shaped, we can assume that its primal graph is a rooted tree with root y (so each edge has a natural orientation away from the root); moreover, that root has a single successor, y_{i1}. We write $z \prec z'$ if z is a (unique) immediate predecessor of z' in such a tree from the corresponding $\varphi'_i(y, \vec{y}_i)$. For each edge (z, z') with $z \prec z'$, we take a fresh role $e_{zz'}$. Let $\mathcal{T}_G$ contain the following inclusions, for all $1 \leq i \leq n$:

$$D \sqsubseteq \exists g_{i,0}^-, \tag{6.17}$$

$$\exists g_{i,0} \sqsubseteq \exists g_{j,1}, \qquad \text{for } 1 \leq j \leq n \text{ with } j \neq i, \tag{6.18}$$

$$g_{i,k} \sqsubseteq g_i, \qquad \text{for } k = 0, 1, \tag{6.19}$$

$$g_{i,1} \sqsubseteq e_{y y_{i1}}, \tag{6.20}$$

$$\exists e_{zz'}^- \sqsubseteq \exists e_{z'z''}, \qquad \text{for } z \prec z' \prec z'', \tag{6.21}$$

$$\exists e_{zz'}^- \sqsubseteq \hat{A}, \qquad \text{for all } \hat{A}(z') \text{ in } \hat{\varphi}_i, \tag{6.22}$$

$$e_{zz'} \sqsubseteq \hat{r}, \qquad \text{for all } \hat{r}(z, z') \text{ in } \hat{\varphi}_i, \tag{6.23}$$

$$\exists e_{zz'}^- \sqcap \hat{A} \sqsubseteq \bot, \qquad \text{for all } \neg\hat{A}(z') \text{ in } \hat{\varphi}_i, \tag{6.24}$$

$$\exists e_{zz'}^- \sqcap \exists \hat{r} \sqsubseteq \bot, \qquad \text{for all } \neg\hat{r}(z', z'') \text{ in } \hat{\varphi}_i \text{ with loose } z', \tag{6.25}$$

where $g_{i,0}$ and $g_{i,1}$ are fresh role names. Let $\mathcal{T}' = \mathcal{T}_D \cup \mathcal{T}_G$. Note that it is crucial that z' is loose in both (6.24) and (6.25)—for otherwise $\mathcal{T} \cup \mathcal{T}'$ would imply emptiness of any interpretation of D. It remains to show that $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}$ if and only if $(\mathcal{T} \cup \mathcal{T}', \mathcal{A}) \models \boldsymbol{q}'$.

Suppose that $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}$ and let $\mathcal{I}$ be a model of $(\mathcal{T} \cup \mathcal{T}', \mathcal{A})$. As $\mathcal{I} \models (\mathcal{T}, \mathcal{A})$, we have $\mathcal{I} \models \boldsymbol{q}$. Then, there exists a match π for some disjunct $\boldsymbol{q}_i$ $(1 \leq i \leq n)$ of $\boldsymbol{q}$ in $\mathcal{I}$. Since $\boldsymbol{q}$ is $\mathrm{UCQ}^{\neg s}$, $\pi(y_{i1})$ belongs to $A^{\mathcal{I}}$, for some concept name A of $\mathcal{T}$, or to $(\exists r)^{\mathcal{I}}$, for some role r of $\mathcal{T}$; whence, $\pi(y_{i1}) \in D^{\mathcal{I}}$. Let $\boldsymbol{q}_*$ consist of all atoms of $\boldsymbol{q}'$ that are not in $\hat{\varphi}_i(\vec{y}_i)$. Since $\mathcal{I} \models \mathcal{T}_G$, there exists a match π_* for $\boldsymbol{q}_*$ in $\mathcal{I}$ with $\pi_*(y_{i1}) = \pi(y_{i1})$. Indeed, by (6.20)–(6.22), the tree of positive atoms of $\boldsymbol{q}_*$ is matched by the tree rooted in the $g_{i,0}^-$-successor of $\pi(y_{i1})$; by (6.24) and (6.25), the negative atoms are also satisfied by π_*. Hence, $\pi \cup \pi_*$ is a match for $\boldsymbol{q}'$ in $\mathcal{I}$.

Conversely, let $\mathcal{I}$ be a model of $(\mathcal{T}, \mathcal{A})$ with $\mathcal{I} \not\models \boldsymbol{q}$. Denote by $\mathcal{I}_0$ an interpretation that coincides with $\mathcal{I}$ on all individuals and concept and role names of $\mathcal{T}$ or $\boldsymbol{q}$ and, additionally, interprets D by $\Delta^{\mathcal{I}}$, $\hat{A}$ by $A^{\mathcal{I}}$, for each concept name A in $\mathcal{T}$ or $\boldsymbol{q}$, and $\hat{p}$ by $p^{\mathcal{I}}$, for each role name p in $\mathcal{T}$ or $\boldsymbol{q}$. By construction, $\mathcal{I}_0 \models (\mathcal{T} \cup \mathcal{T}_D, \mathcal{A})$ and $\mathcal{I}_0 \not\models \boldsymbol{q}$. Denote by $\mathcal{I}_d^g$ the canonical

interpretation of $(\mathcal{T}_G, \{D(d)\})$, for $d \in \Delta^{\mathcal{I}_0}$ (we slightly abuse notation here and treat domain elements as fresh individual names assuming that $d^{\mathcal{I}_d^g} = d$). By definition, each $\mathcal{I}_d^g$ is finite and their domains are pairwise disjoint. Let $\mathcal{I}'$ be the union of $\mathcal{I}_0$ with all $\mathcal{I}_d^g$, $d \in \Delta^{\mathcal{I}_0}$. Since each negative atom of $\boldsymbol{q}$ contains a loose variable, $\mathcal{I}'$ does not violate any negative inclusions of $\mathcal{T}_G$, that is, (6.24) and (6.25). Thus, $\mathcal{I}' \models (\mathcal{T} \cup \mathcal{T}', \mathcal{A})$. Finally, for the sake of contradiction, suppose $\mathcal{I}' \models \boldsymbol{q}'$. Then there is a match π for $\boldsymbol{q}'$ in $\mathcal{I}'$. By the definition of $\boldsymbol{q}'$, $\pi(y)$ must be the element in one of $\mathcal{I}_d^G$ introduced to witness the existential quantifier in (6.17). By (6.18), atoms corresponding to one of the components, say $\boldsymbol{q}_i$, of $\boldsymbol{q}$ must be matched in the part of the original model $\mathcal{I}_0$, contrary to $\mathcal{I}_0 \not\models \boldsymbol{q}_i$, for all i, $1 \leq i \leq n$. ❑

It can be easily verified that the UCQ$^{\neg s}$ and the TBox obtained in the proof of Corollary 6.4 from the proofs of Theorem 6.2 and Lemma 6.3 satisfy the conditions of Lemma 6.5. Thus, we obtain undecidability of answering CQ$^{\neg s}$ over *DL-Lite*$_{core}^{\mathcal{H}}$ ontologies.

Theorem 6.6. *There is a Boolean CQ$^{\neg s}$ and a DL-Lite$_{core}^{\mathcal{H}}$ TBox such that answering CQs$^{\neg s}$ is undecidable.*

This solves the open problem of decidability of CQ$^{\neg s}$ answering over *DL-Lite*$_{core}^{\mathcal{H}}$ [116]. However, since role inclusions are required in the transformation of unions of CQ$^{\neg s}$s to a single CQ$^{\neg s}$, the decidability of the query answering problem over *DL-Lite*$_{core}$ ontologies remains open. On the other hand, by Corollary 6.4, answering *unions* of CQ$^{\neg s}$ over *DL-Lite*$_{core}$ is undecidable. However, the number of CQ$^{\neg s}$s in the union constructed in the proof of Corollary 6.4 depends on the number of states and the size of the alphabet of the universal Turing machine (more precisely, it is $4 + (2 \cdot |Q| + 1) \cdot |\Gamma|$). We can slightly improve the result to a UCQ$^{\neg s}$ with only three disjuncts.

Theorem 6.7. *There is a union of three CQ$^{\neg s}$ and a DL-Lite$_{core}$ TBox, such that answering UCQs$^{\neg s}$ is undecidable.*

Proof. The proof again is by reduction of the halting problem for deterministic Turing machines. Let $M = (\Gamma, Q, q_0, q_1, \delta)$ be a deterministic Turing machine, where Γ is an alphabet (containing the blank symbol $\sqcup$), Q is a set of states, $q_0 \in Q$ and $q_1 \in Q$ are an initial and an accepting state, respectively, and $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$ is a transition function. We use $\emptyset$ for the no-head marker, $*$ for the special tape initialisation marker and abbreviate pairs $(q, a) \in (Q \cup \{\emptyset, *\}) \times \Gamma$ simply as qa.

We represent computations of M in a two-dimensional grid, where role tape points to the representation of the next cell on the tape and role next to the representation of the same cell in the successor configuration. To describe the contents of a cell, current state and head's position, we use a role $\mathsf{content}$, which relates the representation of a cell containing $a \in \Gamma$ in a configuration with state $q \in Q$ with a head over the cell to an individual e_{qa}; if the head is not positioned over the cell then the representation of this cell is $\mathsf{content}$-related to $e_{\emptyset a}$; the representation of the cells in the initial configuration beyond the input word is $\mathsf{content}$-related to a special individual $e_{*\sqcup}$.

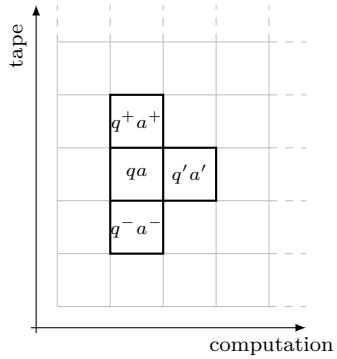


Figure 6.4: Quadruples τ of the form $(q^+a^+, qa, q^-a^-, q'a')$.

Consider quadruples of the form $(q^+a^+, qa, q^-a^-, q'a')$ that are possible in a computation of M (the superscript '$-$' refers to the previous cell on the tape, '$+$' to the next cell on the tape and $'$ to the same cell in the successor configuration; see Fig. 6.4). Such quadruples clearly satisfy one of the following conditions:

$$q \neq \emptyset, \quad \delta(q, a) = (q', a', \sigma) \text{ and } q' = \emptyset;$$

$$q^+ \neq \emptyset, \quad \delta(q^+, a^+) = (q'', a'', +1), \quad q' = \emptyset \text{ and } a' = a;$$

$$q^+ \neq \emptyset, \quad \delta(q^+, a^+) = (q', a'', -1) \text{ and } a' = a;$$

$$q^- \neq \emptyset, \quad \delta(q^-, a^-) = (q'', a'', -1), \quad q' = \emptyset \text{ and } a' = a;$$

$$q^- \neq \emptyset, \quad \delta(q^-, a^-) = (q', a'', +1) \text{ and } a' = a;$$

$$q^- = q = q^+ = q' = \emptyset \text{ and } a' = a.$$

On the other hand, in each computation, the pair $q'a'$ assigned to a cell in any configuration but initial is determined by three pairs from the preceding configuration: the pair qa for the same cell, the pair q^+a^+ for the next cell and the pair q^-a^- for the previous cell on the tape (since the machine is deterministic, the pair $q'a'$ is defined uniquely). We will also need special quadruples for initialisation of the tape of the Turing machine beyond the input word:

$$(*_\sqcup, \emptyset a, \emptyset a', \emptyset a), \qquad \text{for } a, a' \in \Gamma,$$

$$(*_\sqcup, *_\sqcup, \emptyset a, \emptyset_\sqcup), \qquad \text{for } a \in \Gamma,$$

$$(*_\sqcup, *_\sqcup, *_\sqcup, \emptyset_\sqcup).$$

We assume that the input word contains more than two symbols and therefore $*_\sqcup$ does not appear in the first three cells.

Take an individual name e_{qa} for each pair qa and an individual name e_τ for each of the quadruples τ, satisfying the conditions above. Let p, p_+, p_- and p' be role names and let ABox

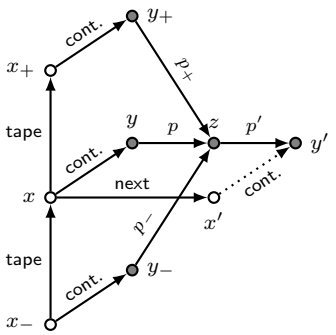


Figure 6.5: Second query in the proof of Theorem 6.7.

$\mathcal{A}_M$ contain the following assertions

$$p(e_{qa}, e_\tau), p_+(e_{q^+a^+}, e_\tau), p_-(e_{q^-a^-}, e_\tau), p'(e_\tau, e_{q'a'}),$$

for each τ of the form $(q^+a^+, qa, q^-a^-, q'a')$ satisfying the conditions above and for each of the tape initialization quadruples.

Another ABox, $\mathcal{A}_{\vec{w}}$, encodes the input $\vec{w} = a_1, \dots, a_n \in \Gamma^*$ on the tape as follows, where $c_1, \dots, c_n$ are fresh individual names, corresponding to the cells of the input, c_0 is a new special individual 'before' the start of the tape, and I is a new concept name for initialisation of the tape beyond the input:

$$\begin{aligned} &\mathsf{tape}(c_0, c_0), \quad \mathsf{content}(c_0, e_{\emptyset_\sqcup}), \qquad \mathsf{tape}(c_0, c_1), \quad \mathsf{content}(c_1, e_{q_0 a_1}), \\ &\qquad\qquad \mathsf{tape}(c_{i-1}, c_i), \quad \mathsf{content}(c_i, e_{\emptyset a_i}), \text{ for } 1 < i \leq n, \\ &\qquad\qquad\qquad\qquad \mathsf{tape}(c_n, c_{n+1}), \quad \mathsf{content}(c_{n+1}, e_{*_\sqcup}), \quad I(e_{*_\sqcup}). \end{aligned}$$

The third part of the ABox, $\mathcal{A}_A$, contains $\neg A(e_{q_1 a})$, for $a \in \Gamma$.

Consider now the UCQ$^{\neg s}$ $\boldsymbol{q}$ with the following negated form of its three CQ$^{\neg s}$s (see Fig. 6.5 for the second query):

$$\mathsf{next}(x, y) \land \mathsf{tape}(x, z) \land \mathsf{next}(z, u) \to \mathsf{tape}(y, u),$$

$$\begin{aligned} \mathsf{next}(x, x') \land \mathsf{content}(x, y) \land p(y, z) \land \\ \mathsf{tape}(x, x_+) \land \mathsf{content}(x_+, y_+) \land p_+(y_+, z) \land \\ \mathsf{tape}(x_-, x) \land \mathsf{content}(x_-, y_-) \land p_-(y_-, z) \land \\ p'(z, y') \to \mathsf{content}(x', y'), \end{aligned}$$

$$\mathsf{tape}(y, x) \land \mathsf{content}(y, v) \land I(v) \to \mathsf{content}(x, v).$$

Let TBox $\mathcal{T}_M$ contain

$$\exists \mathsf{tape} \sqsubseteq \exists \mathsf{next}, \qquad \exists \mathsf{tape}^- \sqsubseteq \exists \mathsf{tape}, \qquad \exists \mathsf{content}^- \sqsubseteq A.$$

It remains to show the following:

$(\mathcal{T}_M, \mathcal{A}_M \cup \mathcal{A}_{\vec{w}} \cup \mathcal{A}_A) \not\models \boldsymbol{q}$ iff M does not accept $\vec{w}$.

($\Rightarrow$) Consider a model $\mathcal{I}$ of $(\mathcal{T}_M, \mathcal{A}_M \cup \mathcal{A}_{\vec{w}} \cup \mathcal{A}_A)$ with $\mathcal{I} \not\models \boldsymbol{q}$. Then there exists an infinite sequence of (not necessarily distinct) domain elements $d_0, d_1, d_2, \dots$ that encodes the initial configuration in a sense that $(d_i, d_{i+1}) \in \mathsf{tape}^{\mathcal{I}}$ for all $i \geq 0$, and each element is connected by the interpretation of $\mathsf{content}$ to the element of the corresponding pair, that is, $\mathsf{content}^{\mathcal{I}}$ contains $(d_0, e^{\mathcal{I}}_{\emptyset_\sqcup})$, $(d_1, e^{\mathcal{I}}_{q_0 a_1})$, all $(d_i, e^{\mathcal{I}}_{\emptyset a_i})$, for $1 < i \leq n$, and all $(d_i, e^{\mathcal{I}}_{*_\sqcup})$, for $i > n$. Note that $d_0 = c_0^{\mathcal{I}}$ is an auxiliary element before the tape, whose role is to match the (positive part of the) second query for the representation of the first cell, and $e_{*_\sqcup}$ serves as a substitute for $e_{\emptyset_\sqcup}$, which is necessary, along with concept I and the third query, to initialise the tape beyond the input. By the first TBox inclusion, there exists a sequence of elements $d'_0, d'_1, d'_2, \dots$ such that $(d_i, d'_i) \in \mathsf{next}^{\mathcal{I}}$. By the first disjunct of the query, they form another tape-connected sequence, that is, $(d'_i, d'_{i+1}) \in \mathsf{tape}^{\mathcal{I}}$ for all i. By $\mathcal{A}_M$ and the second disjunct of the query, the sequence represents the second configuration of M in the same way, except that now $e_{*_\sqcup}$ is not used; instead, by the tape initialization quadruples, all the cells beyond the working space are $\mathsf{content}$-connected to $e_{\emptyset_\sqcup}$. By the same reasoning, there exists a sequence of elements for each configuration of the computation of M. Finally, $\mathcal{A}_A$ guarantees that the accepting state never occurs in the computation, and so, M does not accept $\vec{w}$.

($\Leftarrow$) If M has a non-accepting computation on $\vec{w}$ then it is routine to construct an infinite two-dimensional grid-like interpretation $\mathcal{I}$ satisfying $(\mathcal{T}_M, \mathcal{A}_M \cup \mathcal{A}_{\vec{w}} \cup \mathcal{A}_A)$ but not $\boldsymbol{q}$ (note that all domain elements of the bottom row of the grid have a $\mathsf{tape}^{\mathcal{I}}$-loop).

This finishes the proof of Theorem 6.7. ❑

We note in passing that the query $\boldsymbol{q}$ in the proof of Theorem 6.7 is not tree-shaped and therefore Lemma 6.5 is not applicable.

6.1.3 Queries with Guarded Negation

In this section, we narrow the class of CQs with safe negation and concentrate on CQs with guarded negation. As follows from the results of Bárány et al. [19], answering unions of GNCQs over ontologies expressed with so-called *frontier-guarded tgds* [16] is decidable in coNP in data complexity and, moreover, is in P in data complexity if the query contains one negated atom. Since frontier-guarded tgds subsume $\mathcal{ELI}$ concept and role inclusions, and since negative concept inclusions can be viewed as GNCQs, the upper bounds also apply to $\mathcal{ELI}$ and *DL-Lite*$^{\mathcal{H}}_{core}$ TBoxes. We next prove that the lower bounds match these results even with a TBox containing a single negative concept inclusion.

Lemma 6.8. *There is a Boolean GNCQ with one negated atom and a DL-Lite*$_{core}$ *TBox such that query answering answering is* P*-hard.*

Proof. The proof is by reduction of the complement of Horn-3SAT, the satisfiability problem for Horn clauses with at most 3 literals, which is known to be P-complete; see e.g., [107]. Suppose we are given a conjunction ψ of clauses of the form p, $\neg p$, and $p_1 \wedge p_2 \to p$. Consider the following Boolean GNCQ $\boldsymbol{q}$ in the negated form:

$$p_1(x_1, y) \wedge V(x_1) \wedge p_2(x_2, y) \wedge V(x_2) \wedge p(y, z) \to V(z).$$

Note that $\boldsymbol{q}$ does not depend on ψ. Let $\mathcal{T}_0$ be the TBox containing a single negative concept inclusion: $V \sqcap F \sqsubseteq \bot$. Next, we construct an ABox $\mathcal{A}_\psi$ such that ψ is satisfiable iff $(\mathcal{T}_0, \mathcal{A}_\psi) \not\models \boldsymbol{q}$. The ABox $\mathcal{A}_\psi$ uses an individual name c_p, for each variable p in ψ, and an individual name c_γ for each clause γ of the form $p_1 \wedge p_2 \to p$ in ψ. For each clause γ, the ABox $\mathcal{A}_\psi$ contains the following assertions:

$$\begin{aligned} V(c_p), &\quad \text{if } \gamma = p, \\ F(c_p), &\quad \text{if } \gamma = \neg p, \\ p_1(c_{p_1}, c_\gamma),\ p_2(c_{p_2}, c_\gamma),\ p(c_\gamma, c_p), &\quad \text{if } \gamma = p_1 \wedge p_2 \to p. \end{aligned}$$

Suppose first there is a model $\mathcal{I}$ of $(\mathcal{T}_0, \mathcal{A}_\psi)$ with $\mathcal{I} \not\models \boldsymbol{q}$. It can be easily shown that ψ is satisfiable: for each clause γ of ψ of the form $p_1 \wedge p_2 \to p$ (the other two cases are trivial), if $c_{p_1}^{\mathcal{I}} \in V^{\mathcal{I}}$ and $c_{p_2}^{\mathcal{I}} \in V^{\mathcal{I}}$ then $c_p \in V^{\mathcal{I}}$. Thus, we can define a satisfying assignment $\mathfrak{a}$ for ψ by taking $\mathfrak{a}(p)$ true iff $c_p^{\mathcal{I}} \in V^{\mathcal{I}}$.

Conversely, if ψ is satisfiable then we can construct a model $\mathcal{I}$ of $(\mathcal{T}_0, \mathcal{A}_\psi)$ with $\mathcal{I} \not\models \boldsymbol{q}$. ❑

Lemma 6.9. *There is a Boolean GNCQ with two negated atoms and a DL-Lite$_{core}$ TBox such that query answering is* coNP*-hard.*

Proof. The proof is by reduction of the complement of 2+2CNF, the satisfiability problem for clauses with two positive and two negative literals, which is known to be NP-complete [120]. Suppose we are given a conjunction ψ of clauses of the form $p_1 \vee p_2 \vee \neg p_3 \vee \neg p_4$, where each p_i is either a propositional variable or one of the two truth constants, *true* and *false*. Consider the following Boolean GNCQ $\boldsymbol{q}$ in the negated form:

$$n_1(x, y_3) \ \wedge\ V(y_3) \ \wedge\ n_2(x, y_4) \ \wedge\ V(y_4) \ \wedge\ p_1(x, y_1) \ \wedge\ p_2(x, y_2) \quad \to \quad V(y_1) \ \vee\ V(y_2).$$

Note that $\boldsymbol{q}$ does not depend on ψ. Let $\mathcal{T}_0$ be the TBox with the single axiom $V \sqcap F \sqsubseteq \bot$. We construct an ABox $\mathcal{A}_\psi$ such that ψ is satisfiable iff $(\mathcal{T}_0, \mathcal{A}_\psi) \not\models \boldsymbol{q}$. The ABox $\mathcal{A}_\psi$ uses individual names c_{true} and c_{false} for the truth values, an individual name c_p, for each variable p in ψ, and an individual name c_γ for each clause γ in ψ. For each clause γ of the form $p_1 \vee p_2 \vee \neg p_3 \vee \neg p_4$, the ABox $\mathcal{A}_\psi$ contains the following assertions:

$$p_1(c_{p_1}, c_\gamma), \quad p_2(c_{p_2}, c_\gamma), \quad n_1(c_{p_3}, c_\gamma), \quad n_2(c_{p_4}, c_\gamma).$$

Also, $\mathcal{A}_\psi$ contains $V(c_{true})$ and $F(c_{false})$.

Suppose first there is a model $\mathcal{I}$ of $(\mathcal{T}_0, \mathcal{A}_\psi)$ with $\mathcal{I} \not\models \boldsymbol{q}$. It can be easily shown that ψ is satisfiable: indeed, for each clause of ψ of the form $p_1 \vee p_2 \vee \neg p_3 \vee \neg p_4$, if both $c^{\mathcal{I}}_{p_3} \in V^{\mathcal{I}}$ and $c^{\mathcal{I}}_{p_4} \in V^{\mathcal{I}}$ then we have either $c^{\mathcal{I}}_{p_1} \in V^{\mathcal{I}}$ or $c^{\mathcal{I}}_{p_2} \in V^{\mathcal{I}}$. Since $c^{\mathcal{I}}_{true} \in V^{\mathcal{I}}$ and $c^{\mathcal{I}}_{false} \notin V^{\mathcal{I}}$, we can define a satisfying assignment $\mathfrak{a}$ for ψ by taking $\mathfrak{a}(p)$ true iff $c^{\mathcal{I}}_p \in V^{\mathcal{I}}$.

Conversely, if ψ is satisfiable then we can construct a model $\mathcal{I}$ of $(\emptyset, \mathcal{A}_\psi)$ with $\mathcal{I} \not\models \boldsymbol{q}$. ❑

From Lemmata 6.8 and 6.9 and the upper bounds provided by [19] we obtain the following result.

Theorem 6.10. *Answering GNCQs over DL-Lite$_{core}$, DL-Lite$^{\mathcal{H}}_{core}$, $\mathcal{EL}$, $\mathcal{ELI}$ ontologies is* coNP*-complete in data complexity, and even ontologies with the empty TBox. It is* P*-complete if the query has at most one negation.*

6.2 Answering CQs with Inequalities

We start by analyzing the case of unions of conjunctive queries and show that query answering with inequalities is undecidable even in the simplest of *DL-Lite* languages. In fact, the following proof will demonstrate that even though the ontology language is quite unexpressive, undecidable problems can still be encoded by means of (mostly) UCQs.

Similarly to what we did in the previous section, we will take advantage to the expressive power of Boolean UCQs$^{\neq}$ in negated form. In a nutshell, the proof uses the existential quantifiers in the TBox to create an unbounded supply of elements, whereas the UCQ$^{\neq}$ in negated form allows one to express universal constraints of the form:

$$\varphi(\vec{y}) \rightarrow (t^1_1 = t^1_2 \vee \cdots \vee t^1_n = t^2_n)$$

Constraints of the form above have been called disjunctive EGDs in the literature [54]. The following result was first stated as Theorem 8 in [116], however no proof is provided there. For didactical reasons we provide a proof here, that will also illustrate the expressive power of UCQs$^{\neq}$.

Theorem 6.11. *Answering UCQs$^{\neq}$ is undecidable over DL-Lite$_{core}$ ontologies.*

Proof. The proof is by reduction of the (complement of) $\mathbb{N} \times \mathbb{N}$-tiling problem, which is known to be undecidable [65]. The $\mathbb{N} \times \mathbb{N}$ tiling problem is formulated as follows: given a set $\mathfrak{T}$ of square tile types with the four sides of each tile $t \in \mathfrak{T}$ coloured by $\mathit{top}(t), \mathit{right}(t), \mathit{bottom}(t), \mathit{left}(t)$, respectively, and a tile $t_0 \in \mathfrak{T}$, decide whether $\mathbb{N} \times \mathbb{N}$ can be tiled by $\mathfrak{T}$ with t_0 placed at the origin, i.e., whether there is a function $\tau \colon \mathbb{N} \times \mathbb{N} \to \mathfrak{T}$ such that $\tau(0,0) = t_0$ and $\mathit{top}(\tau(i,j)) = \mathit{bottom}(\tau(i,j+1))$ and $\mathit{left}(\tau(i,j)) = \mathit{right}(\tau(i+1,j))$, for all $(i,j) \in \mathbb{N} \times \mathbb{N}$.

Given an instance of the $\mathbb{N}\times\mathbb{N}$-tiling problem, we construct a *DL-Lite$_{core}$* ontology $(\mathcal{T}, \mathcal{A})$ that encodes the tiling problem by placing tiles over objects in its model. The top and right neighbours of a tile are referred to by roles horizontal and vertical, respectively (from the horizontal and

vertical successor). To represent the type of each tile we take ABox individuals t_i, for $t_i \in \mathfrak{T}$, and a role ttype that connects a tile to its type. So, the TBox $\mathcal{T}$ contains the following concept inclusions:

$$\exists \mathsf{ttype} \sqsubseteq \exists \mathsf{horizontal}, \quad \exists \mathsf{horizontal}^- \sqsubseteq \exists \mathsf{ttype}, \quad \exists \mathsf{ttype} \sqsubseteq \exists \mathsf{vertical}, \quad \exists \mathsf{vertical}^- \sqsubseteq \exists \mathsf{ttype}.$$

We also require two roles, $\mathsf{Nhorizontal}$ and $\mathsf{Nvertical}$, that define impossible horizontal and vertical tile neighbors: let $\mathcal{A}_{\mathfrak{T}}$ contain

$$\begin{aligned} &\mathsf{Nhorizontal}(t_i, t_j), && \text{for each } t_i, t_j \in \mathfrak{T} \text{ with } \mathit{right}(t_i) \neq \mathit{left}(t_j),\\ &\mathsf{Nvertical}(t_i, t_j), && \text{for each } t_i, t_j \in \mathfrak{T} \text{ with } \mathit{top}(t_i) \neq \mathit{bottom}(t_j). \end{aligned}$$

Consider now the Boolean UCQ$^{\neq}$ q, whose negation is equivalent to the conjunction of the following sentences:

$$\forall x, y \, \big(\mathsf{ttype}(x, y) \to \textstyle\bigvee_i (y = t_i)\big),$$

$$\forall x, y, z, v, u \, \big(\mathsf{horizontal}(x, y) \land \mathsf{vertical}(y, v) \land \mathsf{vertical}(x, z) \land \mathsf{horizontal}(z, u) \to (u = v)\big),$$

$$\forall x, y, x', y' \, \big(\mathsf{horizontal}(x, y) \land \mathsf{ttype}(x, x') \land \mathsf{ttype}(y, y') \land \mathsf{Nhorizontal}(x', y') \to \bot\big),$$

$$\forall x, y, x', y' \, \big(\mathsf{vertical}(x, y) \land \mathsf{ttype}(x, x') \land \mathsf{ttype}(y, y') \land \mathsf{Nvertical}(x', y') \to \bot\big).$$

It can be shown that $(\mathcal{T}, \mathcal{A}_{\mathfrak{T}} \cup \{T(a, t_0)\}) \not\models q$ iff $\mathfrak{T}$ tiles $\mathbb{N} \times \mathbb{N}$ with t_0 placed at the origin. Indeed, if q has a counter model then the above formulas guarantee that each tile object is related to one of the t_i, that the $\mathsf{horizontal}$- and $\mathsf{vertical}$-successors from a proper $\mathbb{N} \times \mathbb{N}$-grid and, finally, that the adjacent colors match. ❑

6.2.1 Answering CQs$^{\neq}$ in *DL-Lite*$^{\mathcal{H}}_{\mathit{core}}$

In this section, we prove that CQ$^{\neq}$ answering over *DL-Lite*$^{\mathcal{H}}_{\mathrm{core}}$ is undecidable. In principle, the technique of Lemma 6.5 can be adapted to queries with inequalities, and by modifying the proof of Theorem 6.11 one could prove the claim. The resulting CQ$^{\neq}$ would, however, contain many inequalities. Instead, we substantially rework some ideas of the undecidability proof for CQ$^{\neq}$ answering over $\mathcal{EL}$ [82] and show that even *one* inequality suffices for *DL-Lite*$^{\mathcal{H}}_{\mathrm{core}}$.

Theorem 6.12. *There is a Boolean CQ$^{\neq}$ with one inequality and a DL-Lite$^{\mathcal{H}}_{\mathrm{core}}$ TBox such that query answering is undecidable.*

Proof. The proof is by reduction of the halting problem for deterministic Turing machines (see Theorem 6.2). In this proof we use a two-dimensional grid of similar structure. The grid is established by means of a CQ$^{\neq}$ $\boldsymbol{q}$ with the following negated form:

$$\begin{aligned} \mathsf{next}(x, y) \land \mathsf{tape}(x, z) \land \mathsf{next}(z, v) \land \mathsf{tape}(y, u) \land \\ \mathsf{tape}(u, w) \land \mathsf{tape}(u', w) \land r(t, v) \land r(t, v') \to (u' = v'). \end{aligned}$$

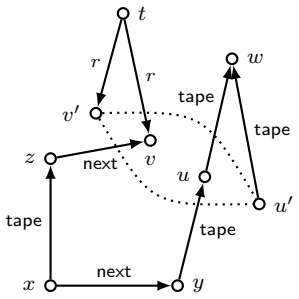


Figure 6.6: Query in the proof of Theorem 6.12.

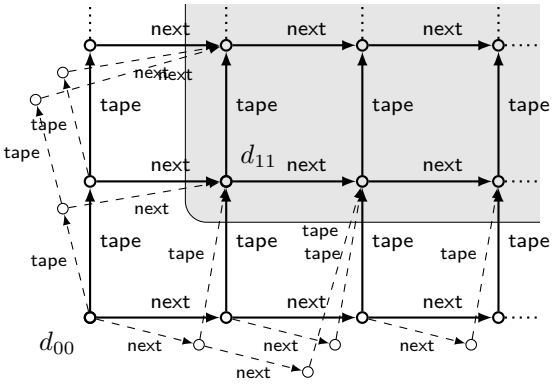


Figure 6.7: Grid structure in the proof of Theorem 6.12.

Note that this formula, in fact, implies $v = v' = u' = u$; see the dotted shape in Fig. 6.6.

We present the construction of a TBox $\mathcal{T}_M$ encoding the problem in a series of smaller TBoxes. As before, we aim to design $\mathcal{T}_M$ in such way that if it has a model $\mathcal{I}$, then it is a counter model of $\boldsymbol{q}$, that is $\mathcal{I} \not\models \boldsymbol{q}$. For each of the building blocks of $\mathcal{T}_M$ we then show that if $\mathcal{I}$ is a model of that "block" then $\mathcal{I}$ enjoys certain structural properties. We say that an interpretation $\mathsf{next}^{\mathcal{I}}$ of a role next is *functional in* $d \in \Delta^{\mathcal{I}}$ if $d' = d''$ whenever both (d, d') and (d, d'') are in $\mathsf{next}^{\mathcal{I}}$. We also denote composition of binary relations by $\circ$:

$$\mathsf{next}^{\mathcal{I}} \circ \mathsf{tape}^{\mathcal{I}} = \{(d, d'') \mid (d, d') \in \mathsf{next}^{\mathcal{I}}, (d', d'') \in \mathsf{tape}^{\mathcal{I}}\}.$$

Let TBox $\mathcal{T}_G$ contain the following concept inclusions:

$$\exists \mathsf{next}^- \sqsubseteq \exists \mathsf{tape}, \qquad \exists \mathsf{tape}^- \sqsubseteq \exists \mathsf{tape}, \qquad \exists \mathsf{next}^- \sqsubseteq \exists r^-.$$

We claim that if $\mathcal{I} \models \mathcal{T}_G$ and $\mathcal{I} \models \exists \mathsf{tape} \sqsubseteq \exists \mathsf{next}$ then the fragment of $\mathcal{I}$ rooted in $d_{00} \in (\exists \mathsf{tape})^{\mathcal{I}}$ has a grid-like structure depicted in Fig. 6.7 (each domain element in $(\exists \mathsf{next}^-)^{\mathcal{I}}$ also has an $r^{\mathcal{I}}$-predecessor, which is not shown).

More formally, the domain elements in the shaded area enjoy the following property.

Claim 5. If $\mathcal{I} \models \mathcal{T}_G$ and $\mathcal{I} \not\models \boldsymbol{q}$ then, for every d with an $(\mathsf{next}^-)^{\mathcal{I}} \circ \mathsf{tape}^{\mathcal{I}} \circ \mathsf{next}^{\mathcal{I}}$-predecessor,

- both $\mathsf{next}^{\mathcal{I}}$ and $\mathsf{tape}^{\mathcal{I}}$ are functional in d,
- the $\mathsf{next}^{\mathcal{I}} \circ \mathsf{tape}^{\mathcal{I}}$- and $\mathsf{tape}^{\mathcal{I}} \circ \mathsf{next}^{\mathcal{I}}$-successors of d coincide,
- $(\mathsf{tape}^{-})^{\mathcal{I}}$ is functional in the $\mathsf{tape}^{\mathcal{I}}$-successor of d,
- $r^{\mathcal{I}}$ is functional in any $r^{\mathcal{I}}$-predecessor of d.

So, $\mathsf{next}^{\mathcal{I}}$ and $\mathsf{tape}^{\mathcal{I}}$ are functional in all domain elements in the shaded area. However, $\mathsf{next}^{\mathcal{I}}$ does not have to be functional in the bottom row, and the same holds for $\mathsf{tape}^{\mathcal{I}}$ in the left column (see Fig. 6.7); $(\mathsf{tape}^{-})^{\mathcal{I}}$ does not have to be functional outside the shaded area and in the first row of the shaded area; $r^{\mathcal{I}}$ does not have to be functional anywhere but in $r^{\mathcal{I}}$-predecessors of the domain elements in the shaded area; $(\mathsf{next}^{-})^{\mathcal{I}}$ and $(r^{-})^{\mathcal{I}}$ do not have to be functional anywhere. For our purposes, however, it suffices that $\mathcal{I}$ has a grid structure starting from d_{11}; moreover, as we will see, the non-functionality of $(\mathsf{next}^{-})^{\mathcal{I}}$ plays a crucial role in the construction.

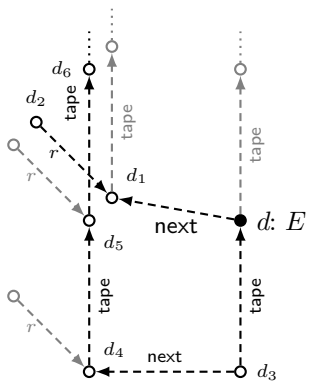
In addition to the grid-like structure of $\mathsf{next}^{\mathcal{I}}$ and $\mathsf{tape}^{\mathcal{I}}$, we also need functionality of $\mathsf{next}^{\mathcal{I}}$ in points outside the grid.

Claim 6. If $\mathcal{I}$ satisfies $\mathcal{T}_G$ and the following concept inclusion

$$E \sqsubseteq \exists \mathsf{tape}^{-}.\exists \mathsf{next} \tag{6.26}$$

and $\mathcal{I} \not\models \boldsymbol{q}$ then $\mathsf{next}^{\mathcal{I}}$ is functional in every $d \in E^{\mathcal{I}}$.

Proof of claim. Since $d \in E^{\mathcal{I}}$ and $\mathcal{I}$ satisfies $\mathcal{T}_G$ and (6.26), if d has an $\mathsf{next}^{\mathcal{I}}$-successor d_1 then the point d occurs in the fragment depicted below.



In particular, d_1 has an $r^{\mathcal{I}}$-predecessor d_2; d also has a $\mathsf{tape}^{\mathcal{I}}$-predecessor d_3, which has an $\mathsf{next}^{\mathcal{I}}$-successor d_4; and d_4, in its turn has a $\mathsf{tape}^{\mathcal{I}}$-successor d_5, which has a $\mathsf{tape}^{\mathcal{I}}$-successor d_6. But then, since $\mathcal{I} \not\models \boldsymbol{q}$, we obtain $d_5 = d_1$ and in fact, any $\mathsf{next}^{\mathcal{I}}$-successor of d will coincide with d_5 and d_1. ❑

We also require role r to be functional not only in $r^{\mathcal{I}}$-predecessors of the grid points but also in the grid points themselves.

Claim 7. If $\mathcal{I}$ satisfies $\mathcal{T}_G$ and the following concept inclusion

$$D \sqsubseteq \exists r.\exists \mathsf{next}^{-}.\exists \mathsf{tape}^{-}.\exists \mathsf{next} \tag{6.27}$$

and $\mathcal{I} \not\models \boldsymbol{q}$ then $r^{\mathcal{I}}$ is functional in every $d \in D^{\mathcal{I}}$.

Proof of claim. The argument is essentially the same as in the proof of Claim 6. ❑

We now describe a TBox that encodes computations of a given Turing machine. Let $M = (\Gamma, Q, q_0, q_1, \delta)$ be a deterministic Turing machine, where $\Gamma = \{1, \sqcup\}$ is a *two-symbol* tape alphabet, Q a set of states, $q_0 \in Q$ an initial and $q_1 \in Q$ an accepting state, and $\delta\colon Q{\times}\Gamma \to Q{\times}\Gamma{\times}\{-1, +1\}$ a deterministic transition function.

We use concept H_q, for $q \in Q$, that contains the representations of all tape cells observed by the head of M (in state q); concept $H_\emptyset$ represents the cells not observed by the head of M. Role next has two sub-roles, $s_\sqcup$ and s_1, for the two symbols of the alphabet Γ to encode cell contents: all cells represented by the range of s_a contain $a \in \Gamma$.

The most natural way of encoding a transition $\delta(q, a) = (q', a', \sigma)$ of M would be to use a concept inclusion of the form $H_q \sqcap \exists s_a^- \sqsubseteq \exists s_{a'} \sqcap \exists s_{q'\sigma}$, where $s_{q'\sigma}$ is also a sub-role of role next, which is functional in the grid. Unfortunately, *DL-Lite*$^{\mathcal{H}}_{\mathsf{core}}$ does not have conjunction on the left-hand side of concept inclusions. The following construction allows us to simulate the required inclusions by using functionality of just two roles, r and next. Let $\mathcal{T}_F$ contain (6.26), (6.27) and the following concept and role inclusions with fresh role names p_q, q_a and p_{qa}, for each $q \in Q \cup \{\emptyset\}$ and $a \in \Gamma$:

$$\begin{array}{ccc}
H_q \sqsubseteq D, & H_q \sqsubseteq \exists p_q, & \exists s_a^- \sqsubseteq \exists q_a, \\
 & p_q \sqsubseteq r, & q_a \sqsubseteq r, \\
\exists p_q^- \sqsubseteq E, & \exists p_q^- \sqsubseteq \exists p_{q_\sqcup}, & \exists p_q^- \sqsubseteq \exists p_{q1}, \\
 & p_{q_\sqcup} \sqsubseteq r, & p_{q1} \sqsubseteq \mathsf{next}, \\
 & q_\sqcup^- \sqsubseteq r, & q_1^- \sqsubseteq \mathsf{next}.
\end{array}$$

Claim 8. If $\mathcal{I} \models \mathcal{T}_G \cup \mathcal{T}_F$ and $\mathcal{I} \not\models \boldsymbol{q}$ then, for each $a \in \Gamma$ and $q \in Q \cup \{\emptyset\}$, we have

$$d \in (\exists p_{qa}^-)^{\mathcal{I}} \quad \text{whenever} \quad d \in H_q^{\mathcal{I}} \cap (\exists s_a^-)^{\mathcal{I}},$$

for every d with an $(\mathsf{next}^-)^{\mathcal{I}} \circ \mathsf{tape}^{\mathcal{I}} \circ \mathsf{next}^{\mathcal{I}}$-predecessor.

Proof of claim. Let $d \in H_q^{\mathcal{I}} \cap (\exists s_a^-)^{\mathcal{I}}$. Then d has a $p_q^{\mathcal{I}}$-successor and a $q_a^{\mathcal{I}}$-successor, which coincide because, by Claim 7, $r^{\mathcal{I}}$ is functional in $d \in D^{\mathcal{I}}$. Let d' be the $r^{\mathcal{I}}$-successor of d.

If $a = 1$ then the *inverse* of q_1 is a sub-role of next, and thus, $(d', d) \in \mathsf{next}^{\mathcal{I}}$. On the other hand, d' has a $p_{q1}^{\mathcal{I}}$-successor d'', whence $(d', d'') \in \mathsf{next}^{\mathcal{I}}$. Since $d' \in E^{\mathcal{I}}$, by Claim 6, $\mathsf{next}^{\mathcal{I}}$ is functional in d', whence $d = d''$. Therefore, $d \in (\exists p_{q1}^-)^{\mathcal{I}}$.

If $a = \sqcup$ then the argument is similar with r replacing next as a super-role of both $q_{\sqcup}^-$ and $p_{q\sqcup}$. By Claim 5, $r^{\mathcal{I}}$ is functional in any $r^{\mathcal{I}}$-predecessor of d, in particular in d'. Therefore, we obtain $d \in (\exists p_{q\sqcup}^-)^{\mathcal{I}}$. ❑

We are now in a position to define the encoding of Turing machine computations. Using the roles p_{qa} from $\mathcal{T}_F$, we can encode transitions:

$$\exists p_{qa}^- \sqsubseteq \exists s_{a'} \sqcap \exists s_{q'\sigma}, \qquad \text{for } \delta(q, a) = (q', a', \sigma), \tag{6.28}$$

$$s_a \sqsubseteq \mathsf{next}, \qquad \text{for } a \in \Gamma, \tag{6.29}$$

$$s_{q\sigma} \sqsubseteq \mathsf{next}, \qquad \text{for } q \in Q \text{ and } \sigma \in \{-1, +1\}, \tag{6.30}$$

where $s_{q,-1}$ and $s_{q,+1}$ are fresh role names used to propagate the new state to the next configuration. Recall now that the ranges of roles $p_{\emptyset a}$ identify cells that are not observed by the head of M; the symbols contained in such cells are then preserved with the help of concept inclusions

$$\exists p_{\emptyset a}^- \sqsubseteq \exists s_a, \qquad \text{for } a \in \Gamma. \tag{6.31}$$

The location of the head in the next configuration is ensured by the following inclusions:

$$\exists s_{q\sigma}^- \sqsubseteq \exists t_{q\sigma}, \qquad \text{for } q \in Q \text{ and } \sigma \in \{-1, +1\}, \tag{6.32}$$

$$\exists t_{q\sigma}^- \sqsubseteq H_q, \qquad \text{for } q \in Q \text{ and } \sigma \in \{-1, +1\}, \tag{6.33}$$

$$t_{q,+1} \sqsubseteq \mathsf{tape} \text{ and } t_{q,-1} \sqsubseteq \mathsf{tape}^-, \qquad \text{or } q \in Q, \tag{6.34}$$

where the roles $t_{q,+1}$ and $t_{q,-1}$ are used to propagate the head in the state q along the tape (both tape and tape^- are functional in the grid); finally, the following concept inclusions are required to propagate the no-head marker $H_\emptyset$:

$$H_q \sqsubseteq \exists t_{\emptyset,+1} \sqcap \exists t_{\emptyset,-1}, \qquad \text{for } q \in Q, \tag{6.35}$$

$$t_{\emptyset,+1} \sqsubseteq \mathsf{tape} \text{ and } t_{\emptyset,-1} \sqsubseteq \mathsf{tape}^-, \tag{6.36}$$

$$\exists t_{\emptyset\sigma}^- \sqsubseteq \exists t_{\emptyset\sigma} \sqcap H_\emptyset, \qquad \text{for } \sigma \in \{-1, +1\}. \tag{6.37}$$

Next, the ABox $\mathcal{A}_{\vec{w}}$ that encodes an input $\vec{w} = a_1, \dots, a_n \in \Gamma^*$ of M is as follows:

$$\begin{aligned} &\mathsf{bottom}(c_{00}, c_{10}), \qquad \mathsf{tape}(c_{10}, c_{11}), \qquad H_{q_0}(c_{11}), \\ &\qquad\qquad \mathsf{tape}(c_{0(i-1)}, c_{0i}) \text{ and } s_{a_i}(c_{0i}, c_{1i}), \quad \text{for } 1 \leq i \leq n, \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad t_0(c_{0n}, c_{0(n+1)}), \end{aligned}$$

where bottom is a fresh role name to create the bottom row of the grid and t_0 is a fresh role name to fill the rest of the tape by blanks:

$$\exists \mathsf{bottom}^- \sqsubseteq \exists \mathsf{bottom}, \qquad \mathsf{bottom} \sqsubseteq \mathsf{next}, \tag{6.38}$$

$$\exists t_0^- \sqsubseteq \exists \mathsf{next}_{\sqcup} \sqcap \exists t_0, \qquad t_0 \sqsubseteq \mathsf{tape}. \tag{6.39}$$

Finally, the following ensures that the accepting state q_1 never occurs in a computation:

$$H_{q_1} \sqsubseteq \bot. \tag{6.40}$$

Let $\mathcal{T}_M$ contain (6.28)–(6.40) encoding computations of M and let $\mathcal{T} = \mathcal{T}_G \cup \mathcal{T}_F \cup \mathcal{T}_M$. If $(\mathcal{T}, \mathcal{A}_{\vec{w}}) \not\models \boldsymbol{q}$ then there is a model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A}_{\vec{w}})$ with $\mathcal{I} \not\models \boldsymbol{q}$. It should then be clear that in this case we can extract a computation of M encoded by $\mathcal{I}$ and that computation does not accept $\vec{w}$. Conversely, if M does not accept $\vec{w}$ then we can construct a model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A}_{\vec{w}})$ such that $\mathcal{I} \not\models \boldsymbol{q}$. First, consider a model $\mathcal{J}$ of $\mathcal{T}_G$ with

$$\Delta^{\mathcal{J}} = \{\, d_{ij} \mid i, j \geq 0 \,\} \cup \{\, d'_{ij}, d''_{ij} \mid i > 0 \text{ and } j \geq 0 \,\} \cup \{\, b_i \mid i > 0 \,\}$$

such that the d_{ij} form a grid structure on roles next and tape, each d'_{ij} is an $r^{\mathcal{J}}$-predecessor of d_{ij} and each d''_{ij} is an $\mathsf{next}^{\mathcal{J}}$-predecessor of d_{ij} (note that d_{ij} has another $\mathsf{next}^{\mathcal{J}}$-predecessor, $d_{(i-1)j}$). Next, we choose the interpretation of concepts and roles in $\mathcal{T}_M$ on the domain of $\mathcal{J}$ in such a way that the part of $\mathcal{J}$ rooted in d_{11} encodes a unique computation of M on $\vec{w}$ and $\mathcal{J} \models (\mathcal{T}_M, \mathcal{A}_{\vec{w}})$. In particular, the computation determines the interpretation of H_q, s_a and $s_{q\sigma}$, for $q \in Q$, $a \in \Gamma$ and $\sigma \in \{-1, +1\}$. To interpret $H_\emptyset$ and $t_{q\sigma}$, for $q \in Q \cup \{\emptyset\}$ and $\sigma \in \{-1, +1\}$: the only non-trivial case is $t_{\emptyset,-1}$, where, in order to satisfy (6.35), we take b_i to be a $t^{\mathcal{J}}_{\emptyset,-1}$-successor (and so, a $\mathsf{tape}^{\mathcal{J}}$-predecessor) of both d_{i1} and b_i, for each $i > 0$ (as we noted, $(\mathsf{tape}^-)^{\mathcal{J}}$ does not have to be functional in any d_{i1}; $\mathsf{tape}^{\mathcal{J}}$, however, must be functional in each d_{i0} and cannot have a $\mathsf{tape}^{\mathcal{J}}$-loop). As the final step of the construction of $\mathcal{J}$, we set

$$\begin{aligned} &(d'_{ij}, d_{ij}) \in p^{\mathcal{J}}_{q_{\sqcup}} \text{ and } (d_{ij}, d'_{ij}) \in r^{\mathcal{J}} \quad \text{if} \quad d_{ij} \in H^{\mathcal{J}}_q \cap (\exists s^-_{\sqcup})^{\mathcal{J}}, \\ &(d''_{ij}, d_{ij}) \in p^{\mathcal{J}}_{q1} \text{ and } (d_{ij}, d''_{ij}) \in r^{\mathcal{J}} \quad \text{if} \quad d_{ij} \in H^{\mathcal{J}}_q \cap (\exists s^-_1)^{\mathcal{J}}. \end{aligned}$$

It remains to show that $\mathcal{J}$ can be extended to satisfy $\mathcal{T}_F$. Observe that only concept names H_q and role names r, next, tape, s_a and p_{qa}, for $q \in Q \cup \{\emptyset\}$ and $a \in \Gamma$, are shared between $\mathcal{T}_F$

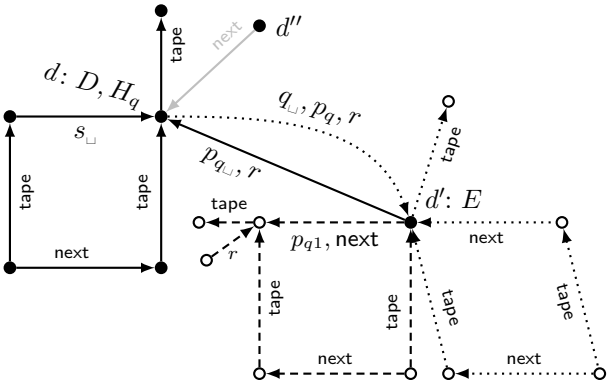


Figure 6.8: Extending $\mathcal{I}$ to $\mathcal{J}$: the case of $p_{q_\sqcup}$.

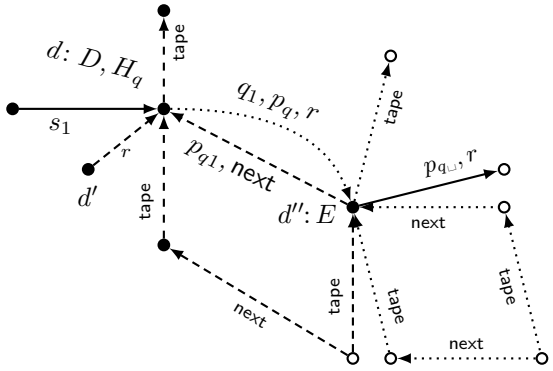


Figure 6.9: Extending $\mathcal{I}$ to $\mathcal{J}$: the case of p_{q1}.

and $\mathcal{T}_G \cup \mathcal{T}_M$; all other concept and roles names in $\mathcal{T}_F$ are *fresh in* $\mathcal{T}_F$. We show that $\mathcal{J}$ can be extended (by fresh domain elements) to a model of $\mathcal{T}_F$ without changing concepts and roles on grid, i.e., the domain elements of $\mathcal{J}$.

Claim 9. If $\mathcal{J} \models \mathcal{T}_G$ and $\mathcal{J} \not\models \boldsymbol{q}$ then $\mathcal{J}$ can be extended to a model $\mathcal{I}$ of $\mathcal{T}_F$ so that
(a) $d \in H_q^{\mathcal{I}} \cap (\exists s_a^-)^{\mathcal{I}}$ whenever $d \in (\exists p_{qa}^-)^{\mathcal{I}}$, for every $d \in \Delta^{\mathcal{J}}$ with an $(\mathsf{next}^-)^{\mathcal{J}} \circ \mathsf{tape}^{\mathcal{J}} \circ \mathsf{next}^{\mathcal{J}}$-predecessor, an $r^{\mathcal{J}}$-predecessor d' and another $\mathsf{next}^{\mathcal{J}}$-predecessor d'', and
(b) $A^{\mathcal{I}} \cap \Delta^{\mathcal{J}} = A^{\mathcal{J}}$ and $p^{\mathcal{I}} \cap (\Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}}) = p^{\mathcal{J}}$, for all concept names A and role names p that are *not fresh* in $\mathcal{T}_F$.

Proof of claim. The cases of $p_{q_\sqcup}$ and p_{q1} are illustrated in Figs. 6.8 and 6.9, respectively; some edges are not shown to avoid clutter: each domain element in $(\exists \mathsf{next}^-)^{\mathcal{I}}$ also has an incoming $r^{\mathcal{I}}$-edge and each $\mathsf{tape}^{\mathcal{I}}$-edge starts an infinite chain of $\mathsf{tape}^{\mathcal{I}}$-edges.
The three black (solid, dashed and dotted) patterns of edges in Fig. 6.8 correspond to the three sets of positive atoms of $\boldsymbol{q}$ so that the inequality atom, $(u' = v')$, 'identifies' certain domain elements of the pattern.

Similarly, the two black (dashed and dotted) patterns of edges in Fig. 6.9 correspond to the two sets of positive atoms of $\boldsymbol{q}$ that 'identify' certain domain elements.

Black nodes are in the domain of $\mathcal{J}$, while white nodes are in the domain of $\mathcal{I}$ proper. It can be seen that d is added only to D, and (d, d') or (d, d''), depending on the $(\exists s_a^-)^{\mathcal{J}}$, are added only to roles p_q and q_a (which are all fresh in $\mathcal{T}_F$). ❑

So, $(\mathcal{T}, \mathcal{A}_{\vec{w}}) \not\models \boldsymbol{q}$ iff M does not accept $\vec{w}$. Take M to be a fixed deterministic *universal* Turing machine, i.e., a machine that accepts $\vec{w}$ iff the empty input is accepted by the Turing machine encoded by $\vec{w}$. This finishes the proof of Theorem 6.12. ❑

6.2.2 Two Lower Bounds for CQs$^{\neq}$ Answering in *DL-Lite*$_{core}$

In the previous sections, we established undecidability of CQ$^{\neg s}$ and CQ$^{\neq}$ answering over *DL-Lite*$^{\mathcal{H}}_{core}$. However, all the proofs make use of role inclusions either explicitly in the TBox, or UCQs$^{\neq}$ and UCQ$^{\neg s}$. Leaving the problems of decidability of CQ$^{\neg s}$ and CQ$^{\neq}$ answering over *DL-Lite*$_{core}$ open. We establish lower complexity bounds for the second case, which make it clear that even if CQ$^{\neq}$ answering in *DL-Lite*$_{core}$ is decidable the problem is computationally hard.

Theorem 6.13. *There is a Boolean CQ$^{\neq}$ $\boldsymbol{q}$ with one inequality and a DL-Lite$_{core}$ TBox such that query answering is* P*-hard.*

Proof. The proof is by reduction of the complement of Horn-3SAT, the satisfiability problem for Horn clauses with at most 3 literals, which is known to be P-complete (see e.g., [107]). Suppose we are given a conjunction ψ of clauses of the form p, $\neg p$, and $p_1 \wedge p_2 \to p$. Fix a TBox $\mathcal{T}$ containing the following concept inclusions:

$$G \sqsubseteq \exists t, \qquad \exists t^- \sqsubseteq \exists t \sqcap V,$$

and a Boolean CQ$^{\neq}$ $\boldsymbol{q}$ which is the existential closure of the negation of the following:

$$V(x) \wedge s(x, y) \wedge r(y, z_1) \wedge t(y, z_2) \to (z_1 = z_2).$$

Note that $\mathcal{T}$ and $\boldsymbol{q}$ do not depend on ψ. Next, we construct an ABox $\mathcal{A}_\psi$ such that ψ is satisfiable iff $(\mathcal{T}, \mathcal{A}_\psi) \not\models \boldsymbol{q}$. The ABox $\mathcal{A}_\psi$ uses an individual name c_p, for each variable p in ψ, and individual names $c_{\gamma 1}$ and $c_{\gamma 2}$ for each clause γ of the form $p_1 \wedge p_2 \to p$ in ψ. For each clause γ, the ABox $\mathcal{A}_\psi$ contains the following assertions:

$$\begin{aligned}
V(c_p), &\quad \text{if } \gamma = p,\\
F(c_p), &\quad \text{if } \gamma = \neg p,\\
s(c_{p_1}, c_{\gamma 1}),\ G(c_{\gamma 1}),\ r(c_{\gamma 1}, c_{\gamma 2}),&\\
s(c_{p_2}, c_{\gamma 2}),\ r(c_{\gamma 2}, c_p), &\quad \text{if } \gamma = p_1 \wedge p_2 \to p.
\end{aligned}$$

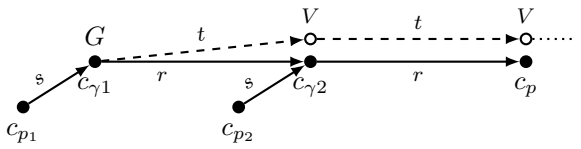


Figure 6.10: Proof of Theorem 6.13.

Suppose first there is a model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A}_\psi)$ with $\mathcal{I} \not\models \boldsymbol{q}$. We show that ψ is satisfiable. For each clause γ of ψ of the form $p_1 \wedge p_2 \to p$ (the other two cases are trivial), $\mathcal{I}$ contains a configuration depicted in Fig. 6.10 (the black nodes represent ABox individuals and the white ones—anonymous individuals generated by the TBox).

If $c^{\mathcal{I}}_{p_1} \in V^{\mathcal{I}}$ then the $t^{\mathcal{I}}$- and $r^{\mathcal{I}}$-successors of $c^{\mathcal{I}}_{\gamma 1}$ coincide, whence $c^{\mathcal{I}}_{\gamma 2} \in (\exists t)^{\mathcal{I}}$, which triggers the second 'application' of the query to identify $c^{\mathcal{I}}_p$ with the $t^{\mathcal{I}}$-successor of $c^{\mathcal{I}}_{\gamma 2}$ resulting in $c^{\mathcal{I}}_p \in V^{\mathcal{I}}$ but *only if* $c^{\mathcal{I}}_{p_2} \in V^{\mathcal{I}}$. So, as follows from the argument above, we can define a satisfying assignment $\mathfrak{a}$ for ψ by taking $\mathfrak{a}(p)$ true iff $c^{\mathcal{I}}_p \in V^{\mathcal{I}}$.

Conversely, if ψ is satisfiable then we can construct a model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A}_\psi)$ with $\mathcal{I} \not\models \boldsymbol{q}$. ❑

Theorem 6.14. *There is a Boolean CQ$^{\neq}$ $\boldsymbol{q}$ with two inequalities and a TBox such that query answering is* coNP*-hard.*

Proof. The proof is by reduction of the complement of 3SAT, which is known to be coNP-complete (see e.g., [107]). Suppose we are given a conjunction ψ of clauses of the form $\ell_1 \vee \ell_2 \vee \ell_3$, where the ℓ_k are literals (we can assume that all literals in each clause are distinct). Fix a TBox $\mathcal{T}$ containing the following concept inclusions:

$$V \sqsubseteq \exists t \sqcap \exists f, \qquad \exists t^- \sqsubseteq V, \qquad \exists t^- \sqcap \exists f^- \sqsubseteq \bot, \qquad A_1 \sqcap A_2 \sqsubseteq \bot,$$

and a Boolean CQ$^{\neq}$ $\boldsymbol{q}$ which is the existential closure of the negation of the following:

$$V(x) \wedge p(x, y) \wedge t(x, y_1) \wedge f(x, y_2) \ \to (y = y_1) \vee (y = y_2).$$

Claim 10. Let $\mathcal{I}$ be a model of $\mathcal{T}$ with $\mathcal{I} \not\models \boldsymbol{q}$. If $d \in V^{\mathcal{I}}$ and $(d, d_k) \in p^{\mathcal{I}}$ with $d_k \in A^{\mathcal{I}}_k$, for $k = 1, 2$, then

- either $(d, d_1) \in f^{\mathcal{I}}$ and $(d, d_2) \in t^{\mathcal{I}}$
- or $(d, d_1) \in t^{\mathcal{I}}$ and $(d, d_2) \in f^{\mathcal{I}}$.

Proof of claim. Clearly, each pair (d, d_k) belongs either to $t^{\mathcal{I}}$ or $f^{\mathcal{I}}$. Suppose to the contrary that $(d, d_k) \in t^{\mathcal{I}}$. Consider $\boldsymbol{q}$ with $x \mapsto d$, $y \mapsto d_1$, $y_1 \mapsto d_2$ and any $f^{\mathcal{I}}$-successor of d as y_2. By disjointness of the A_k, $d_1 \neq d_2$, and so, we can only choose $y = y_2$, whence $(d, d_1) \in f^{\mathcal{I}}$ contrary to disjointness of $\exists t^-$ and $\exists f^-$. ❑

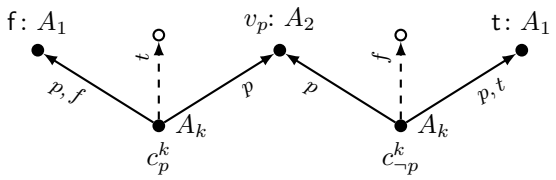


Figure 6.11: Proof of Theorem 6.14.

Again, $\mathcal{T}$ and $\boldsymbol{q}$ do not depend on ψ. The ABox $\mathcal{A}_\psi$ is constructed as follows. Let $\mathtt{t}$ and $\mathtt{t}$ be two individuals with $A_1(\mathtt{t})$ and $A_1(\mathtt{f})$ in $\mathcal{A}_\psi$. For each propositional variable p of ψ, take the following assertions, for $k = 1, 2$, with 5 individuals v_p, $c_{\neg p}^k$ and c_p^k:

$$A_2(v_p), \quad p(c_p^k, v_p), \quad p(c_p^k, \mathtt{f}), \quad f(c_p^k, \mathtt{f}), \quad A_k(c_p^k),$$
$$p(c_{\neg p}^k, v_p), \quad p(c_{\neg p}^k, \mathtt{t}), \quad t(c_{\neg p}^k, \mathtt{t}), \quad A_k(c_{\neg p}^k)$$

where the c_p^k and $c_{\neg p}^k$ represent the literals p and $\neg p$, respectively, see Fig. 6.11.

Observe that, by Claim 10, if $(c_{\neg p}^k)^{\mathcal{I}} \in V^{\mathcal{I}}$ in a model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A}_\psi)$ with $\mathcal{I} \not\models \boldsymbol{q}$ then $v_p^{\mathcal{I}} \in (\exists f^-)^{\mathcal{I}}$, that is, if the literal $\neg p$ is chosen (by means of V) then p must be false; on the other hand, if $\neg p$ is not chosen (that is, $(c_{\neg p}^k)^{\mathcal{I}} \notin V^{\mathcal{I}}$) then $v_p^{\mathcal{I}}$ does not have to be in $(\exists f^-)^{\mathcal{I}}$ and p can be anything; and similarly for $(c_p^k)^{\mathcal{I}}$ with $v_p^{\mathcal{I}} \in (\exists t^-)^{\mathcal{I}}$.

Next, $\mathcal{A}_\psi$ contains, for each clause γ of the form $\ell_1 \vee \ell_2 \vee \ell_3$ in ψ, the following assertions, where $c_{\gamma 1}$ and $c_{\gamma 2}$ are two fresh individuals:

$$V(c_{\gamma 1}), \qquad p(c_{\gamma 1}, c_{\ell_1}^1), \quad p(c_{\gamma 1}, c_{\gamma 2}), A_2(c_{\gamma 2}), \qquad p(c_{\gamma 2}, c_{\ell_2}^1), \quad p(c_{\gamma 2}, c_{\ell_3}^2).$$

It should be clear that ψ is satisfiable iff $(\mathcal{T}, \mathcal{A}_\psi) \not\models \boldsymbol{q}$. Indeed, if there is a model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A}_\psi)$ with $\mathcal{I} \not\models \boldsymbol{q}$ then, by Claim 10 and the observation above, we can construct a satisfying assignment $\mathfrak{a}$ for ψ by taking $\mathfrak{a}(p)$ true iff $v_p^{\mathcal{I}} \in V^{\mathcal{I}}$. The converse direction is straightforward. ❑

Related Work and Conclusions

The first investigation to consider inequalities in the OBDA framework is the work by [41] which shows that, opposite to answering CQs, answering CQs$^{\neq}$ over the very expressive DL $\mathcal{DLR}$ is undecidable. Later on, [116] showed undecidability of answering CQs with safe negation and inequalities over the fairly unexpressive DL $\mathcal{AL}$. As discussed above, Horn DLs were also considered by Rosati [116]. In the context of the Datalog$^{\pm}$ ontology languages allowing for equalities in the head of the rules have been considered. Notably, [30] investigate a restriction on the interaction of equalities (EGDs) with Datalog$^{\pm}$ constraints that guarantees decidability of the query-answering problem. Recently, extensions of Datalog$^{\pm}$ languages with non-monotonic negation using well-founded semantics for normal logic programs have been also investigated [67].

Adding to our results to those already know from the literature, we can conclude that attaining decidability of ontological query answering with negations is rather unfeasible. And

	DL-Lite$_{core}$	*DL-Lite*$_{core}^{\mathcal{H}}$	$\mathcal{EL}$	$\mathcal{ELI}_{\bot}$
UCQ$^{\neg s}$	undec. Cor. 6.4	undec.	undec. [116]	undec.
CQ$^{\neg s}$	coNP-hard	undec. Thm. 6.6	coNP-hard	undec. Thm. 6.2
1 guarded neg	PTime Thm. 6.10			
$\geq$ 2 guarded negs	coNP Thm. 6.10			
UCQ$^{\neq}$	undec.	undec.	undec.	undec.
$\geq$ 2-CQ$^{\neq}$	coNP-hard Thm. 6.14	undec.	undec. [82]	undec.
1-CQ$^{\neq}$	PTime-hard Thm. 6.13	undec. Thm. 6.12	undec. [82]	undec.

Table 6.1: Panorama of the complexity of query answering with negation

even for the decidable cases, the complexity of ontological query answering with some form of negation is high. We summarize the results in Table 6.1.

The complexity in the case of *DL-Lite*$_{core}$ ontologies remains open; and only 2 lower bounds we provided, which demonstrate that even for such unexpressive ontology languages the problem becomes intractable when the query has at least two inequalities (or two guarded negations). The major challenge to provide an algorithmic approach to that problem (and thus an upper bound for the complexity) is that since CQs with any form of negations are not preserved under homomorphisms, one cannot rely on techniques using the canonical model of the ontology.

Finally, observe that from the fact that (U)CQs$^{\neq}$ (in negated form) can express role functionality (e.g., consider the query $r(x, y_1), r(x, y_2) \rightarrow y_1 = y_2$), we can then conclude that (U)CQs$^{\neq}$ is not *finitely controllable* for ontology languages that contain inverse roles, and cyclic CIs. Indeed, there is a query q and an ontology $\mathcal{O}$ such that $\mathcal{O} \models q$ but $\mathcal{O} \not\models_{\mathsf{fin}} q$. Moreover, note that our undecidability proofs rely on the encoding of infinite structures (e.g., infinite grid), and therefore do not hold for the finite case.

Chapter

7 Reasoning in Conceptual Models

One of the prominent applications of description logics is in providing deductive capabilities to conceptual models such as ER or UML diagrams (see Section 1.3). Unfortunately, some of the semantic assumptions in conceptual modeling are not fully matched on the DLs side. For example, in database applications is normally assumed that the intended models (i.e., database instances) are finite. However, to faithfully capture some aspects of conceptual models we indeed need to use DLs lacking the finite model property; for example, to reason on the ER model one needs to use the expressive $\mathcal{ALCQI}$. Hence one actually should regard finite model reasoning in such DLs instead of the unrestricted one. It is also worth noticing that using $\mathcal{ALCQI}$ to describe the semantics of conceptual modeling languages allows to unify various specifications of conceptual models in different formalisms in a single specification in DLs. Another important assumption regarding the application of conceptual diagrams (e.g., in configuration management) is that of full satisfiability, that is, it is required the existence of an instantiation (model) of the diagram where all the classes have at least one instance. The notion of full satisfiability clearly does not coincide with the standard notion of satisfiability in DLs and therefore it needs to be explicitly considered.

In this Chapter, we take a closer look at the use of description logics to reason on conceptual models. We specifically show how to reason under some of the semantic assumptions described above. Interestingly, we discuss how to apply to reasoning in conceptual models some of the results on finite reasoning obtained in previous chapters. In the remaining of this Chapter we will investigate the complexity of reasoning in CDs under the finite model assumption as well as the complexity of deciding full satisfiability of CDs. Table 7.1 summarizes the results.

7.1 Complexity of Reasoning in CDs

Recall that $\mathcal{ALCQI}$ lacks the finite model property due to the presence of cyclic inclusions, inverse roles and functionality, and that the translation of CDs into $\mathcal{ALCQI}$ TBoxes makes

Class Diagrams	Finite Satisfiability	Full Satisfiability
$\mathcal{D}_{full}$	ExpTime Thm. 7.2	ExpTime Thm. 7.9
$\mathcal{D}_{bool}$	in ExpTime	NP Thm. 7.12
$\mathcal{D}^{-}_{Horn}$	ExpTime Thm. 7.4	ExpTime
lite–$\mathcal{D}^{-}_{Horn}$	PTime	PTime
$\mathcal{D}_{ref}$	in ExpTime	NLogSpace Thm. 7.14
$\mathcal{D}^{-}_{ref}$	NLogSpace Thm 7.3	PTime

Table 7.1: Complexity of reasoning in CDs

use of both inverse roles and number restrictions to capture the semantics of $\mathcal{D}_{full}$ diagrams (see Table 7.2). This means that for reducing reasoning in $\mathcal{D}_{full}$ to finite model reasoning in $\mathcal{ALCQI}$ one needs to define a mapping between finite instantiations corresponding to a CD $\mathcal{D}$ and finite interpretations of the TBox $\mathcal{T}_{\mathcal{D}}$ derived from $\mathcal{D}$. However, due to the possible presence of relations with arity greater than two this mapping is not one-one. Recall that n-ary relations are encoded in DLs by *reification*, using a concept name to represent the set of 'tuples' in the relation, and using one role to connect each component of the relation to the corresponding 'tuple'. For class diagrams, the extension of a relation in an instantiation is a set of tuples. Therefore, it is implicit that there cannot be two tuples connected through all components of the relation to exactly the same elements in the domain. But this is not a condition that can be enforced in the TBox $\mathcal{T}_{\mathcal{D}}$. More precisely, one would need to ensure that $\mathcal{T}_{\mathcal{D}}$ is such that there is exactly one instance of the concept A_R representing exactly one tuple in the relation R. Nevertheless, by the model theoretical properties of $\mathcal{ALCQI}$ it can be shown that this condition can be assumed [35], and reasoning in CDs can be reduced to finite model reasoning in $\mathcal{ALCQI}$.

Theorem 7.1. *[35] Let $\mathcal{D}$ be a $\mathcal{D}_{full}$ diagram, C_1, C_2 be two classes in $\mathcal{D}$, and $\mathcal{T}_{\mathcal{D}}$ be the $\mathcal{ALCQI}$ TBox encoding $\mathcal{D}$. Restricting to finite models the following holds:*

1. *C_1 is satisfiable in $\mathcal{D}$ if and only if $\mathcal{T}_{\mathcal{D}} \not\models_{\mathsf{fin}} A_{C_1} \sqsubseteq \bot$.*
2. *C_1 isa C_2 holds in $\mathcal{D}$ if and only if $\mathcal{T}_{\mathcal{D}} \models_{\mathsf{fin}} A_{C_1} \sqsubseteq A_{C_2}$.*

The previous results states that given a diagram $\mathcal{D}$, deciding subsumption between two classes C_1, C_2 amounts to decide finite subsumption between the concepts formalizing C_1 and C_2, i.e, whether for the DL concepts A_{C_1}, A_{C_2}, in every finite model of $\mathcal{T}_{\mathcal{D}}$ every instance of A_{C_1} is also an instance of A_{C_2}. Analogously, deciding the satisfiability of a class C_1 amounts to verifying the existence of a finite model of $\mathcal{T}_{\mathcal{D}}$ where the set of instances of A_{C_1} is not empty, i.e, A_{C_1} is not finitely subsumed by $\bot$ w.r.t. $\mathcal{T}_{\mathcal{D}}$.

On the other hand, satisfiability of a diagram $\mathcal{D}$ does not correspond directly to *finite satisfiability* of the TBox $\mathcal{T}_{\mathcal{D}}$. Indeed, due to the requirement of the 'witnessing' instantiation

CD construct	$\mathcal{ALCQI}$ **axioms**
isa between classes C_1 and C_2	$A_{C_1} \sqsubseteq A_{C_2}$
isa between n-ary relations R_1 and R_2	$A_{R_1} \sqsubseteq A_{R_2}$
Relation R with C as the component $R[i]$	$A_R \sqsubseteq \forall p_{R[i]}.A_C \sqcap \exists p_{R[i]}.A_C \sqcap (\leqslant 1\ p_{R[i]}\ A_C)$
Cardinality constraint $(m..n)$ on C as $R[i]$	$A_C \sqsubseteq (\geqslant m\ p^-_{R[i]}\ A_R) \sqcap (\leqslant n\ p^-_{R[i]}\ A_R)$

Table 7.2: $\mathcal{ALCQI}$ axioms derived from an $\mathcal{D}_{full}$ CD

not to be the empty one, i.e., the trivial (finite) instantiation where the set of instances of each class C in $\mathcal{D}$ is empty. Instead, satisfiability of $\mathcal{D}$ amounts to verifying, e.g., whether the ontology

$$(\mathcal{T}_\mathcal{D}, \{A_C(a)\})$$

is finitely satisfiable, for some class C encoded by A_C. Note that in the latter the ABox $\mathcal{A} = \{A_C(a)\}$ forces that in every (finite) model of $\mathcal{T}_\mathcal{D}$ satisfying $\mathcal{A}$, the concept A_C has at least one instance. Analogously, deciding whether there is an instantiation of $\mathcal{D}$ where the set of instances of each class is non-empty could be reduced to the problem of deciding whether $(\mathcal{T}_\mathcal{D}, \mathcal{A}')$ is satisfiable, where $\mathcal{A}'$ is the set of assertions:

$$\{\ A_{C_1}(a_1), \ldots A_{C_n}(a_n)\ \},$$

where $C_1, \ldots, C_n$ are all the classes occurring in $\mathcal{D}$. As we show in the following, it is also possible to reduce full satisfiability –which additionally requires all the relations to be nonempty– to (finite model) reasoning in $\mathcal{ALCQI}$. Observe that verifying full satisfiability is of importance to detect the presence of an unsatisfiable class or relation. The latter means that either the diagram contains unnecessary information that should be removed, or there is some modeling error.

The reduction provided by Theorem 7.1 and the complexity of finite model reasoning in $\mathcal{ALCQI}$ [94] imply that deciding (class) satisfiability and subsumption in $\mathcal{D}_{full}$ can be done in ExpTime on the size of $\mathcal{D}$. Moreover, this complexity bound is tight, as shown in [21], reasoning in $\mathcal{ALC}$ can be reduced to reasoning in a fragment of UML class diagrams that correspond to our $\mathcal{D}_{full}$ diagrams. Hence the following complexity result is implied.

Theorem 7.2. *Deciding subsumption, and (class) satisfiability of $\mathcal{D}_{full}$ diagrams w.r.t. finite models is* ExpTime*-complete.*

The previous result indicates that for rather expressive CDs the complexity of reasoning is high. However, it has been shown that for the restricted $\mathcal{D}_{bool}$ and $\mathcal{D}_{ref}$ diagrams the complexity of verifying class subsumption and satisfiability decreases in comparison to $\mathcal{D}_{full}$. Verifying

CD construct	$\mathcal{ALCQI}$ **axioms**
isa between classes C_1 and C_2	$A_{C_1} \sqsubseteq A_{C_2}$
disjoint constraint among $C_1 \ldots, C_n$	$A_{C_i} \sqcap A_{C_j} \sqsubseteq \bot, \quad 1 \leq i < j \leq n$
complete constraint on C isa $\{C_1, \ldots, C_n\}$	$C_i \sqsubseteq C,$ $C \sqsubseteq C_1 \sqcup \cdots \sqcup C_n$
Relation R with C as the component $R[i]$	$A_R \sqsubseteq \exists p_{R[i]}, \quad (\geqslant 2\, p_{R[i]}) \sqsubseteq \bot,$ $\exists p_{R[i]} \sqsubseteq A_R,$ $\exists p^-_{R[i]} \sqsubseteq A_C.$
Cardinality constraint $[m..n]$ on C as $R[i]$	$A_C \sqsubseteq (\geqslant m\, p^-_{R[i]})$ $A_C \sqsubseteq (\leqslant n\, p^-_{R[i]})$

Table 7.3: *DL-Lite*$^{\mathcal{N}}_{Bool}$ axioms derived from an $\mathcal{D}_{bool}$ CD

subsumption in $\mathcal{D}_{bool}$ CDs in NP-complete; and for $\mathcal{D}_{ref}$ diagrams is NLogSpace-complete. Artale et al. [8] provide the upper complexity bound using the following reduction:

- every $\mathcal{D}_{bool}$ diagram can be encoded into a *DL-Lite*$^{\mathcal{N}}_{Bool}$ TBox $\mathcal{T}$, in which every class and relation is encoded by an atomic concept. The encoding is such that a class (relation) C is satisfiable if and only if the corresponding concept A_C (A_R) is satisfiable w.r.t. $\mathcal{T}$;
- similarly, every $\mathcal{D}_{ref}$ diagram can be encoded into a *DL-Lite*$^{\mathcal{N}}_{core}$ TBox.

In Table 7.3, we present the encoding of $\mathcal{D}_{bool}$ diagrams into *DL-Lite*$^{\mathcal{N}}_{Bool}$. Observe that for diagrams in $\mathcal{D}_{ref}$, i.e., those without complete constraints on class hierarchies, the encoding according to Table 7.3 corresponds to a *DL-Lite*$^{\mathcal{N}}_{core}$ TBox. The matching lower complexity bounds are proven in [8] by direct reduction to well-known problems complete for the corresponding complexity classes. However, the complexity results apply only for the unrestricted model case. Nonetheless, using the reduction and the results of complexity of finite model reasoning in *DL-Lite*$^{\mathcal{F}}_{core}$ the following can be shown.

Theorem 7.3. *Deciding (finite) class subsumption and (class) satisfiability for* $\mathcal{D}^-_{ref}$ *diagrams is* NLogSpace*-complete.*

Observe that for the case of $\mathcal{D}^-_{bool}$ the best known complexity upper bound for finite class subsumption and (class) satisfiability is ExpTime, that is, the same as for $\mathcal{D}_{full}$. Indeed, although satisfiability in *DL-Lite*$^{\mathcal{N}}_{Bool}$ in NP-complete, this logic lacks the finite model property.

Our results on finite model reasoning in these logics (see Chapter 4) imply the following result:

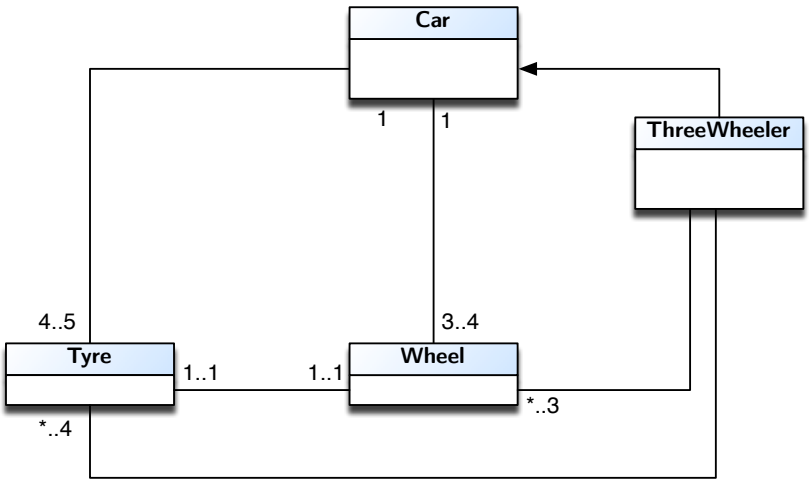
Theorem 7.4. *Deciding finite subsumption and (class) satisfiability for* $\mathcal{D}^-_{Horn}$ *is* ExpTime*-complete. Further, it is* PTime*-complete for lite-*$\mathcal{D}^-_{Horn}$ *diagrams.*

Although the complexity of finite model reasoning in Horn-$\mathcal{ALCQI}$ is high (ExpTime-complete), our results from Chapter 4 show a promising way to realize an implementation for a finite model reasoner via the calculus from Section 4.1. Such a reasoner would provide as well an implementation for reasoning in $\mathcal{D}_{Horn}$ diagrams.

7.2 Full Satisfiability of CDs

As discussed above, full satisfiability of a class diagram amounts to verifying whether there is an instantiation of the diagram in which *all* the classes (and relations) have at least one instance. Thus, a class diagram is not fully satisfiable even if it admits an instantiation satisfying all constraints, but where a particular class can never be populated (due to over-restrictive constraints). The rationale is that such a 'permanently empty' class constitutes a specification error and hence should be reported to the user. Full satisfiability of CDs that support cardinality constraints is relevant for example in *configuration management*. A *configuration* is an arrangement of functional units according to their nature, number and chief characteristics [26]. Functional units may be a software or hardware component (e.g, electronic circuits, or parts of a machine). In configuration management, is paramount to specify admissible arrangements in a natural way, to set up according to certain criteria of optimality and to maintain them when requirements change. In this context, a CD is a *specification* describing the component types, their properties and interrelations; and the collection of concrete instances together with their relations forms a *configuration*.

Example 12. Consider the following diagram specifying that *Cars* come with *three* or *four wheels* each requiring a *tire*; and that *Cars* have to be delivered with *four* or *five tires* (including the spare tire).



Observe that according to this specification in every proper instantiation of the diagram the class ThreeWheeler is empty. Indeed, every instantiation admits only four wheeled cars without

a spare tire, since the *at-least* constraint 3 and the *at-most* constraint 5 cannot be satisfied in proper instances (e.g., in finite instances). Thus, the diagram is not fully satisfiable.

For this small example, it can be seen that the problem rises from mixing two incompatible view on tires: those delivered with the car versus those attached to wheels. Such mistakes however are natural in specifications developed by several people, and more difficult to detect in complex diagrams. $\overline{\wedge}$

In this section, we will show results on the computational complexity of deciding (finite) full satisfiability of CDs. More precisely, we show that deciding full satisfiability is

- ExpTime-complete for $\mathcal{D}_{full}$ diagrams;
- NP-complete for $\mathcal{D}_{bool}$ diagrams w.r.t. possibly infinite instantiations and
- NLogSpace-complete for $\mathcal{D}_{ref}$.

These results build on the formalization of CDs in terms of DLs (Tables 7.2 and 7.3). The upper bounds are derived from reducing full satisfiability to class satisfiability on CDs—or equivalently to (concept) satisfiability on DL ontologies. Showing that the complexity bound obtained from such reduction are tight requires slightly more involved proofs.

Full Satisfiability of TBoxes

We start by proving results on the complexity of *full satisfiability* of TBoxes, and we then transfer these results to CDs. We first define that notion for TBoxes.

Definition 7.1 (TBox Full Satisfiability)**.** Let $\mathcal{L}$ be a description language. An $\mathcal{L}$ TBox $\mathcal{T}$ is *fully satisfiable* iff there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $A^{\mathcal{I}} \neq \emptyset$, for every atomic concept $A \in \mathsf{CN}(\mathcal{T})$. We say that $\mathcal{I}$ is a *full model* of $\mathcal{T}$. $\triangle$

We provide the following lower bound on the complexity of deciding full satisfiability of $\mathcal{ALC}$ TBoxes is ExpTime-hard.

Lemma 7.5. *Concept satisfiability w.r.t. $\mathcal{ALC}$ TBoxes can be linearly reduced to full satisfiability of $\mathcal{ALC}$ TBoxes.*

Proof. Let $\mathcal{T}$ be an $\mathcal{ALC}$ TBox and C an $\mathcal{ALC}$ concept. A well known result (see e.g., [27]) is that, C is satisfiable w.r.t. $\mathcal{T}$ if and only if $C \sqcap A_{\mathcal{T}}$ is satisfiable w.r.t. the TBox $\mathcal{T}_1$ consisting of the single assertion,

$$A_{\mathcal{T}} \sqsubseteq \bigsqcap_{C_1 \sqsubseteq C_2 \in \mathcal{T}} (\neg C_1 \sqcup C_2) \sqcap \bigsqcap_{1 \leq i \leq n} \forall p_i . A_{\mathcal{T}},$$

where $A_{\mathcal{T}}$ is a fresh atomic concept and $p_1, \ldots, p_n$ are all the role names occurring in $\mathcal{T}$ and C. In order to reduce the latter problem to full satisfiability, we extend $\mathcal{T}_1$ to $\mathcal{T}_2 = \mathcal{T}_1 \cup \{A_C \sqsubseteq C \sqcap A_{\mathcal{T}}\}$, with A_C a fresh atomic concept. We will now show that:

(†) $C \sqcap A_{\mathcal{T}}$ is satisfiable w.r.t. $\mathcal{T}_1$ if and only if $\mathcal{T}_2$ is fully satisfiable.

For the '$\Rightarrow$' direction of (†). Let $\mathcal{I}$ be a model of $\mathcal{T}_1$ such that $(C \sqcap A_{\mathcal{T}})^{\mathcal{I}} \neq \emptyset$. We construct an interpretation of $\mathcal{T}_2$, $\mathcal{J} = (\Delta^{\mathcal{I}} \cup \{d^{top}\}, \cdot^{\mathcal{J}})$, with $d^{top} \notin \Delta^{\mathcal{I}}$, such that:

$$\begin{array}{rcl} A_{\mathcal{T}}^{\mathcal{J}} & = & A_{\mathcal{T}}^{\mathcal{I}}, \qquad A_C^{\mathcal{J}} = (C \sqcap A_{\mathcal{T}})^{\mathcal{I}}, \\ A^{\mathcal{J}} & = & A^{\mathcal{I}} \cup \{d^{top}\}, \quad \text{for each concept name } A \text{ in } \mathcal{T} \text{ and } C, \\ p^{\mathcal{J}} & = & p^{\mathcal{I}}, \quad \text{for each role name } p \text{ in } \mathcal{T} \text{ and } C. \end{array}$$

Clearly, the extension of every atomic concept is non-empty in $\mathcal{J}$. Next, we show that $\mathcal{J}$ is a model of $\mathcal{T}_2$, by relying on the fact (easily proved by structural induction) that $D^{\mathcal{I}} \subseteq D^{\mathcal{J}}$, for each subconcept D of concepts in $\mathcal{T}_1$ or of C. Then, it is easy to show that $\mathcal{J}$ satisfies the two assertions in $\mathcal{T}_2$.

The '$\Leftarrow$' direction follows from the observation that every *full model* $\mathcal{J}$ of $\mathcal{T}_2$ is also a model of $\mathcal{T}_1$ with $(C \sqcap A_{\mathcal{T}})^{\mathcal{J}} \neq \emptyset$, as $A_C^{\mathcal{J}} \subseteq (C \sqcap A_{\mathcal{T}})^{\mathcal{J}}$.

❑

We can now prove the following easily.

Theorem 7.6. *Full satisfiability of $\mathcal{ALC}$ TBoxes is* ExpTime*-complete.*

Proof. The ExpTime membership is straightforward since full satisfiability of an $\mathcal{ALC}$ TBox $\mathcal{T}$ can be reduced to satisfiability of the TBox $\mathcal{T} \cup \bigcup_{1 \leq i \leq n} \{\top \sqsubseteq \exists p'.A_i\}$, where $A_1, \ldots, A_n$ are all the atomic concepts in $\mathcal{T}$, and p' is a fresh role name. The ExpTime-hardness follows from Lemma 7.5.

❑

Next, we show that the same result holds for syntactically simpler $\mathcal{ALC}$ TBoxes. A *primitive* $\mathcal{ALC}$ TBox, is an $\mathcal{ALC}$ TBox that contains only assertions of the form:

$$A \sqsubseteq B, \qquad A \sqsubseteq \neg B, \qquad A \sqsubseteq B \sqcup B', \qquad A \sqsubseteq \forall p.B, \qquad A \sqsubseteq \exists p.B,$$

where A, B, B' are concept names, and p is a role name. We now modify the reduction of Lemma 7.5 so that it applies also to primitive $\mathcal{ALC}$ TBoxes.

Theorem 7.7. *Full satisfiability of primitive $\mathcal{ALC}$ TBoxes is* ExpTime*-complete.*

Proof. The ExpTime membership follows from Theorem 7.6. For proving the ExpTime-hardness, we use a result in [21] showing that concept satisfiability in $\mathcal{ALC}$ can be reduced to atomic concept satisfiability w.r.t. primitive $\mathcal{ALC}$ TBoxes. Let $\mathcal{T} = \{A_j \sqsubseteq D_j \mid 1 \leq j \leq m\}$ be a primitive $\mathcal{ALC}$ TBox, and A_0 an atomic concept. By the proof of Lemma 7.5, we have that A_0 is satisfiable w.r.t. $\mathcal{T}$ if and only if the TBox $\mathcal{T}_2'$ consisting of the assertions

$$A_{\mathcal{T}^-} \sqsubseteq \prod_{A_j \sqsubseteq D_j \in \mathcal{T}} (\neg A_j \sqcup D_j) \sqcap \prod_{1 \leq i \leq n} \forall p_i . A_{\mathcal{T}}, \tag{7.1}$$

$$A_0' \sqsubseteq A_0 \sqcap A_{\mathcal{T}}\,, \tag{7.2}$$

is fully satisfiable, with $A_{\mathcal{T}}$, A_0' fresh atomic concepts.

Although $\mathcal{T}_2'$ is not a primitive $\mathcal{ALC}$ TBox, it is indeed equivalent to the TBox containing the assertions:

$$\begin{array}{lcl@{\qquad}lcl@{\qquad}lcl}
A_0' & \sqsubseteq & A_{\mathcal{T}} & A_{\mathcal{T}} & \sqsubseteq & \neg A_1 \sqcup D_1 & A_{\mathcal{T}} & \sqsubseteq & \forall p_1 . A_{\mathcal{T}} \\
 & & & & \vdots & & & \vdots & \\
A_0' & \sqsubseteq & A_0 & A_{\mathcal{T}} & \sqsubseteq & \neg A_m \sqcup D_m & A_{\mathcal{T}} & \sqsubseteq & \forall p_n . A_{\mathcal{T}},
\end{array}$$

Finally, to get a primitive $\mathcal{ALC}$ TBox, $\mathcal{T}_2$, we replace each assertion of the form $A_{\mathcal{T}} \sqsubseteq \neg A_j \sqcup D_j$ by

$$A_{\mathcal{T}} \sqsubseteq B_j^1 \sqcup B_j^2, \qquad B_j^1 \sqsubseteq \neg A_j,$$

and $B_j^2 \sqsubseteq D_j$, with B_j^1 and B_j^2 fresh atomic concepts, for $j \in \{1, \ldots, m\}$.

We show now that $\mathcal{T}_2'$ is fully satisfiable iff $\mathcal{T}_2$ is fully satisfiable:

($\Rightarrow$) Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a full model of $\mathcal{T}_2'$. We extend $\mathcal{I}$ to a model $\mathcal{J}$ of $\mathcal{T}_2$. Let $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}} \cup \{d^+, d^-\}$, with $\{d^+, d^-\} \cap \Delta^{\mathcal{I}} = \emptyset$, and define $(\cdot^{\mathcal{J}})$ as follows:

$$\begin{array}{rcl}
A_{\mathcal{T}^-}^{\mathcal{J}} & = & A_{\mathcal{T}}^{\mathcal{I}}, \quad {A_0'}^{\mathcal{J}} = {A_0'}^{\mathcal{I}}, \\
A^{\mathcal{J}} & = & A^{\mathcal{I}} \cup \{d^+\}, \quad \text{for every other atomic concept } A \text{ in } \mathcal{T}_2', \\
{B_j^1}^{\mathcal{J}} & = & (\neg A_j)^{\mathcal{J}} \text{ and } {B_j^2}^{\mathcal{J}} = D_j^{\mathcal{J}}, \quad \text{for each } A_{\mathcal{T}} \sqsubseteq B_j^1 \sqcup B_j^2 \in \mathcal{T}_2, \\
p^{\mathcal{J}} & = & p^{\mathcal{I}} \cup \{(d^+, d^+)\}, \quad \text{for each role name } p \text{ in } \mathcal{T}_2.
\end{array}$$

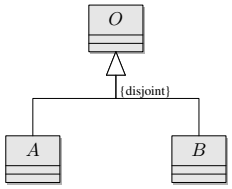
It is easy to see that $\mathcal{J}$ is a full model of $\mathcal{T}_2$.

($\Leftarrow$) Trivial since every model of $\mathcal{T}_2$ is a model of $\mathcal{T}_2'$.

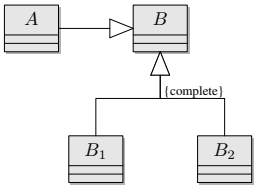
❑

Full Satisfiability of CDs

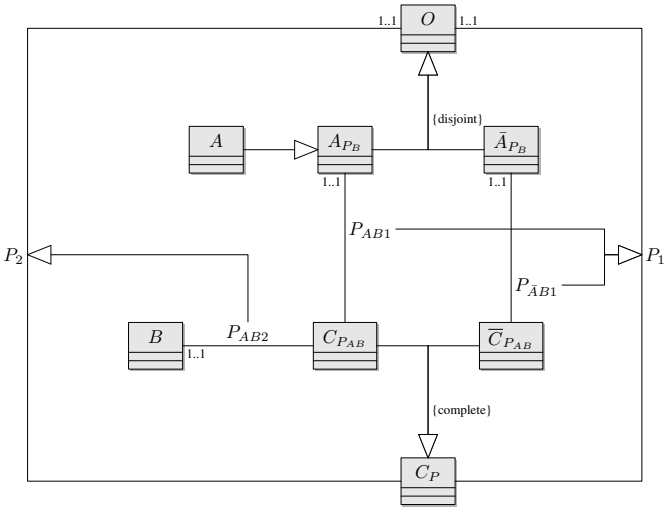
We now address the complexity of full satisfiability of $\mathcal{D}_{full}$. We reduce full satisfiability of primitive $\mathcal{ALC}$ TBoxes to full satisfiability of $\mathcal{D}_{full}$.



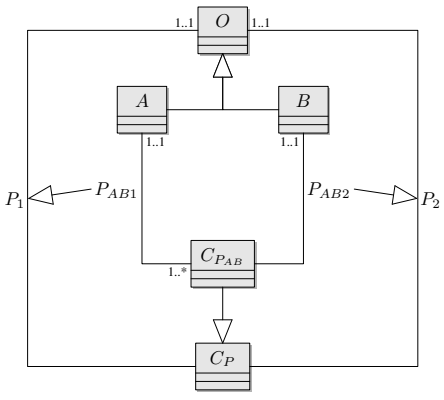
(a) Encoding of $A \sqsubseteq \neg B$



(b) Encoding of $A \sqsubseteq B_1 \sqcup B_2$



(c) Encoding of $A \sqsubseteq \forall p.B$



(d) Encoding of $A \sqsubseteq \exists p.B$

Figure 7.1: Reducing $\mathcal{ALC}$ TBox full satisfiability to $\mathcal{D}_{full}$

Given a primitive $\mathcal{ALC}$ TBox $\mathcal{T}$, construct an CD $\Sigma(\mathcal{T})$ as follows: for each atomic concept A in $\mathcal{T}$, introduce a class A in $\Sigma(\mathcal{T})$. Additionally, introduce a class O that generalizes (possibly indirectly) all the classes in $\Sigma(\mathcal{T})$ that encode an atomic concept in $\mathcal{T}$. For each role name $p \in \mathsf{role}(\mathcal{T})$, introduce a class C_P, which reifies p. Further, introduce two functional relations P_1, and P_2 that represent, respectively, the first and second component of p. The assertions in $\mathcal{T}$ are encoded as follows:

- For each assertion of the form $A \sqsubseteq B$, introduce a generalization between the classes A and B.
- For each assertion of the form $A \sqsubseteq \neg B$, construct the hierarchy in Fig. 7.1a.
- For each assertion of the form $A \sqsubseteq B_1 \sqcup B_2$, introduce an *auxiliary* class B, and construct the diagram shown in in Fig. 7.1b.
- For each assertion of the form $A \sqsubseteq \forall p.B$, add the auxiliary classes $C_{P_{AB}}$, $\overline{C}_{P_{AB}}$, A_{P_B}, and $\overline{A}_{P_B}$, and the relations P_{AB1}, $P_{\overline{A}B1}$, and P_{AB2}, and construct the diagram shown in Fig. 7.1c.
- For each assertion of the form $A \sqsubseteq \exists p.B$, add the auxiliary class $C_{P_{AB}}$ and the relations P_{AB1} and P_{AB2}, and construct the diagram shown in Fig. 7.1d.

Lemma 7.8. *A primitive $\mathcal{ALC}$ TBox $\mathcal{T}$ is fully satisfiable if and only if the $\mathcal{D}_{full}$ diagram $\Sigma(\mathcal{T})$, constructed as above, is fully satisfiable.*

Proof.
"$\Leftarrow$" Let $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ be a full model of $\Sigma(\mathcal{T})$. We construct a full model $\mathcal{I} = (\Delta^{\mathcal{I}}.^{\mathcal{I}})$ of $\mathcal{T}$ by taking $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$. Further, for every concept name A and for every role name p in $\mathcal{T}$, we define respectively $A^{\mathcal{I}} = A^{\mathcal{J}}$ and $p^{\mathcal{I}} = (P_1^-)^{\mathcal{J}} \circ P_2^{\mathcal{J}}$ ($R_1 \circ R_2$ denotes the composition of two binary relations R_1 and R_2). Let us show that $\mathcal{I}$ satisfies every assertion in $\mathcal{T}$. We distinguish the following cases:

- For assertions of the form $A \sqsubseteq B$, $A \sqsubseteq \neg B$, and $A \sqsubseteq B_1 \sqcup B_2$, the statement easily follows from the construction of $\mathcal{I}$.
- For assertions of the form $A \sqsubseteq \forall p.B$. Let $d \in A^{\mathcal{I}} = A^{\mathcal{J}}$ and $d' \in \Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$, such that $(d, d') \in r^{\mathcal{I}}$. Since $p^{\mathcal{I}} = (P_1^-)^{\mathcal{J}} \circ P_2^{\mathcal{J}}$, there is $d'' \in \Delta^{\mathcal{J}}$ such that $(d, d'') \in (P_1^-)^{\mathcal{J}}$, and $(d'', d') \in P_2^{\mathcal{J}}$. Then, $d'' \in C_p^{\mathcal{J}}$, and by the completeness constraint, $d'' \in C_{P_{AB}}^{\mathcal{J}} \cup \overline{C}_{P_{AB}}^{\mathcal{J}}$. We claim that $d'' \in C_{P_{AB}}^{\mathcal{J}}$. Suppose otherwise, then there is a unique $d_1 \in \Delta^{\mathcal{J}}$, such that $(d'', d_1) \in P_{\overline{A}B1}^{\mathcal{J}}$ and $d_1 \in \overline{A}_{P_B}^{\mathcal{J}}$. It follows from $P_{\overline{A}B1}^{\mathcal{J}} \subseteq P_1^{\mathcal{J}}$ and by the multiplicity constraint over C_P, that $d_1 = d$. This gives rise to a contradiction,because $d \in A^{\mathcal{J}} \subseteq A_{P_B}^{\mathcal{J}}$ and, $A_{P_B}^{\mathcal{J}}$ and $\overline{A}_{P_B}^{\mathcal{J}}$ are disjoint. Then $d'' \in C_{P_{AB}}^{\mathcal{J}}$. Further, there is a unique $d_2 \in \Delta^{\mathcal{J}}$ with $(d'', d_2) \in P_{AB2}^{\mathcal{J}}$ and $d_2 \in B^{\mathcal{J}}$. From $P_{AB2}^{\mathcal{J}} \subseteq P_2^{\mathcal{J}}$ and the multiplicity constraint on C_P, it follows that $d_2 = d'$. Thus, we have that $d' \in B^{\mathcal{J}} = B^{\mathcal{I}}$, and therefore, $d \in (\forall p.B)^{\mathcal{I}}$.

- For each assertion of the form $A \sqsubseteq \exists p.B$. Let $d \in A^{\mathcal{I}} = A^{\mathcal{J}}$. Then, there is $d' \in \Delta^{\mathcal{J}}$ such that $(d', d) \in P^{\mathcal{J}}_{AB1}$ and $d' \in C^{\mathcal{J}}_{P_{AB}}$. Since $d' \in C^{\mathcal{J}}_{P_{AB}}$, there is $d'' \in \Delta^{\mathcal{J}}$ with $(d', d'') \in P^{\mathcal{J}}_{AB2}$ and $d'' \in B^{\mathcal{J}} = B^{\mathcal{I}}$. Then we can conclude that $(d, d'') \in p^{\mathcal{I}}$ since $P^{\mathcal{J}}_{AB2} \subseteq P^{\mathcal{J}}_2$, $P^{\mathcal{J}}_{AB1} \subseteq P^{\mathcal{J}}_1$ and $P^{\mathcal{I}} = (P^{-}_1)^{\mathcal{J}} \circ P^{\mathcal{J}}_2$. Therefore, $d \in (\exists p.B)^{\mathcal{I}}$ as required.

"$\Rightarrow$" Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a full model of $\mathcal{T}$. We extend $\mathcal{I}$ to an instantiation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ of $\Sigma(\mathcal{T})$, by assigning suitable extensions to the auxiliary classes and relations in $\Sigma(\mathcal{T})$. Let $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}} \cup \Gamma \cup \Lambda$, where:

$$\Lambda = \biguplus_{A \sqsubseteq \forall p.B \in \mathcal{T}} \{a_{A_{P_B}}, a_{\overline{A}_{P_B}}\},$$

such that $\Delta^{\mathcal{I}} \cap \Lambda = \emptyset$, and

$$\Gamma = \biguplus_{p \in role(\mathcal{T})} \Delta_P, \text{ with } \Delta_p = p^{\mathcal{I}} \cup \bigcup_{A \sqsubseteq \forall p.B \in \mathcal{T}} \{(a_{A_{P_B}}, b), (a_{\overline{A}_{P_B}}, \overline{o})\},$$

where b is an arbitrary instance of B, and $\overline{o}$ an arbitrary element of $\Delta^{\mathcal{I}}$. We set $O^{\mathcal{J}} = \Delta^{\mathcal{I}} \cup \Lambda$, $A^{\mathcal{J}} = A^{\mathcal{I}}$ for each class A corresponding to an atomic concept in $\mathcal{T}$, and $C^{\mathcal{J}}_P = \Delta_p$ for each $p \in role(\mathcal{T})$. Additionally, the extensions of the relations P_1 and P_2 are defined as follows: $P^{\mathcal{J}}_1 = \{((o_1, o_2), o_1) \mid (o_1, o_2) \in C^{\mathcal{J}}_P\}$, $P^{\mathcal{J}}_2 = \{((o_1, o_2), o_2) \mid (o_1, o_2) \in C^{\mathcal{J}}_P\}$. We now show that $\mathcal{J}$ is a full model of $\Sigma(\mathcal{T})$.

- For the portions of $\Sigma(\mathcal{T})$ due to TBox assertions of the form $A \sqsubseteq B$, $A \sqsubseteq \neg B$, and $A \sqsubseteq B_1 \sqcup B_2$, the statement follows from the construction of $\mathcal{J}$.

- For each TBox assertion in $\mathcal{T}$ of the form $A \sqsubseteq \forall p.B$, let us define the extensions for the *auxiliary* classes and relations as follows:

$$\begin{aligned}
&A^{\mathcal{J}}_{P_B} = A^{\mathcal{I}} \cup \{a_{A_{P_B}}\}, && \overline{A}^{\mathcal{J}}_{P_B} = O^{\mathcal{J}} \setminus A^{\mathcal{J}}_{P_B},\\
&C^{\mathcal{J}}_{P_{AB}} = \{(o, o') \in C^{\mathcal{J}}_P \mid o \in A^{\mathcal{J}}_{P_B}\}, && \overline{C}^{\mathcal{J}}_{P_{AB}} = \{(o, o') \in C^{\mathcal{J}}_P \mid o \in \overline{A}^{\mathcal{J}}_{P_B}\},\\
&P^{\mathcal{J}}_{AB1} = \{((o, o'), o) \in P^{\mathcal{J}}_1 \mid o \in A^{\mathcal{J}}_{P_B}\}, && P^{\mathcal{J}}_{\overline{A}B1} = \{((o, o'), o) \in P^{\mathcal{J}}_1 \mid o \in \overline{A}^{\mathcal{J}}_{P_B}\},\\
&P^{\mathcal{J}}_{AB2} = \{((o, o'), o') \in P^{\mathcal{J}}_2 \mid o \in A^{\mathcal{J}}_{P_B}\}.
\end{aligned}$$

  It is not difficult to see that $\mathcal{J}$ satisfies the fragment of $\Sigma(\mathcal{T})$ as shown in Fig. 7.1c. It remains to show that each class and each relation has a non-empty extension. This is clearly the case for classes that encode atomic concepts in $\mathcal{T}$. For the classes A_{P_B}, $\overline{A}_{P_B}$, $C_{P_{AB}}$, and $\overline{C}_{P_{AB}}$ we have that

$$a_{A_{P_B}} \in A^{\mathcal{J}}_{P_B}, \quad a_{\overline{A}_{P_B}} \in \overline{A}^{\mathcal{J}}_{P_B}, \quad (a_{A_{P_B}}, b) \in C^{\mathcal{J}}_{P_{AB}}, \quad (a_{\overline{A}_{P_B}}, \overline{o}) \in \overline{C}^{\mathcal{J}}_{P_{AB}}.$$

  For the relations P_1, P_2, P_{AB1}, P_{AB2}, and $P_{\overline{A}B1}$ we have that $((a_{A_{P_B}}, b), a_{A_{P_B}}) \in P^{\mathcal{J}}_{AB1} \subseteq P^{\mathcal{J}}_1$, $((a_{\overline{A}_{P_B}}, \overline{o}), a_{\overline{A}_{P_B}}) \in P^{\mathcal{J}}_{\overline{A}B1}$, $((a_{A_{P_B}}, b), b) \in P^{\mathcal{J}}_{AB2} \subseteq P^{\mathcal{J}}_2$.

- For each TBox assertion in $\mathcal{T}$ of the form $A \sqsubseteq \exists p.B$, let us define:

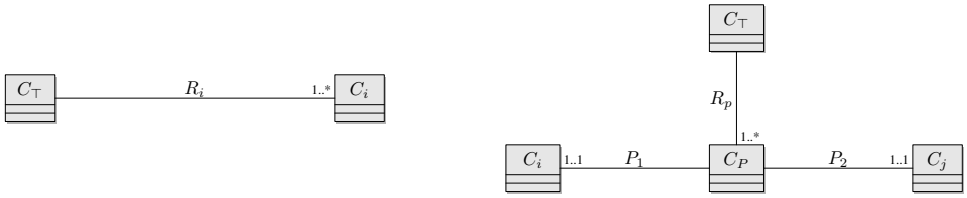


Figure 7.2: Reducing $\mathcal{D}_{full}$ full satisfiability to class satisfiability

$$
\begin{aligned}
C^{\mathcal{J}}_{P_{AB}} &= \{(o,o') \in C^{\mathcal{J}}_P \mid o \in A^{\mathcal{I}} \text{ and } o' \in B^{\mathcal{I}}\},\\
P^{\mathcal{J}}_{AB1} &= \{((o,o'),o) \in P^{\mathcal{J}}_1 \mid (o,o') \in C^{\mathcal{J}}_{P_{AB}}\},\\
P^{\mathcal{J}}_{AB2} &= \{((o,o'),o') \in P^{\mathcal{J}}_2 \mid (o,o') \in C^{\mathcal{J}}_{P_{AB}}\}\,.
\end{aligned}
$$

We have that $C^{\mathcal{J}}_{P_{AB}} \neq \emptyset$ as there exists a pair $(a,b) \in \Delta_p$ with $a \in A^{\mathcal{I}}$, and $b \in B^{\mathcal{I}}$. Since $C^{\mathcal{J}}_{P_{AB}} \neq \emptyset$, we have that $P^{\mathcal{J}}_{AB1} \neq \emptyset$ and $P^{\mathcal{J}}_{AB2} \neq \emptyset$.

❑

We can now establish the following.

Theorem 7.9. *Full satisfiability of $\mathcal{D}_{full}$ diagrams is* ExpTime*-complete.*

Proof. We establish the upper bound by a reduction to class satisfiability in $\mathcal{D}_{full}$ CDs, which is known to be ExpTime-complete by Theorem 7.2. Given a $\mathcal{D}_{full}$ diagram $\mathcal{D}$, with classes $C_1, \ldots, C_n$, we construct a $\mathcal{D}_{full}$ diagram $\mathcal{D}'$ by adding to $\mathcal{D}$ a new class $C_\top$ and new relations R_i, for $i \in \{1, \ldots, n\}$, as shown in the left part of Fig. 7.2. Furthermore, to check that every relation is populated we use reification, i.e., we replace each relation P in the diagram $\mathcal{D}$ between the classes C_i and C_j (such that neither C_i nor C_j is constrained to participate at least once to P) with a class C_P and two functional relations P_1 and P_2 to represent each component of P. Finally, we add the constraints shown in the right part of Fig. 7.2. Intuitively, we have that if there is an instantiation $\mathcal{I}$ of the extended diagram $\mathcal{D}'$ in which $C^{\mathcal{I}}_\top \neq \emptyset$, then the multiplicity constraint $1..*$ on the relation R_P forces the existence of at least one instance o of C_P. By the functionality of P_1 and P_2 there are at least two elements o_i and o_j, such that $o_i \in C^{\mathcal{I}}_i$, $o_j \in C^{\mathcal{I}}_j$, $(o,o_i) \in P^{\mathcal{I}}_1$ and $(o,o_j) \in P^{\mathcal{I}}_2$. Then, one instance of P can be the pair (o_i,o_j). Conversely, if there is a full model $\mathcal{J}$ of $\mathcal{D}$, it is easy to extend it to a model $\mathcal{I}$ of $\mathcal{D}'$ that satisfies $C_\top$.

The ExpTime-hardness follows from Lemma 7.8 and Theorem 7.7.

❑

Note that the proof for the lower bound (Lemma 7.8) being based on $\mathcal{ALC}$ holds also for the finite model case. Hence we have the following.

Corollary 7.10. *Full satisfiability of $\mathcal{D}_{full}$ diagrams is* ExpTime*-complete under the finite model assumption.*

We now move forward to showing the complexity of deciding full satisfiability for the two sub-languages $\mathcal{D}_{bool}$ and $\mathcal{D}_{ref}$. By building on the techniques used for the satisfiability proofs in [8], we show that also in this case checking for full satisfiability has the same complexity as for satisfiability.

We consider first $\mathcal{D}_{bool}$ diagrams, and show that deciding full satisfiability is NP-complete. For the lower bound, we provide a polynomial reduction of the 3SAT problem (which is known to be NP-complete) to full satisfiability of $\mathcal{D}_{bool}$ CDs.

Let an instance of 3SAT be given by a set $\phi = \{c_1, \ldots, c_m\}$ of 3-clauses over a finite set Π of propositional variables. Each clause is such that $c_i = \ell_i^1 \vee \ell_i^2 \vee \ell_i^3$, for $i \in \{1, \ldots, m\}$, where each ℓ_j^k is a literal, i.e., a variable or its negation. We construct an $\mathcal{D}_{bool}$ diagram $\mathcal{D}_\phi$ as follows: $\mathcal{D}_\phi$ contains the classes $C_\phi, C_\top$, one class C_i for each clause $c_i \in \phi$, and two classes C_p and $C_{\neg p}$ for each variable $p \in \Pi$.

To describe the constraints imposed by $\mathcal{D}_\phi$, we provide the corresponding CIs since they are more compact and concise than drawing the diagram. For every $i \in \{1, \ldots, m\}$, $j \in \{1, 2, 3\}$, and $p \in \Pi$, we have the assertions

$$\begin{array}{rclrclrcl} C_\phi & \sqsubseteq & C_\top & C_i & \sqsubseteq & C_\top & C_{l_i^j} & \sqsubseteq & C_i \\ C_p & \sqsubseteq & C_\top & C_\phi & \sqsubseteq & C_i & C_i & \sqsubseteq & C_{\ell_i^1} \sqcup C_{\ell_i^2} \sqcup C_{\ell_i^3} \\ C_{\neg p} & \sqsubseteq & C_\top & C_\top & \sqsubseteq & C_p \sqcup C_{\neg p} & C_{\neg p} & \sqsubseteq & \neg C_p \end{array}$$

Clearly, the size of $\mathcal{D}_\phi$ is polynomial in the size of ϕ.

Lemma 7.11. *A set ϕ of 3-clauses is satisfiable if and only if the $\mathcal{D}_{bool}$ class diagram $\mathcal{D}_\phi$, constructed as above, is fully satisfiable.*

Proof.
"$\Rightarrow$" Let $\mathcal{J} \models \phi$. Define an interpretation $\mathcal{I} = (\{0, 1\}, \cdot^\mathcal{I})$, with

$$C_\top^\mathcal{I} = \{0, 1\} \qquad C_\ell^\mathcal{I} = \begin{cases} \{1\}, & \text{if } \mathcal{J} \models \ell \\ \{0\}, & \text{otherwise} \end{cases}$$

$$C_i^\mathcal{I} = C_{\ell_i^1}^\mathcal{I} \cup C_{\ell_i^2}^\mathcal{I} \cup C_{\ell_i^3}^\mathcal{I}, \quad \text{for } c_i = \ell_i^1 \vee \ell_i^2 \vee \ell_i^3 \qquad C_\phi^\mathcal{I} = C_1^\mathcal{I} \cap \cdots \cap C_m^\mathcal{I}.$$

Clearly, $C^\mathcal{I} \neq \emptyset$ for every class C representing a clause or a literal, and for $C = C_\top$. Moreover, as at least one literal ℓ_i^j in each clause is such that $\mathcal{J} \models \ell_i^j$, then $1 \in C_i^\mathcal{I}$ for every $i \in \{1, \ldots, m\}$, and therefore $1 \in C_\phi^\mathcal{I}$. It is straightforward to check that $\mathcal{I}$ satisfies $\mathcal{T}$.

"$\Leftarrow$" Let $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ be a full model of $\mathcal{D}_\phi$. We construct a model $\mathcal{J}$ of ϕ by taking an element $o \in C_\phi^\mathcal{I}$, and setting, for every variable $p \in \Pi$, $\mathcal{J} \models p$ if and only if $o \in C_p^\mathcal{I}$. Let us show that $\mathcal{J} \models \phi$. Indeed, for each $i \in \{1, \ldots, m\}$ since $o \in C_\phi^\mathcal{I}$ and by the generalization $C_\phi \sqsubseteq C_i$, we have that $o \in C_i^\mathcal{I}$, and by the completeness constraint $C_i \sqsubseteq C_{\ell_i^1} \sqcup C_{\ell_i^2} \sqcup C_{\ell_i^3}$, there is some $j_i \in \{1, 2, 3\}$ such that $o \in C_{\ell_i^{j_i}}$. If $\ell_i^{j_i}$ is a variable, then $\mathcal{J} \models \ell_i^{j_i}$ by construction, and thus $\mathcal{J} \models c_i$. Otherwise,

if $\ell_i^{j_i} = \neg p$ for some variable p, then, by the disjointness constraint $C_{\neg p} \sqsubseteq \neg C_p$, we have that $o \notin C_p^{\mathcal{I}}$. Thus, $\mathcal{J} \models \neg p$, and therefore, $\mathcal{J} \models c_i$.

❑

Theorem 7.12. *Full satisfiability of $\mathcal{D}_{bool}$ is* NP*-complete*

Proof. To prove the NP upper bound, we reduce full satisfiability to class satisfiability, which, for the case of $\mathcal{D}_{bool}$, is known to be in NP. We use an encoding similar to the one used in the proof of Theorem 7.9 (see Fig. 7.2).

❑

We turn now to $\mathcal{D}_{ref}$ class diagrams and show that full satisfiability in this case is NLogSpace-complete. We provide a reduction of the reachability problem on (acyclic) directed graphs, which is known to be NLogSpace-complete (see e.g., [107]) to the complement of full satisfiability of $\mathcal{D}_{ref}$ diagrams.

Let $G = (V, E, s, t)$ be an instance of reachability, where V is a set of vertices, $E \subseteq V \times V$ is a set of directed edges, s is the start vertex, and t the terminal vertex. We construct an $\mathcal{D}_{ref}$ diagram $\mathcal{D}_G$ from G as follows:

- $\mathcal{D}_G$ has two classes C_v^1 and C_v^2, for each vertex $v \in V \setminus \{s\}$, and one class C_s corresponding to the start vertex s.
- For each edge $(u, v) \in E$ with $u \neq s$ and $v \neq s$, $\mathcal{D}_G$ contains the following constraints (again expressed as DL inclusion assertions):

$$C_u^1 \sqsubseteq C_v^1, \qquad C_u^2 \sqsubseteq C_v^2.$$

- For each edge $(s, v) \in E$, $\mathcal{D}_G$ contains the following constraints: $C_s \sqsubseteq C_v^1$, $C_s \sqsubseteq C_v^2$.
- For each edge $(u, s) \in E$, $\mathcal{D}_G$ contains the following constraints:

$$C_u^1 \sqsubseteq C_s, \qquad C_u^2 \sqsubseteq C_s.$$

- The classes C_t^1 and C_t^2 are constrained to be disjoint in $\mathcal{D}$, expressed by:

$$C_t^1 \sqsubseteq \neg C_t^2.$$

The following lemma establishes the correctness of the reduction.

Lemma 7.13. *t is reachable from s in G iff $\mathcal{D}_G$ is not fully satisfiable.*

Proof. "⇒" Let $\pi = v_1, \ldots, v_n$ be a path in G with $v_1 = s$ and $v_n = t$. We claim that the class C_s in the constructed diagram $\mathcal{D}_G$ is unsatisfiable. Suppose otherwise that there is a model $\mathcal{I}$ of $\mathcal{D}_G$ with $o \in C_s^{\mathcal{I}}$, for some $o \in \Delta^{\mathcal{I}}$. From π, a number of generalization constraints hold in $\mathcal{D}_G$, i.e., $C_s^{\mathcal{I}} \subseteq C_t^{1\mathcal{I}}$ and $C_s^{\mathcal{I}} \subseteq C_t^{2\mathcal{I}}$. Thus, we obtain that $o \in (C_t^1)^{\mathcal{I}}$ and $o \in (C_t^2)^{\mathcal{I}}$, which violates the disjointness between the classes C_t^1 and C_t^2, in contradiction to $\mathcal{I}$ being a model of $\mathcal{D}_G$. Hence, C_s is unsatisfiable, and therefore $\mathcal{D}_G$ is not fully satisfiable.

"⇐" Let us consider the contrapositive. Assume that t is not reachable from s in G. We construct a full model $\mathcal{I}$ of $\mathcal{D}_G$. Let $\Delta^{\mathcal{I}} = \{d_s\} \cup \bigcup_{v \in V \setminus \{s\}} \{d_v^1, d_v^2\}$. Define inductively a sequence of interpretations as follows:

$\mathcal{I}^0 = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}^0})$, such that: $C_s^{\mathcal{I}^0} = \{d_s\},\ C_v^{i\,\mathcal{I}^0} = \{d_v^i\}, \forall i \in \{1,2\}, v \in V \setminus \{s\}$,

$\mathcal{I}^{n+1} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}^{n+1}})$, such that: $C_s^{\mathcal{I}^{n+1}} = C_s^{\mathcal{I}^n} \cup \bigcup_{(u,s) \in E} (C_u^{1\,\mathcal{I}^n} \cup C_u^{2\,\mathcal{I}^n}),\ C_v^{i\,\mathcal{I}^{n+1}} = C_v^{i\,\mathcal{I}^n} \cup \bigcup_{(u,v) \in E,\, u \neq s} C_u^{i\,\mathcal{I}^n} \cup \bigcup_{(s,v) \in E} C_s^{\mathcal{I}^n}$.

The definition induces a monotone operator over a complete lattice, and hence it has a fixed point. Let $\mathcal{I}$ be defined by such a fixed point. It is easy to check that $\mathcal{I}$ is such that for all $i \in \{1,2\}$, and $u, v \in V \setminus \{s\}$ the following holds:

1. For each class C_v^i, we have that $d_v^i \in C_v^{i\mathcal{I}}$.
2. $d_s \in C_s^{\mathcal{I}}$.
3. For all $d \in \Delta^{\mathcal{I}}$, $d \in C_u^{i\mathcal{I}}$ implies $d \in C_v^{i\mathcal{I}}$ iff v is reachable from u in G.
4. For all $d_u^i \in \Delta^{\mathcal{I}}$, $d_u^i \in C_v^{j\mathcal{I}}$ for $i \neq j$ iff s is reachable from u in G, and v is reachable from s in G.
5. $d_s \in C_v^{i\mathcal{I}}$ iff v is reachable from s in G.

From (1) and (2) we have that all classes in $\mathcal{D}_G$ are populated in $\mathcal{I}$. It remains to show that $\mathcal{I}$ satisfies $\mathcal{D}_G$. A generalization between the classes C_u^i and C_v^i corresponds to the edge $(u,v) \in E$. This means that v is reachable from u in G, and therefore, by (3) we have that $C_u^{i\mathcal{I}} \subseteq C_v^{i\mathcal{I}}$. A similar argument holds for generalizations involving the class C_s. Furthermore, the classes C_t^1 and C_t^2 are disjoint under $\mathcal{I}$. To show this, suppose that there is an element $d \in \Delta^{\mathcal{I}}$ such that $d \in C_t^{1\mathcal{I}} \cap C_t^{2\mathcal{I}}$. Then by (5), $d \neq d_s$, as t is not reachable from s. Moreover, $d \neq d_v^i$ for all $i \in \{1,2\}$ and $v \in V \setminus \{s\}$. Indeed, suppose w.l.o.g. that $i = 1$. Then, by (4), $d_v^1 \in C_t^{2\mathcal{I}}$ iff s is reachable from v, and t is reachable from s, which leads to a contradiction. Hence, $C_t^{1\mathcal{I}} \cap C_t^{2\mathcal{I}} = \emptyset$. ❑

Theorem 7.14. *Full satisfiability of $\mathcal{D}_{ref}$ class diagrams is* NLogSpace*-complete.*

Proof. The NLogSpace membership follows from the NLogSpace membership of class satisfiability [8], and a reduction similar to the one used in Theorem 7.12. Since NLogSpace = coNLogSpace (by the Immerman-Szelepcsényi theorem; see, e.g., [107]), and as the above reduction is logspace bounded, it follows that full satisfiability of $\mathcal{D}_{ref}$ diagrams is NLogSpace-hard. ❑

Finite Model Assumption

Finally, we can argue that the complexity boundaries for full satisfiability of $\mathcal{D}_{full}$ and $\mathcal{D}_{ref}$ are tight also under the finite model assumption. More precisely, we say that a class diagram $\mathcal{D}$ is *full satisfiable under the finite model assumption* iff there is an instantiation $\mathcal{I}$ of $\mathcal{D}$ such that the domain of $\mathcal{I}$ is finite and every class has at least one instance.

Theorem 7.15. *Full satisfiability under the finite model assumption of $\mathcal{D}_{full}$ diagrams is* ExpTime *complete.*

Proof. For the case of $\mathcal{D}_{full}$ diagrams, the ExpTime-hardness follows from Lemma 7.8, Theorem 7.7 and the fact that $\mathcal{ALC}$ has the finite model property. The upper bound, on the other hand follows from a reduction of full satisfiability on $\mathcal{D}_{full}$ to finite satisfiability on $\mathcal{ALCQI}$ (which is ExpTime complete). Indeed, let $\mathcal{D}$ be a $\mathcal{D}_{full}$ diagram, and let $\mathcal{T}_{\mathcal{D}}$ the corresponding $\mathcal{ALCQI}$ TBox. We can show the following: $\mathcal{D}$ is fully-satisfiable by a finite instantiation iff the ontology $(\mathcal{T}_{\mathcal{D}}, \mathcal{A}_{\mathcal{D}})$ is finitely satisfiable, where $\mathcal{A}_{\mathcal{D}} := \{A_C(a_C) \mid C \text{ is a class in } \mathcal{D}\} \cup \{A_R(a_R) \mid R \text{ is a relation in } \mathcal{D}\}$. Note that the ABox $\mathcal{A}_{\mathcal{D}}$ forces the existence of at least one object a_C from each class C, and a 'tuple' a_R for each relation R. ❑

Analogous results can be obtained for restricted class diagrams that are formalized by the tractable DLs from the *DL-Lite* family.

Theorem 7.16. *Full satisfiability under the finite model assumption is decidable in* PTime *for lite–$\mathcal{D}^{-}_{Horn}$ and, is* NLogSpace*-complete for $\mathcal{D}^{-}_{ref}$ diagrams.*

Proof. In both cases, the upper complexity bound follows from the encoding of these classes of diagrams in DLs. Recall that a *lite–$\mathcal{D}^{-}_{Horn}$* diagram $\mathcal{D}$ can be encoded into a *DL-Lite*$^{\mathcal{F}}_{Horn}$ TBox $\mathcal{T}_{\mathcal{D}}$, is such a way that a class C in $\mathcal{D}$ is finitely satisfiable iff the concept A_C (encoding C) is finitely satisfiable w.r.t. $\mathcal{T}_{\mathcal{D}}$, and that finite (concept) satisfiability is PTime-complete for *DL-Lite*$^{\mathcal{F}}_{Horn}$. Analogously, $\mathcal{D}^{-}_{ref}$ diagrams can be encoded using *DL-Lite*$^{\mathcal{F}}_{core}$, for which deciding finite satisfiability is NLogSpace-complete.

The lower bound for the case of $\mathcal{D}^{-}_{ref}$ follows from the reduction in Lemma 7.13. ❑

Chapter

8 Conclusions

In this thesis, we have investigated the complexity of ontological reasoning under assumptions that are relevant for database applications.

Finite Models

In Chapter 3, we established the computational complexity of the standard reasoning (satisfiability, subsumption and ABox consistency) under the finite model assumption in DLs that lack the finite model property. The study on that Chapter focused on so-called Horn DLs, including the lightweight logics of the *DL-Lite* family with functional roles [38], and the more expressive logics Horn-$\mathcal{ALCFI}$ and Horn-$\mathcal{ALCQI}$ [71]. We have shown that the *cycle reversion technique*, originally introduced to axiomatize finite implication of functional dependencies and inclusion dependencies by Cosmadakis et al. [45] in the context of databases, and later extended to *DL-Lite*$^{\mathcal{F}}_{core}$ by Rosati [117], extends all the way to the Horn-$\mathcal{ALCQI}$ description logic. From our results on this extension, we can conclude the following.

- For the Horn DLs considered, the cycle reversion technique provides a complete axiomatization of finite model entailment; and as a consequence,
- finite model reasoning in these logics can be reduced to unrestricted reasoning w.r.t. the ontology obtained after cycle reversion (Theorems 3.10, 3.24 and 3.28).
- For the case of *DL-Lite*$^{\mathcal{F}}_{Horn}$, the previous Theorem 3.10 implies that the complexity of finite model reasoning coincides with that of reasoning over unrestricted models, and therefore it is PTime-complete.
- For the case of Horn-$\mathcal{ALCQI}$, on the other hand, the cycle reversion defined in Section 3.3 produces an exponential blow up in the input TBox, hence a double exponential decision procedure for finite model reasoning in Horn-$\mathcal{ALCQI}$, which is far from optimal since

complexity of that problem was known to be in ExpTime since finite model reasoning in $\mathcal{ALCQI}$ is ExpTime complete.

Consequence driven Finite Model Reasoning. Since the cycle reversion technique developed in Chapter 3 does not provides an optimal decision procedure for finite model reasoning in Horn-$\mathcal{ALCQI}$, we devised in Chapter 4 an algorithmic approach to finite model reasoning in Horn DLs. We adapted and extend a well known consequence driven algorithm for deciding subsumption in Horn-$\mathcal{SHIQ}$. The obtained method provides a sound and complete decision procedure to realize finite model reasoning in Horn-$\mathcal{ALCFI}$. In fact, our procedure can be used to compute all subsumptions $A \sqsubseteq B$ between concept names holding on finite models of the input ontology. The cycle reversion technique implemented by our rule **R9** avoids the exponential blow-up on the input TBox and ensures that the procedure is computationally optimal (Theorem 4.9). Moreover, the addition of rules **R10** to **R12** enables ABox reasoning in Horn-$\mathcal{ALCFI}$. Completeness of the approach follows from the reduction of finite ABox consistency in Horn$\mathcal{ALCFI}$ to the unrestricted model case provided by Theorem 3.28. Finally, we have shown that a Horn-$\mathcal{ALCQI}$ $\mathcal{T}$ TBox can be transformed into a Horn-$\mathcal{ALCFI}$ TBox $\mathcal{T}'$, that is, "at least" number restrictions for $n > 1$ present in a Horn-$\mathcal{ALCQI}$ can be coded out in such a way that finite model reasoning in $\mathcal{T}$ and $\mathcal{T}'$ coincides. However, it is not obvious how to adapt the transformation to account also for a Horn-$\mathcal{ALCQI}$ ABox.

Ontological Query Answering under the Finite Model Assumption. The finite model assumption is also relevant for ontological query answering, when the ontology is expressed in a logic that lacks the *finite model property*. Indeed, the finite model version of that problem is different to the unrestricted case since ontological query answering amounts to entailment. In Chapter 5, we have shown that we can actually reduce finite query entailment w.r.t. an ontology to unrestricted (model) query entailment, provided that the TBox has been extended using cycle reversion (Theorem 5.1).

Our reduction allows us to establish the computational complexity of finite model query entailment w.r.t. an ontology. As for the standard reasoning tasks, the complexity of finite ontological query answering coincides with that of query answering w.r.t. unrestricted models on the ontology.

- Finite PEQ entailment in Horn-$\mathcal{ALCFI}$ is decidable;
- ExpTime-complete in combined complexity;
- and PTime-complete in data complexity.

The reduction of finite query entailment to unrestricted entailment from Theorem 5.1 covers also the fragments $\mathcal{ELIF}$ and *DL-Lite*$^{\mathcal{F}}_{Horn}$ of Horn-$\mathcal{ALCFI}$. Indeed, the properties of the finite models established in Propositions 5.3 and 5.4 hold also for these fragments. For the case of $\mathcal{ELIF}$ the complexity of finite query entailment coincides with that of Horn-$\mathcal{ALCFI}$ since this also holds for the unrestricted case. On the other hand, since the size of the $\mathsf{finClosure}(\mathcal{T})$ of a

DL-Lite$^{\mathcal{F}}_{Horn}$ TBox $\mathcal{T}$ is bounded by a polynomial on the size of $\mathcal{T}$ and it does not depend on the size of the ABox, the complexity of query entailment in *DL-Lite*$^{\mathcal{F}}_{Horn}$ for the unrestricted case and Theorem 5.1 imply that:

- Finite query entailment in *DL-Lite*$^{\mathcal{F}}_{Horn}$ is PTIME-complete in combined complexity;
- and AC^0 in data complexity.

Expressive Query Languages

In Chapter 6, we studied ontological query answering on Horn DLs, in the scenario where queries contain some form of negation. In classical database theory, positive queries have played a key role due to their desirable theoretical properties (e.g., they are invariant under homomorphisms), unfortunately, they do not allow to express e.g., *complementation* and *difference*, which might be of interest in practice. While it is well-known in database theory that conjunctive query answering with negation is harder that answering positive queries, some interest has risen to provide more expressive means to query ontologies [6, 19, 116].

The results in this thesis regarding ontological query answering confirm that the extension with negations on queries is unfeasible even on fairly restricted scenarios. Indeed, it was known from the results by Rosati that answering both UCQs$^{\neq}$ and UCQs$^{\neg s}$ in *DL-Lite*$^{\mathcal{H}}_{core}$ is undecidable [116]. Rosati also showed that the situation is similar for $\mathcal{EL}$: answering *unions of* CQs$^{\neg}$ undecidable. Moreover, the result by Klenke showed that (single) CQ$^{\neq}$ answering in $\mathcal{EL}$ is also undecidable [82]. However, from those results it was not clear whether for *DL-Lite* the source of undecidability was to consider *unions* of queries with negation. Notably, this kind of queries amount to very expressive constraints. Indeed, recall that entailment of a query q over an ontology $\mathcal{O}$ is equivalent to decide whether $\mathcal{O} \wedge \neg q$ is satisfiable (i.e., to find a counter model). Then, the negation of a union of CQs$^{\neq}$ can express, for example, a set of complex functionality constraints (the so-called *disjunctive EGDs*).

Safe Negation. For the case of queries with negation, we have shown that answering CQs$^{\neg s}$ over $\mathcal{ELI}_{\bot}$ ontologies is undecidable. It turns out that a single query with safe negation is enough to encode undecidable problems if the ontology is expressed in *DL-Lite with* role inclusion axioms. In particular, we have shown that answering CQs$^{\neg s}$ in *DL-Lite*$^{\mathcal{H}}_{core}$ is undecidable. This result indicates that safe negation is not a condition strong enough to retain decidability of ontological query answering, although, the decidability of answering CQs$^{\neg s}$ in *DL-Lite without* role inclusions remains open. Even if decidability can be attained for the case without role inclusions, our results on guarded negation indicate that the problem is not tractable.

Guarded Negation. We considered queries with guarded negation, a restriction that was proposed to provide decidability of ontological query answering [19]. We have shown that answering GNCQs over *DL-Lite*$_{core}$, *DL-Lite*$^{\mathcal{H}}_{core}$, $\mathcal{EL}_{\bot}$ and $\mathcal{ELI}_{\bot}$ is coNP-complete in data complexity. Further, if we restrict the number of negations to only *one*, the problem becomes

PTime-complete. One possible explanation for the latter comes from the observation that such restriction disallows expressing *disjunctive knowledge* in $\mathcal{O} \wedge \neg q$, which is one of the characteristics of Horn DLs and the reason why ontological positive query answering is tractable in those DLs. The restriction on the number of guarded negations is tight. Indeed, the proof to show coNP-hardness required only two guarded negations.

Inequalities. For the case of conjunctive queries with inequalities, it is known that the number of inequalities in the query may have an impact on the complexity of ontological query answering [83, 96]. We have shown that for *DL-Lite*, the presence of role inclusions makes irrelevant the number of inequalities. In particular, that answering CQs$^{\neq}$ with *one* inequality in *DL-Lite*$^{\mathcal{H}}_{core}$ is undecidable.

The decidability of CQ$^{\neq}$ answering in *DL-Lite*$_{core}$ remains open, although in case that problem is decidable, tractability cannot be expected even for restricted queries with at most 2 inequalities. We provided a coNP lower bound for that scenario.

Future Work

More Expressive DLs

As future research, it would be interesting to extend the results in this thesis to Horn-$\mathcal{SHIQ}$ [71], that is, to add role hierarchies and transitive roles. Actually, transitive roles can be "reduced out" is such a way that TBox finite reasoning and finite ABox consistency is preserved. On the other hand, role hierarchies need to be explicitly taken into account when defining the cycle reversion technique and would have to be built directly into our construction of finite models. Reducing out role hierarchies does not seem easily possible in the finite[1]. It would also be interesting to provide a transparent reduction that does not require to eliminate at least number restrictions for $n > 1$. Recall that the transformation from Horn-$\mathcal{ALCQI}$ TBoxes to Horn-$\mathcal{ALCFI}$ in Section 3.4 does not preserve ABox consistency, and therefore is useful only for TBox reasoning.

For query entailment, we expect transitive roles to cause significant additional challenges, see for example [52, 100]. In particular, transitive roles result in an additional way in which the finite model property is lost. Consider for example the TBox $\mathcal{T}$ containing the axioms:

$$A \sqsubseteq \exists r.A, \qquad \mathsf{trans}(r)$$

and the conjunctive query

$$q = \exists x\, r(x, x).$$

For the ABox $\mathcal{A} = \{A(a)\}$, we have $(\mathcal{T}, \mathcal{A}) \not\models q$, but $(\mathcal{T}, \mathcal{A}) \models_{\mathsf{fin}} q$ although neither counting nor inverse roles are present in $\mathcal{T}$, which is formulated in the extension of $\mathcal{EL}$ with transitive roles. Finite model reasoning in versions of Datalog$^{\pm}$ [28] that extend $\mathcal{EL}_{\mathsf{trans}}$ has recently been studied in [57, 58].

[1] In contrast to what we have claimed in [74].

Size of Finite Models

In this thesis, we have not analyzed the size of finite models. It is, however, easy to prove a double exponential lower bound on the size of finite models for satisfiability in Horn-$\mathcal{ALCFI}$ by enforcing a tree of exponential depth in which no two elements can be identical. A matching upper bound follows from Pratt-Hartmann's result that every finitely satisfiable formula in first-order logic with two variables and counting quantifiers has a model of at most double exponential size [111]. Nonetheless, analyzing the size of finite (counter)models for query entailment seems a promising line for future work.

Expressive Query Languages

Regarding ontological query answering with expressive query languages, the most promising line for future work could be the study of syntactical restrictions of CQs$^{\neq}$. In fact, not only the number of inequalities affects the complexity of query answering, but also restrictions on the way variables occurring on inequalities can be bound in a match of the query do. Remarkably, Arenas et al. [6] have investigated such restriction in the context of data exchange.

Bibliography

[1] S. Abiteboul and O. M. Duschka. Complexity of answering queries using materialized views. In *Proc. of PODS*, pages 254–263. ACM Press, 1998. ISBN 0-89791-996-3.

[2] S. Abiteboul and O. M. Duschka. Complexity of answering queries using materialized views. Technical Report Gemo Report 383, INRIA Saclay, 1999.

[3] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995. ISBN 0-201-53771-0.

[4] N. Alon. Tools from higher algebra. In R. L. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of combinatorics (vol. 2)*, pages 1749–1783. MIT Press, Cambridge, MA, USA, 1995. ISBN 0-262-07171-1.

[5] H. Andréka, I. Németi, and J. van Bentham. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27:217–274, 1998.

[6] M. Arenas, P. Barceló, and J. L. Reutter. Query languages for data exchange: Beyond unions of conjunctive queries. *Theory Comput. Syst.*, 49(2):489–564, 2011.

[7] A. Artale, F. Cesarini, and G. Soda. Describing database objects in a concept language environment. *IEEE Trans. Knowl. Data Eng.*, 8(2):345–351, 1996.

[8] A. Artale, D. Calvanese, R. Kontchakov, V. Ryzhikov, and M. Zakharyaschev. Reasoning over Extended ER Models. In *Proc. of the 26th Int. Conf. on Conceptual Modeling (ER 2007)*, volume 4801 of *Lecture Notes in Computer Science*, pages 277–292. Springer, 2007.

[9] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The *DL-Lite* family and relations. *J. of Artificial Intelligence Research*, 36:1–69, 2009.

[10] A. Artale, D. Calvanese, and A. Ibanez-Garcia. Checking full satisfiability of conceptual models. In *Proc. of the 23nd Int. Workshop on Description Logic (DL 2010)*, volume 573 of *CEUR Electronic Workshop Proceedings, http://ceur-ws.org/*, pages 55–66, 2010.

[11] A. Artale, D. Calvanese, and A. Ibanez-Garcia. Full Satisfiability of UML Class Diagrams. In J. Parsons, M. Saeki, P. Shoval, C. C. Woo, and Y. Wand, editors, *ER*, volume 6412 of *Lecture Notes in Computer Science*, pages 317–331. Springer, 2010. ISBN 978-3-642-16372-2.

[12] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[13] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 364–369, 2005.

[14] F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL – a polynomial-time reasoner for life science ontologies. In *Proc. IJCAR-06*, volume 4130 of *LNCS*, pages 287–291. Springer, 2006. ISBN 3-540-37187-7.

[15] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope further. In K. Clark and P. F. Patel-Schneider, editors, *In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*, 2008.

[16] J.-F. Baget, M.-L. Mugnier, S. Rudolph, and M. Thomazo. Walking the complexity lines for generalized guarded existential rules. In T. Walsh, editor, *IJCAI*, pages 712–717. IJCAI/AAAI, 2011. ISBN 978-1-57735-516-8.

[17] V. Bárány, G. Gottlob, and M. Otto. Querying the guarded fragment. In *LICS*, pages 1–10, 2010.

[18] V. Bárány, B. ten Cate, and L. Segoufin. Guarded negation. In *Automata, Languages and Programming - 38th International Colloquium (ICALP)*, pages 356–367, 2011.

[19] V. Bárány, B. ten Cate, and M. Otto. Queries with guarded negation. *PVLDB*, 5(11): 1328–1339, 2012.

[20] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv.*, 18(4):323–364, Dec. 1986. ISSN 0360-0300.

[21] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1–2):70–118, 2005.

[22] S. Bergamaschi and C. Sartori. On taxonomic reasoning in conceptual design. *ACM Trans. Database Syst.*, 17(3):385–422, Sept. 1992. ISSN 0362-5915.

[23] A. Borgida, M. Lenzerini, and R. Rosati. Description logics for databases. In Baader et al. [12], pages 462–484. ISBN 0-521-78176-0.

[24] R. J. Brachman and H. J. Levesque, editors. *Readings in Knowledge Representation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1985. ISBN 093461301X.

[25] G. Brewka, T. Eiter, and S. A. McIlraith, editors. *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*, 2012. AAAI Press. ISBN 978-1-57735-560-1.

[26] B. Bruegge and A. H. Dutoit. *Object-Oriented Software Engineering Using UML, Patterns, and Java*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009. ISBN 0136061257, 9780136061250.

[27] M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *J. of Artificial Intelligence Research*, 1:109–138, 1993.

[28] A. Calì, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In J. Paredaens and J. Su, editors, *PODS*, pages 77–86. ACM, 2009. ISBN 978-1-60558-553-6.

[29] A. Calì, G. Gottlob, and A. Pieris. New expressive languages for ontological query answering. In W. Burgard and D. Roth, editors, *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. AAAI Press, 2011. URL `http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3620`.

[30] A. Calì, G. Gottlob, G. Orsi, and A. Pieris. On the interaction of existential rules and equality constraints in ontology querying. In *Correct Reasoning - Essays on Logic-Based AI in Honour of Vladimir Lifschitz*, pages 117–133, 2012.

[31] D. Calvanese. Finite model reasoning in description logics. In *KR*, pages 292–303, 1996.

[32] D. Calvanese and G. De Giacomo. Expressive description logics. In Baader et al. [12], pages 178–218.

[33] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. 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.

[34] D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, pages 229–264. Kluwer Academic Publishers, 1998.

[35] D. Calvanese, M. Lenzerini, and D. Nardi. Unifying class-based representation formalisms. *J. of Artificial Intelligence Research*, 11:199–240, 1999.

[36] D. Calvanese, G. De Giacomo, and M. Lenzerini. Ontology of integration and integration of ontologies. In C. A. Goble, D. L. McGuinness, R. Möller, and P. F. Patel-Schneider, editors, *Description Logics*, volume 49 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2001.

[37] D. Calvanese, G. De Giacomo, and M. Lenzerini. Description logics for information integration. In A. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond Essays in Honour of Robert A. Kowalski*, volume 2408 of *Lecture Notes in Computer Science*, pages 41–60. Springer, 2002.

[38] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. *DL-Lite*: Tractable description logics for ontologies. In M. M. Veloso and S. Kambhampati, editors, *AAAI*, pages 602–607. AAAI Press / The MIT Press, 2005. ISBN 1-57735-236-X.

[39] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In P. Doherty, J. Mylopoulos, and C. A. Welty, editors, *KR*, pages 260–270. AAAI Press, 2006. ISBN 978-1-57735-271-6.

[40] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[41] D. Calvanese, G. De Giacomo, and M. Lenzerini. Conjunctive query containment and answering under description logic constraints. *ACM Trans. Comput. Log.*, 9(3), 2008.

[42] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, and R. Rosati. Ontologies and databases: The *DL-Lite* approach. In S. Tessaris, E. Franconi, T. Eiter, C. Gutierrez, S. Handschuh, M.-C. Rousset, and R. A. Schmidt, editors, *Reasoning Web*, volume 5689 of *Lecture Notes in Computer Science*, pages 255–356. Springer, 2009. ISBN 978-3-642-03753-5.

[43] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Int. J. Cooperative Inf. Syst.*, 2(4):375–398, 1993.

[44] P. P.-S. Chen. The entity-relationship model—toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, Mar. 1976. ISSN 0362-5915.

[45] S. S. Cosmadakis, P. C. Kanellakis, and M. Y. Vardi. Polynomial-time implication problems for unary inclusion dependencies. *J. ACM*, 37(1):15–46, 1990.

[46] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In B. Hayes-Roth and R. E. Korf, editors, *AAAI*, pages 205–212. AAAI Press / The MIT Press, 1994. ISBN 0-262-61102-3.

[47] G. De Giacomo and M. Lenzerini. TBox and ABox reasoning in expressive description logics. In L. C. Aiello, J. Doyle, and S. C. Shapiro, editors, *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96),*

Cambridge, Massachusetts, USA, November 5-8, 1996., pages 316–327. Morgan Kaufmann, 1996. ISBN 1-55860-421-9.

[48] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In R. Meersman, Z. Tari, and S. M. Stevens, editors, *Database Semantics - Semantic Issues in Multimedia Systems, IFIP TC2/WG2.6 Eighth Working Conference on Database Semantics (DS-8), Rotorua, New Zealand, January 4-8, 1999*, volume 138 of *IFIP Conference Proceedings*, pages 351–369. Kluwer, 1999. ISBN 0-7923-8405-9.

[49] A. Deutsch, A. Nash, and J. B. Remmel. The chase revisited. In *Proc. of the 27th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 149–158. ACM Press, 2008.

[50] F. M. Donini. Complexity of reasoning. In Baader et al. [12], pages 96–136.

[51] T. Eiter, G. Gottlob, M. Ortiz, and M. Simkus. Query answering in the description logic Horn-$\mathcal{SHIQ}$. In *Proc. JELIA-08*, volume 5293 of *LNCS*, pages 166–179. Springer, 2008. ISBN 978-3-540-87802-5.

[52] T. Eiter, C. Lutz, M. Ortiz, and M. Simkus. Query answering in description logics with transitive roles. In *Proc. IJCAI-09*, pages 759–764, 2009.

[53] T. Eiter, M. Ortiz, M. Šimkus, T.-K. Tran, and G. Xiao. Towards practical query answering for Horn-$\mathcal{SHIQ}$. In *Proc. DL-12*, volume 846 of *CEUR-WS.org*, 2012.

[54] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[55] B. Glimm, C. Lutz, I. Horrocks, and U. Sattler. Conjunctive query answering for the description logic shiq. *J. Artif. Intell. Res. (JAIR)*, 31:157–204, 2008.

[56] F. Goasdoué, V. Lattès, and M. Rousset. The use of CARIN language and algorithms for information integration: The PICSEL system. *Int. J. Cooperative Inf. Syst.*, 9(4): 383–401, 2000. doi: 10.1142/S0218843000000181. URL `http://dx.doi.org/10.1142/S0218843000000181`.

[57] T. Gogacz and J. Marcinkowski. Converging to the chase - a tool for finite controllability. In *Proc. LICS-13*, pages 540–549. IEEE Computer Society, 2013. ISBN 978-1-4799-0413-6.

[58] T. Gogacz and J. Marcinkowski. On the BDD/FC conjecture. In *Proc. PODS-13*, pages 127–138. ACM, 2013. ISBN 978-1-4503-2066-5.

[59] E. Grädel. Description logics and guarded fragments of first order logic. In E. Franconi, G. D. Giacomo, R. M. MacGregor, W. Nutt, and C. A. Welty, editors, *Proceedings of the 1998 International Workshop on Description Logics (DL'98), IRST, Povo - Trento,*

Italy, June 6-8, 1998, volume 11 of *CEUR Workshop Proceedings*. CEUR-WS.org, 1998. URL `http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-11/graedel.ps`.

[60] E. Grädel and M. Otto. On logics with two variables. *Theor. Comput. Sci.*, 224(1-2): 73–113, 1999. doi: 10.1016/S0304-3975(98)00308-9. URL `http://dx.doi.org/10.1016/S0304-3975(98)00308-9`.

[61] E. Grädel and I. Walukiewicz. Guarded fixed point logic. In *14th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 45–54, 1999.

[62] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. F. Patel-Schneider, and U. Sattler. OWL 2: The next step for OWL. *J. Web Sem.*, 6(4):309–322, 2008. doi: 10.1016/j.websem.2008.05.001. URL `http://dx.doi.org/10.1016/j.websem.2008.05.001`.

[63] V. Gutiérrez-Basulto, Y. A. Ibáñez-García, and R. Kontchakov. An update on query answering with restricted forms of negation. In M. Krötzsch and U. Straccia, editors, *Proc of the 6th International Conference on Web Reasoning and Rule Systems - RR 2012*, volume 7497 of *Lecture Notes in Computer Science*, pages 75–89. Springer, 2012. ISBN 978-3-642-33202-9.

[64] V. Gutiérrez-Basulto, Y. A. Ibáñez-García, R. Kontchakov, and E. V. Kostylev. Conjunctive queries with negation over *DL-Lite*: A closer look. In W. Faber and D. Lembo, editors, *Proc of the 7th International Conference on Web Reasoning and Rule Systems - RR 2013*, volume 7994 of *Lecture Notes in Computer Science*, pages 109–122. Springer, 2013. ISBN 978-3-642-39665-6. Best Student Paper Award.

[65] D. Harel. Effective transformations on infinite trees, with applications to high undecidability, dominoes, and fairness. *J. ACM*, 33(1):224–248, 1986.

[66] P. J. Hayes. The logic of frames. In Brachman and Levesque [24], pages 287–295. ISBN 093461301X.

[67] A. Hernich, C. Kupke, T. Lukasiewicz, and G. Gottlob. Well-founded semantics for extended datalog and ontological reasoning. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, (PODS 2013)*, pages 225–236, 2013.

[68] S. Heymans, L. Ma, D. Anicic, Z. Ma, N. Steinmetz, Y. Pan, J. Mei, A. Fokoue, A. Kalyanpur, A. Kershenbaum, E. Schonberg, K. Srinivas, C. Feier, G. Hench, B. Wetzstein, and U. Keller. Ontology reasoning with large data repositories. In M. Hepp, P. D. Leenheer, A. de Moor, and Y. Sure, editors, *Ontology Management*, volume 7 of *Semantic Web And Beyond Computing for Human Experience*, pages 89–128. Springer, 2008. ISBN 978-0-387-69900-4.

[69] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: the making of a web ontology language. *J. Web Sem.*, 1(1):7–26, 2003. doi: 10.1016/j.websem.2003.07.001. URL `http://dx.doi.org/10.1016/j.websem.2003.07.001`.

[70] R. Hull. Managing semantic heterogeneity in databases: A theoretical prospective. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS '97, pages 51–61, New York, NY, USA, 1997. ACM. ISBN 0-89791-910-6.

[71] U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In L. P. Kaelbling and A. Saffiotti, editors, *IJCAI*, pages 466–471. Professional Book Center, 2005. ISBN 0938075934.

[72] U. Hustadt, B. Motik, and U. Sattler. Reasoning in description logics by a reduction to disjunctive datalog. *J. Autom. Reasoning*, 39(3), 2007.

[73] Y. A. Ibáñez-García. Finite model reasoning in *DL-Lite* with cardinality constraints. In *Proc. of the 25th Int. Workshop on Description Logics (DL 2012)*, 2012.

[74] Y. A. Ibáñez-García, C. Lutz, and T. Schneider. Finite model reasoning in Horn-$\mathcal{SHIQ}$. In T. Eiter, B. Glimm, Y. Kazakov, and M. Krötzsch, editors, *Proc. of the 26th Int. Workshop on Description Logics (DL 2013)*, volume 1014 of *CEUR Electronic Workshop Proceedings, http://ceur-ws.org/*, pages 234–245. CEUR-WS.org, 2013.

[75] Y. A. Ibáñez-García, C. Lutz, and T. Schneider. Finite model reasoning in Horn-$\mathcal{ALCQI}$. In *Proc. of the 14th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2014)*. "AAAI Press", 2014.

[76] T. Imielinski and W. L. Jr. Incomplete information in relational databases. *J. ACM*, 31 (4):761–791, 1984.

[77] D. S. Johnson and A. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. In *Proceedings of the 1st ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, PODS '82, pages 164–169, New York, NY, USA, 1982. ACM. ISBN 0-89791-070-2. doi: 10.1145/588111.588138.

[78] K. Kaneiwa and K. Satoh. Consistency checking algorithms for restricted UML class diagrams. In *Proceedings of the 4th International Conference on Foundations of Information and Knowledge Systems*, FoIKS'06, pages 219–239, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-31782-1, 978-3-540-31782-1.

[79] Y. Kazakov. Consequence-driven reasoning for Horn-$\mathcal{SHIQ}$ ontologies. In *Proc. IJCAI-09*, pages 2040–2045, 2009.

[80] Y. Kazakov, M. Krötzsch, and F. Simančík. Concurrent classification of $\mathcal{EL}$ ontologies. In *Proc. ISWC-11 (1)*, volume 7031 of *LNCS*, pages 305–320. Springer, 2011. ISBN 978-3-642-25072-9.

[81] S. Kikot, R. Kontchakov, and M. Zakharyaschev. Conjunctive query answering with OWL 2 QL. In Brewka et al. [25]. ISBN 978-1-57735-560-1.

[82] T. Klenke. Über die Entscheidbarkeit von konjunktiv Anfragen mit Ungleichheit in der Beschreibungslogik $\mathcal{EL}$. Master's thesis, Universität Bremen, 2010.

[83] A. Klug. On conjunctive queries containing inequalities. *J. ACM*, 35(1):146–160, 1988.

[84] P. Kogut, S. Cranefield, L. Hart, M. Dutra, K. Baclawski, M. Kokar, and J. Smith. UML for ontology development. *The Knowledge Engineering Review*, 17(01):61–64, 2002.

[85] R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev. The combined approach to query answering in *DL-Lite*. In *Proc. of the 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*. AAAI Press, 2010.

[86] R. Kontchakov, F. Wolter, and M. Zakharyaschev. Logic-based ontology comparison and module extraction with an application to *DL-Lite*. *AIJ*, 174(15):1093–1141, 2010.

[87] D. Kozen and J. Tiuryn. Logics of programs. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 789–840. North Holland, Amsterdam, 1990.

[88] A. Krisnadhi and C. Lutz. Data complexity in the $\mathcal{EL}$ family of DLs. In *Proc. DL-07*, volume 250 of *CEUR-WS.org*, 2007.

[89] M. Lenzerini. Data integration: A theoretical perspective. In L. Popa, S. Abiteboul, and P. G. Kolaitis, editors, *PODS*, pages 233–246. ACM, 2002. ISBN 1-58113-507-6.

[90] M. Lenzerini and P. Nobili. On the satisfiability of dependency constraints in entity-relationship schemata. *Information Systems*, 15(4):453–461, 1990.

[91] L. Libkin. Expressive power of SQL. *Theor. Comput. Sci.*, 296(3):379–404, 2003. doi: 10.1016/S0304-3975(02)00736-3. URL `http://dx.doi.org/10.1016/S0304-3975(02)00736-3`.

[92] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Comput. Surv.*, 22(3):267–293, Sept. 1990. ISSN 0360-0300.

[93] C. Lutz. The complexity of conjunctive query answering in expressive description logics. In A. Armando, P. Baumgartner, and G. Dowek, editors, *IJCAR*, volume 5195 of *Lecture Notes in Computer Science*, pages 179–193. Springer, 2008. ISBN 978-3-540-71069-1.

[94] C. Lutz, U. Sattler, and L. Tendera. The complexity of finite model reasoning in description logics. *Information and Computation*, 199:132–171, 2005.

[95] C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 09)*, pages 2070–2075, 2009.

[96] A. Madry. Data exchange: On the complexity of answering queries with inequalities. *Inf. Process. Lett.*, 94(6):253–257, 2005.

[97] D. Maier, A. O. Mendelzon, and Y. Sagiv. Testing implications of data dependencies (abstract). In P. A. Bernstein, editor, *SIGMOD Conference*, page 152. ACM, 1979. ISBN 0-89791-001-X.

[98] R. Meyden. Logical approaches to incomplete information: A survey. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, volume 436 of *The Springer International Series in Engineering and Computer Science*, pages 307–356. Springer US, 1998. ISBN 978-1-4613-7582-1. doi: 10.1007/978-1-4615-5643-5_10.

[99] M. Minsky. A framework for representing knowledge. In Brachman and Levesque [24], pages 245–262. ISBN 093461301X.

[100] M. Mosurovic, N. Krdzavac, H. Graves, and M. Zakharyaschev. A decidable extension of $\mathcal{SROIQ}$ with complex role chains and unions. *J. Artif. Intell. Res. (JAIR)*, 47:809–851, 2013.

[101] D. Nardi and R. J. Brachman. An introduction to description logics. In Baader et al. [12], pages 1–40.

[102] Object Management Group, editor. *UML 2 Object Constraint Language Specification*. Object Management Group (OMG), 2006. URL `http://www.omg.org/spec/OCL/ISO/19507/PDF`.

[103] M. Ortiz, D. Calvanese, and T. Eiter. Characterizing data complexity for conjunctive query answering in expressive description logics. In *AAAI*, pages 275–280. AAAI Press, 2006.

[104] M. Ortiz, S. Rudolph, and M. Šimkus. Query answering in the Horn fragments of the description logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. In *Proc. IJCAI-11*, pages 1039–1044. IJCAI/AAAI, 2011.

[105] M. Otto. Modal and guarded characterisation theorems over finite transition systems. *Ann. Pure Appl. Logic*, 130(1-3):173–205, 2004.

[106] M. Otto. Highly acyclic groups, hypergraph covers, and the guarded fragment. *J. ACM*, 59(1):5, 2012.

[107] C. H. Papadimitriou. *Computational complexity*. Academic Internet Publ., 2007. ISBN 978-1-4288-1409-7.

[108] B. Piza, K.-D. Schewe, and J. W. Schmidt. Term subsumption with type constructors. In *Proc. of the Int. Conf. on Information and Knowledge Management (CIKM-92)*, pages 449–456. Citeseer, 1992.

[109] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.

[110] I. Pratt-Hartmann. Complexity of the two-variable fragment with (binary-coded) counting quantifiers. *CoRR*, cs.LO/0411031, 2004.

[111] I. Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *J. of Logic, Language and Information*, 14(3):369–395, 2005.

[112] I. Pratt-Hartmann. Data-complexity of the two-variable fragment with counting quantifiers. *Inf. Comput.*, 207(8):867–888, 2009.

[113] M. R. Quillian. Word concepts: A theory and simulation of some basic semantic capabilities. In Brachman and Levesque [24], pages 97–118. ISBN 093461301X.

[114] M. Rodriguez-Muro, R. Kontchakov, and M. Zakharyaschev. Ontology-based data access: Ontop of databases. In *The Semantic Web - (ISWC 2013) - 12th International Semantic Web Conference*, pages 558–573, 2013.

[115] R. Rosati. On the decidability and finite controllability of query processing in databases with incomplete information. In S. Vansummeren, editor, *Proc. of PODS*, pages 356–365. ACM, 2006. ISBN 1-59593-318-2.

[116] R. Rosati. The limits of querying ontologies. In *Proc. of the 11th Int. Conf. on Database Theory (ICDT)*, volume 4353 of *LNCS*, pages 164–178. Springer, 2007. ISBN 3-540-69269-X.

[117] R. Rosati. Finite model reasoning in *DL-Lite*. In *Proceedings of the Fifth European Semantic Web Conference (ESWC 2008)*, volume 5021 of *Lecture Notes in Computer Science*, pages 215–229. Springer, 2008.

[118] R. Rosati. On the finite controllability of conjunctive query answering in databases under open-world assumption. *J. Comput. Syst. Sci.*, 77(3):572–594, 2011.

[119] B. Rossman. Homomorphism preservation theorems. *J. ACM*, 55(3), 2008. doi: 10.1145/1379759.1379763. URL `http://doi.acm.org/10.1145/1379759.1379763`.

[120] A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *Journal of Intelligent Information Systems*, 2(3):265–278, 1993.

[121] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'91, pages 466–471, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc. ISBN 1-55860-160-0.

[122] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artif. Intell.*, 48(1):1–26, 1991.

[123] A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv.*, 22(3):183–236, Sept. 1990. ISSN 0360-0300.

[124] B. Thalheim. Extended entity-relationship model. In L. Liu and M. T. Özsu, editors, *Encyclopedia of Database Systems*, pages 1083–1091. Springer US, 2009. ISBN 978-0-387-35544-3, 978-0-387-39940-9.

[125] G. Thomas, G. R. Thompson, C.-W. Chung, E. Barkmeyer, F. Carter, M. Templeton, S. Fox, and B. Hartman. Heterogeneous distributed database systems for production use. *ACM Comput. Surv.*, 22(3):237–266, Sept. 1990. ISSN 0360-0300.

[126] M. Thomazo, J. Baget, M. Mugnier, and S. Rudolph. A generic querying algorithm for greedy sets of existential rules. In Brewka et al. [25]. ISBN 978-1-57735-560-1. URL `http://www.aaai.org/ocs/index.php/KR/KR12/paper/view/4542`.

[127] R. van der Meyden. The complexity of querying indefinite data about linearly ordered domains. *J. Comput. Syst. Sci.*, 54(1):113–135, 1997.

[128] M. Y. Vardi. The complexity of relational query languages (extended abstract). In *Proc. of 14th Annual ACM Symposium on Theory of Computing (STOC)*, pages 137–146. ACM, 1982.

[129] M. Y. Vardi. On the integrity of databases with incomplete information. In *Proceedings of the Fifth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, March 24-26, 1986, Cambridge, Massachusetts*, pages 252–266. ACM, 1986. ISBN 0-89791-179-2.

[130] W3C OWL Working Group, editor. *OWL 2 Web Ontology Language: Document Overview– W3C Recommendation*. W3C, 2nd edition, 11 December 2012. Available at `http://www.w3.org/TR/2012/REC-owl2-overview-20121211/`.