FINE-GRAINED FORGETTING FOR EXPRESSIVE DESCRIPTION LOGICS: DEDUCTIVE, SEMANTIC, AND QUERY FORGETTING IN ONE FRAMEWORK

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF SCIENCE AND ENGINEERING

2024

Mostafa Sakr

Department of Computer Science

Contents

Word Count: 78897

List of Tables

List of Figures

Abstract

FINE-GRAINED FORGETTING FOR EXPRESSIVE DESCRIPTION LOGICS: DEDUCTIVE, SEMANTIC, AND QUERY FORGETTING IN ONE FRAMEWORK

Mostafa Sakr

A thesis submitted to The University of Manchester
for the degree of Doctor of Philosophy, 2024

The aim of this thesis is to study different notions of forgetting in the context of the description logic $\mathcal{ALC}$. Forgetting is an engineering task that eliminates subsets of the vocabulary from an ontology, and preserves the information relative to the remainder of the vocabulary. Forgetting offers solutions to many important applications such as ontology creation, reuse, debugging, information hiding, abduction, and agent communication. The main contribution of this thesis is a fine-grained forgetting framework, that is capable of forgetting concept names from $\mathcal{ALC}$ ontologies, and obtaining representations of the deductive, query and semantic forgetting views. The framework also computes sets of axioms that indicate the differences between the contents of these forgetting views, which allows for a new understanding of the different forms of forgetting and the relationships between them. Moreover, we develop an algorithm whereby the content of the forgetting views can be enhanced with some of these axioms according to user requirements in a way that reveals a spectrum of fine-grained forgetting views in-between the deductive, the query, and the semantic forgetting views. These views can be used to address more applications and use cases than currently anticipated. Another contribution of this thesis is a semantic forgetting method for $\mathcal{ALC}$ ontologies that preserves the syntactic structure of the input ontology, and improves the readability of the forgetting view. The deductive customization of the forgetting framework

has been evaluated against the forgetting system Lethe in six different evaluations. In most experiments our deductive customization successfully terminated in less than half of the time consumed by Lethe, despite computing more information. The semantic forgetting method was evaluated on its own, and in comparison with the semantic forgetting system Fame. In two evaluations out of three, our semantic forgetting method terminated successfully in less than 42 % of the time consumed by Fame.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made **only** in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see `http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420`), in any relevant Thesis restriction declarations deposited in the University Library, The University Library's regulations (see `http://www.library.manchester.ac.uk/about/regulations/`) and in The University's policy on presentation of Theses

Acknowledgements

First and foremost I would like to thank my supervisor Dr.-Ing. Renate A. Schmidt for her support and guidance throughout this PhD project, for her trust in me, and for her understanding of my personal circumstances.

My sincere thanks to my examiners, professor Cláudia Nalon and Dr. Konstantin Korovin, who turned my examination into a rich and deep discussion and gave me plenty of ideas for future research.

I must thank my colleagues and journey partners, Sen, Ruba, David, Mariam, Kiana, and Martina with whom I shared great moments and treasured memories. Guys, thank you very much for keeping me sane during this journey.

I am deeply grateful to my lifelong friends Mohamed Elmorsy and Muhammad Askar who supported me through every hardship, joined me in every cheer, and remained firmly with me through every life-changing event.

I am forever in debt to my parents who constantly encourage and inspire me, to my brother and sister whose advice has helped me make the right choices and directions.

I am sincerely thankful to my partner Eman who ignited the challenge in me and helped me find my true self. Our small talks were incredible helpful for organizing my thoughts and planning my steps. I am truely lucky to have you by my side.

Lastly, I am grateful to my little daughter Nur who brought joy and determination to family and who actively participated in editing Chapter 5 despite being a few months old. All of this is for you, Nur!

Chapter 1

Introduction

Description logic ontologies provide means to capture and encode the terminological knowledge of a domain. They provide the common vocabulary, allow for encoding the complex concepts built from the vocabulary, and precisely define the interrelations between these concepts in a way that preserves the exact semantics of the knowledge for the domain of interest. Their simple and intuitive way of capturing the knowledge, as well as the ability to capture the precise semantics, has led to the use of description logic ontologies in formalizing complex and large-scale amount of data in the biomedical, medicinal, and other knowledge extensive domains.

One example is the SNOMED CT ontology [SPSW01], a large-scale biomedical ontology developed by the health sectors in several countries. The SNOMED CT ontology provides terminological data for more than 367,000 concept names and over 240 roles, and encodes different biomedical topics such as clinical findings, symptoms, diagnoses, procedures, body structures, organisms, substances, pharmaceuticals, devices, and specimens.

Despite the simplicity provided by description logic ontologies compared to other formalisms, the ever growing complexity of knowledge is mirrored by increased complexity of the ontology. Furthermore, the interconnected nature of knowledge is mirrored by the increase of the size of the ontologies. As a result, developing, maintaining, and even reasoning with ontologies become a rather difficult task.

In real life applications, it is often that small portions of the knowledge are relevant and sufficient for the applications to perform. This accumulates the need for engineering tools that can extract the relevant portion of knowledge to the application of concern. Forgetting is a knowledge extraction process that lends itself to solve this

problem nicely. It takes two inputs: an ontology and a subset of the ontology's vocabulary called *the forgetting signature*. Using reasoning, it computes another ontology that does not use any symbol from the forgetting signature but preserves the information relative to the remainder of the vocabulary. Since reasoning is used, the output ontology preserves not only the explicit knowledge encoded in the input ontology but also the relevant implicit knowledge. We shall hereafter call the output ontology a *forgetting view*- a name imported from the database realm where it denotes the output of a SELECT-FROM-WHERE query.

In this thesis, we study several notions of forgetting and develop a framework for computing, comparing, and customizing forgetting views. The framework addresses gaps in the literature in a way that makes forgetting more powerful and beneficial to users.

In this chapter we briefly discuss the current understanding of forgetting, identify the gaps of the literature and explain the significance of addressing these gaps. We also discuss the work presented in this thesis, and highlight the contributions of the presented work.

1.1 Overview of Forgetting

The topic of concern in this thesis is forgetting for description logic ontologies. Forgetting is an engineering task in which subsets of the vocabulary of an ontology are eliminated and all semantics relative to the remaining vocabulary is preserved.

Although as a term forgetting has been introduced in [LR94a], the origins of forgetting go as far back as to Boole [Boo54] who noted that forgetting irrelevant symbols from a formula can highlight non-trivial and obscure relations between other symbols of concern. Since then, forgetting has been manifested often implicitly in many of the most fundamental problems of knowledge representation and mathematical logic.

One example is the famous Davis-Putnam algorithm, which decides the satisfiability of a first-order theory by essentially forgetting all symbols from it [DP60]. The ideas of the Davis-Putnam algorithm [DP60], and its successor, the DPLL algorithm [DLL62], are still used in modern forgetting methods [GO92, HM08, KS13c, ASS22].

Another example is the problem of second-order quantifier elimination [Ack35, GSS08], which was later considered a notion of forgetting [LR94a]. A predicate symbol can be forgotten from an ontology by quantifying over this symbol, making it

second-order, and then eliminating this second-order quantifier. This correspondence allowed transferring methods and approaches from the realm of second-order quantifier elimination to the wider realm of forgetting [GSS08]. Furthermore, undecidability results [Ack35] of second-order quantifier elimination implied that forgetting views of first-order theories are generally not representable in first-order logic [LR94a].

In the following two decades forgetting for description logics was developed in two directions. The first is concerned with existence of forgetting views for description logics. The second direction, is concerned with the use of forgetting in applications. For the first direction, forgetting views were shown to always exist when forgetting concept names from ontologies in the description logic DL-Lite [WWTP08]. Similar results could not be obtained for ontologies expressed in the description logics $\mathcal{EL}$, $\mathcal{ALC}$, or their extensions. The results of the second direction were more promising. Forgetting was found to offer solutions to many important applications. We briefly highlight some of them below. Deeper discussions of these applications are presented in Chapter 3.

Ontology Creation: Developing new axioms against the relevant portion of an ontology was shown to be more efficient than developing the axioms against the whole ontology. This is because various validations of the new axioms become easier to perform [CAS+19, ASDG21, DSG22, DPSG+23]. Forgetting offers the way to extract the relevant portions of concern [LLM03, LLA+21].

Ontology Reuse: Ontology-based systems often reuse modules of existing ontologies, rather than building their ontologies from scratch [GHKS07, GHKS08, SSZ09, KLWW09, KWZ10, GKW14]. Obvious savings in time and cost would be gained by reusing these modules. For ontologies developed in a monolithic form rather than a modular form, extracting the required modules can be non-trivial. Forgetting allows for extracting precisely the required modules.

Information Hiding: Data privacy is a challenging requirement for systems that encodes confidential information in their ontology or knowledge bases. Examples of such systems are ontology-based medical and financial systems. When a view of the ontology or the knowledge base is shared or queried, confidential information should not be revealed to users with restricted privileges [MS04, Swe02, CG10, CGM10]. One possible solution to the problem is using a forgetting view where the confidential terms have been forgotten.

Updating agent beliefs: The beliefs of an agent, in multi-agent environments, are continuously updated by forgetting old ones and learning new ones [LLM03, DHLM08].

Thus, forgetting is central in regards to erasing old beliefs. Interestingly though, forgetting has also been shown important in learning new beliefs in the multi-agent setting where one agent learns through communications with other agents [TSP22, DŁS01]. When information is taught from one agent to the other, the first agent must express this information over the vocabulary known to the second agent, a process which can be performed by forgetting the symbols that are unknown to the second agent.

The findings of the two directions of research has led to the following interesting question. *If the application that uses the forgetting view is known, can we come up with a weaker, albeit easier to compute, notion of forgetting where only the information sufficient for the application is preserved?* The quest to answer this question has led to new notions of forgetting. Namely, deductive forgetting, and query forgetting [KWW09, LW10, BKL$^+$17]. The classical notion of forgetting is now widely known as semantic forgetting [ZS15, Zha18].

The notion of deductive forgetting (see Definition 3.5.14) coincides with the notion of uniform interpolation [Hen63, GLW06, Vis96, TCMV06, ZZ10]. It preserves all concept subsumption axioms over the non-forgotten vocabulary in the source logic, that is, the logic of the input ontology. It is suitable for applications such as classification, where entailment checking for axioms over the non-forgetting symbols in the source logic is required.

Query forgetting (see Definition 3.5.15) is a new notion whereby query forgetting views coincide with the input ontology in regards to the answers to conjunctive queries against arbitrary ABoxes [KWW09, LW10, WWTP08]. More precisely, suppose $\mathcal{O}$ is an ontology, and F a subset of the symbols occurring in $\mathcal{O}$. The ontology $\mathcal{V}^q$ is a query forgetting view of $\mathcal{O}$ with respect to F, if the symbols from F are not used in $\mathcal{V}^q$, and for any ABox $\mathcal{A}$, the two knowledge bases $(\mathcal{O};\mathcal{A})$ and $(\mathcal{V}^q;\mathcal{A})$ coincide on their answers to any conjunctive query over the symbols from $\mathcal{O}$ that are not in F. Query forgetting is suitable for ontology-based data access applications where databases are enhanced with ontologies [CDL$^+$05, EGOŠ08, RG10, KKL$^+$11, OŠ12, Ort13, BOvX13, BO15, Bie16, EHEVGJ21, Pan21]. A weaker notion of query forgetting concerned only with instance queries rather than conjunctive queries was considered in [KWW09].

Semantic forgetting (see Definition 3.5.16) implies that the set of models of the original ontology coincides with the set of models of forgetting view when only the interpretations of the non-forgotten symbols are considered [LR94a, ZZ10]. Because interpretations of the non-forgotten symbols are preserved, the two ontologies coincide

on all consequences over the non-forgotten symbols in any logic up to second-order logic [WWTP08, BKL$^+$17]. The two ontologies can therefore replace one another in any application where only the consequences over the non-forgotten symbols are required.

The relationship between the three types of forgetting has been studied for different description logics. Semantic forgetting subsumes deductive and query forgetting, because it preserves the interpretations of the non-forgotten symbols, a property that is independent of the description logic considered [BKL$^+$17]. As such, any semantic forgetting view is also a query and a deductive forgetting view. In Horn logics, it was shown that query forgetting subsumes deductive forgetting [KWW09]. In non-Horn logics such as the description logic $\mathcal{ALC}$, the relation between query and deductive forgetting has not be considered so far. We show in this thesis that query forgetting does not subsume deductive forgetting in the context of the description logic $\mathcal{ALC}$.

Deductive forgetting views of $\mathcal{EL}$ ontologies exist, possibly infinitely, in $\mathcal{EL}$. Likewise, the deductive forgetting views of $\mathcal{ALC}$ ontologies exist, possibly infinitely, in $\mathcal{ALC}$ [KWW09, LW11, KS13c]. Several deductive forgetting methods have been developed for ontologies in extensions of the description logic $\mathcal{ALC}$ [HM08, LK14, KS13b, KS13c, WWT$^+$14, KS14, KS15a].

In the context of the description logic $\mathcal{EL}$ and its extensions, deductive and query forgetting methods are proposed in [KWW09]. Query forgetting for $\mathcal{ALC}$ ontologies has not been investigated.

Although semantic forgetting views of ontologies in the extensions of $\mathcal{ALC}$ do not exist in general, semantic forgetting methods have been developed for ontologies in extensions of the description logic $\mathcal{ALC}$ [ZS15, ZS16, ZS17, ZS18]. The methods may not always compute the semantic forgetting views, particularly in the cases where they do not exist even in any first-order language.

1.2 Research Gaps

As we see from the previous section, in the modern view of forgetting there are different forms of forgetting. Each one of them is tailored towards certain applications. Forgetting in this view gives a lot of flexibility, and allows for extracting forgetting views with sufficient informativeness for applications. Meeting the requirements of these applications may not have been feasible when only the classic notion of semantic forgetting was considered, because semantic forgetting views may not exist [LR94a].

However, there are gaps in this vision that are not yet addressed. We highlight them in the following.

1. As noted earlier, query forgetting has not been investigated in the context of the description logic $\mathcal{ALC}$ or its extensions. Addressing this gap would allow for optimizing the ontology-based data access applications, where data instances are enhanced with ontologies in the expressive description logic $\mathcal{ALC}$ or its extensions, and where substituting the query forgetting views for the deployed ontologies would reduce the time cost and complexity of query answering. This would complement the ongoing effort of designing efficient query answering methods in the context of expressive description logics [GHLS07, OSE08, OCE08, EOŠ12, CEO14, GBIGJ17].

   Let $\mathcal{O}$ be an ontology, and $\mathcal{F}$ a forgetting signature. An ontology $\mathcal{V}^q$ is a query forgetting view of $\mathcal{O}$ with respect to $\mathcal{F}$, if $\mathcal{V}^q$ uses only the vocabulary from the set S of symbols occurring in $\mathcal{O}$ but not in $\mathcal{F}$, and for every ABox $\mathcal{A}$ over S and conjunctive query $q(\vec{x})$ over S the two knowledge bases $(\mathcal{O};\mathcal{A})$ and $(\mathcal{V}^q;\mathcal{A})$ coincide on their answers to the conjunctive query q.

   The existing $\mathcal{EL}$ query forgetting method [KWW09] cannot be easily extended to $\mathcal{ALC}$. One particular challenge is that the interpretations of the knowledge bases $(\mathcal{O};\mathcal{A})$ and $(\mathcal{V}^q;\mathcal{A})$ are different when $\mathcal{O}$ and $\mathcal{V}^q$ are expressed in $\mathcal{ALC}$ than when they are expressed in $\mathcal{EL}$. When the ontology component of a knowledge base is expressed in $\mathcal{EL}$, or other Horn logic, there is one model of the knowledge base, often called the minimal model, that is representative of all other models of the knowledge base. As such, verifying that $(\mathcal{O};\mathcal{A})$ and $(\mathcal{V}^q;\mathcal{A})$ coincide on their answers to the conjunctive query q can be performed by comparing the minimal model of $(\mathcal{O};\mathcal{A})$ to the minimal model of $(\mathcal{V}^q;\mathcal{A})$. When the ontologies are expressed in $\mathcal{ALC}$ or more expressive logic, each knowledge base can have several minimal models, and verifying coincidence on the answers to conjunctive queries requires comparing all minimal models of one knowledge base to all minimal models of the other knowledge base.

2. Although effort has been made to explore the relations between the different notions of forgetting as mentioned earlier [BKL$^+$17], the differences in content between forgetting views of the three notions are not yet understood. In other words, *what can be the information relative to the non-forgetting symbols that are not preserved by the deductive or the query forgetting view, but would have*

been preserved by the semantic forgetting view? Likewise, what information is preserved by a query forgetting view but not a deductive forgetting view, and vice versa? Addressing this gap allows for controlling the informativeness of the forgetting view in a way relevant to the data rather than the usage of the forgetting view. For example, if the content differences between the semantic and the deductive forgetting views are known, then the deductive forgetting view can be enhanced with some of this information according to the requirements of the user. In this way, we can customize the content of the forgetting views, and obtain a spectrum of fine-grained forgetting views in-between the semantic and the deductive forgetting views. Users can reason over these fine-grained forgetting views in ways that are not limited to the reasoning tasks supported by deductive forgetting and query forgetting. So, another gap that would be addressed by identifying the differences in content between forgetting views is obtaining forgetting views with sufficient content for more diverse reasoning tasks.

The main challenge is the representation of such information. Consider for instance the content difference between the deductive and the semantic forgetting views of $\mathcal{ALC}$ ontologies. The definitions of deductive forgetting and semantic forgetting imply that the two ontologies coincide on all entailed $\mathcal{ALC}$ axioms over the non-forgetting symbols. So, the content difference cannot be represented as $\mathcal{ALC}$ axioms over the non-forgotten symbols. Meanwhile, we would desire a consistent representation for the content difference between semantic and deductive forgetting views, semantic and query forgetting views, and query and deductive forgetting views. The consistent representation of this data would allow for consistent ways of working with it.

3. Suppose we have two systems. One of them (the receiver) is requesting information from the other system (the sender). On the one hand, the sender may decide to publish a forgetting view rather than its entire ontology, for example to hide portions of the knowledge that may contain sensitive information. On the other hand, the type of forgetting is a requirement of the receiver, because it is the system that is going to use the forgetting view. The current vision of forgetting assumes that the usage of the forgetting view is known and communicated to the sender before computing the forgetting view. However, this is not necessarily true in real life. It can be that only partial information about the usage is known to the receiver when the forgetting view is requested, and the complete

list of reasoning tasks is known at a later stage. Or it may be fully known to the receiver when the forgetting view is requested, but cannot be disclosed to the sender for confidentiality reasons. One solution is publishing the semantic forgetting view of the ontology. However, if the semantic forgetting view does not exist, an alternative is publishing an intermediate ontology that does not contain the sensitive information, and can be consumed at a later time by the receiver to extract only the content relevant to the reasoning tasks.

The challenge here is how to express the intermediate ontology in a form that eliminates the forgetting symbols, and is suitable at the same time for further processing to obtain the final forgetting view.

In addition to the above gaps, existing forgetting methods introduced in the context of the description logic $\mathcal{ALC}$ and its extensions can be criticized for the syntactic representation of the computed forgetting view. The forgetting views obtained by these forgetting methods do not preserve the syntactic structure of the input ontologies. Their axioms usually use, from a user perspective, seemingly arbitrary syntactic structure, which reduces the readability and the clarity of the forgetting view, and is not ideal to users bound by strict modeling guidelines on their ontologies. This problem has not been considered in the literature.

1.3 Contributions

The main contribution of this thesis is a fine-grained forgetting framework. It allows for forgetting concept names from $\mathcal{ALC}$ ontologies, and consists of two segregated processes that run one after the other.

The first process is the forgetting process. It takes the input ontology and forgetting signature, and computes an intermediate ontology which does not use any of the forgetting symbols, nevertheless, it preserves all information relative to the remaining symbols of the input ontology. Depending on the input ontology and the forgetting signature, the intermediate ontology may use auxiliary concept symbols, which we refer to as *definers*. In this sense, the intermediate ontology can be seen as an approximation to the semantic forgetting view because it preserves the required information but may do so by using definers. Although the semantic forgetting view may not exist for some input ontologies and forgetting signatures, the intermediate ontology is always computable for any input $\mathcal{ALC}$ ontology and forgetting signature.

The second process in the framework is the customization process. The input to this process is the intermediate ontology computed in the first process, and the output is the final forgetting view. The customization process is responsible for two operations.

1. Extracting a portion of the intermediate ontology which coincides in content with the desired forgetting view.

2. Eliminating the definers from the extracted portion.

We provide two implementations of the customization process. The first implementation is the deductive customization. As mentioned in Section 1.1, deductive forgetting views of $\mathcal{ALC}$ ontologies may be infinite. If a finite deductive forgetting view is feasible, the output of the customization is a deductive forgetting view. If not, the customization computes a finite representation of the deductive forgetting view that uses auxiliary definers. Technically, the representation is not a deductive forgetting view because it uses definers. However, it coincides with the possibly infinite deductive forgetting view on all information relative to the non-forgetting symbols, and can be used in reasoning tasks in the same way as the deductive forgetting view.

The second implementation is the query customization. The customization computes two representations of the query forgetting view. The first represents the query forgetting view in the description logic $\mathcal{ALC}$, and may use definers. The second represents the query forgetting view in the description logic $\mathcal{ALCI}$. We obtain the second representation by eliminating some of the definers which occur in the first representation, a process where at least inverse roles are needed. One of the findings is that a query forgetting view may be infinite in the same way as the deductive forgetting view. If a finite query forgetting view is feasible, the second representation does not contain definers. However, if an infinite representation is required or if it is not practical to compute the finite representation due to time and size complexity, only a subset of the definers will be eliminated in the $\mathcal{ALCI}$ representation. The first representation of the query forgetting view also has the advantage of being a deductive forgetting view. Thus, it can be used in further reasoning tasks.

The representation of the intermediate ontology is designed to simplify the extraction of the desired portion. In the two customizations, extracting the desired portion amounts to extracting a syntactically characterized subset of the intermediate ontology. The axioms of the intermediate ontology which are discarded from this extraction process indicate the content difference between the intermediate ontology, and the extracted portion relative to the non-forgetting symbols. Denote by D the set of axioms

which are not extracted. Since the intermediate ontology coincides with the input ontology on all information relative to the non-forgetting symbols, the D axioms indicate the information entailed by the input ontology but not the extracted portion relative to the non-forgetting symbols.

In the deductive customization, the content of the computed forgetting view, when definers are used, relative to the non-forgetting symbols coincide with the content of the possibly infinite deductive forgetting view. Thus, in this customization, the axioms in D indicate the information entailed by the input ontology but not the deductive forgetting view relative to the non-forgetting symbols.

The set D extracted in the deductive customization can be used to enhance the deductive forgetting view with more information, and obtain a spectrum of fine-grained forgetting views with customized content. The fine-grained forgetting views are more informative than the deductive forgetting view, and at most as informative as the semantic forgetting view. This idea is used in the query customization to obtain the $\mathcal{ALC}$ representation of the query forgetting view.

In the query customization, the $\mathcal{ALC}$ representation is computed by the idea explained in the previous paragraph. In this way, this representation is a fine-grained forgetting view in the spectrum described above, which explains why this representation is also a deductive forgetting view. The set D obtained in this customization indicates the content difference between the input ontology and the $\mathcal{ALC}$ representation of the query forgetting view, it is a subset of the set D obtained by the deductive customization. The difference between the two sets indicates the information preserved in the $\mathcal{ALC}$ representation of the query forgetting view, but not preserved by the deductive forgetting view.

The forgetting framework achieves several benefits and goals.

1. We present a first query forgetting method for $\mathcal{ALC}$ ontologies. Moreover, our method allows query forgetting views to be customized.

2. Having an $\mathcal{ALC}$ representation of the query forgetting view supports users who require their ontologies to be expressed in $\mathcal{ALC}$, and allows using it in systems where only $\mathcal{ALC}$ reasoners are used. The $\mathcal{ALCI}$ representation is targeted to the users who do not want definers in their ontologies.

3. The intermediate ontology can be used as a representation of the semantic forgetting view. Although it uses definers, it contains the same content relative to the non-forgetting symbols, and it is always computable and expressible in $\mathcal{ALC}$.

4. The set D obtained when extracting the deductive forgetting view and the query forgetting view enhances our understanding of the information relative to the non-forgetting symbols that is not preserved by the forgetting views obtained from the two customizations.

5. By enhancing the forgetting views of the two customizations with axioms from the set D, we obtain a spectrum of fine-grained forgetting views that are customized according to user requirements.

6. Developing the forgetting framework in two segregated components allows distributing the intermediate ontology, and performing the customization at a later stage when required. As such, the use case described in the third gap explained in the previous section is addressed.

The time performance of our fine-grained forgetting framework was compared to the performance of the deductive forgetting system Lethe in six different evaluations using real-life ontologies from the NCBO BioPortal [MBP13]. The ontologies that we used were large-scale with an average of 22663 axioms in each, and our forgetting experiments were designed to eliminate up to half of the concept names in each ontology. Furthermore, different selection methods of the forgetting signature were used to test different use cases. In most experiments our deductive customization successfully terminated in less than half of the time consumed by Lethe, despite computing more information. In some experiments this even went down to almost 25% of the time consumed by Lethe.

A second contribution of this thesis is the development of a semantic forgetting method that preserves the syntactic structure of the input ontology. This is the first forgetting method in the literature to consider the syntactic representation of the forgetting view as a requirement. The method uses definers to overcome the non-existence problem of semantic forgetting views of $\mathcal{ALC}$ ontologies that was mentioned in Section 1.1. The performance of our semantic forgetting method was evaluated on its own, and in comparison with the semantic forgetting system Fame [Zha18]. In two evaluations out of three, our semantic forgetting method terminated successfully in less than 42 % of the time consumed by Fame. Moreover, the number of definers was at most 8% of the number of forgetting symbols, suggesting that the definers occurring in the semantic forgetting view, if any, are often significantly fewer than the number of forgetting symbols.

1.4 Structure of the Thesis

This thesis is structured as follows. Chapter 2 gives the required background on the description logics $\mathcal{ALC}$ and $\mathcal{ALCI}$ used in this thesis. The chapter is aimed to readers with little knowledge of description logics. Readers familiar with description logics may skip this chapter, or use it as a reference when needed.

Chapter 3 discusses the background of forgetting in the literature. It gives an overview of theoretical results of forgetting, and explains two state-of-the-art forgetting methods that share ideas with the methods developed in this thesis, and used in the evaluation of our forgetting methods. The chapter groups together the main definitions used in this thesis, and proves some fundamental lemmas that are used throughout the thesis.

Chapter 4 is the first main chapter of the thesis. It builds the forgetting process in the fine-grained forgetting framework, and develops mechanisms for deductive customization. The chapter describes and explains how fine-grained forgetting views can be obtained in the spectrum between the deductive and the semantic forgetting views. The content of this chapter was presented at the Description Logic Workshop in 2022 [SS22].

Chapter 5 develops the second customization. That is, the query customization. It applies ideas from Chapter 4 regarding computing fine-grained forgetting views, and consequently obtains two representations for the query forgetting view.

Chapter 6 develops a semantic forgetting method that aims at preserving the syntactic structure of the input ontology. The chapter also discusses the reasons for not preserving the syntactic structure of input ontologies in the forgetting framework of Chapter 4, as well as other state-of-the-art forgetting methods. The content of this chapter was presented at the conference Frontiers of Combining Systems (FroCoS) in 2021 [SS21].

Chapter 7 provides an empirical evaluation of our fine-grained forgetting framework extended with the deductive customization developed in Chapter 4, as well as an empirical evaluation of the semantic forgetting method developed in Chapter 6.

Finally, Chapter 8 presents the conclusions and findings of our research and discusses the future work.

Chapter 2

Basics of Description Logics

In this chapter we build the necessary knowledge of the description logics $\mathcal{ALC}$ and $\mathcal{ALCI}$, the underpinning languages of all formalisms considered in this Thesis. The discussions in this chapter are tailored to readers with little or novice knowledge of description logics. We define the basic notations of the two description logics, as well as preliminary definitions that we will use in the remainder of the thesis.

The chapter is structured as follows. Section 2.1 gives a brief discussion on description logics and overviews some of the applications founded on top of them.

Section 2.2 explains the syntax and semantics of the description logic $\mathcal{ALC}$. Section 2.3 explains different extensions of the description logic $\mathcal{ALC}$, and defines the syntax and semantics of the description logic $\mathcal{ALCI}$. Section 2.4 describes the notions of axioms, ontologies, and knowledge bases. It also explains properties such as positions of concepts, and polarities. Section 2.5 explains relevant reasoning such as entailment and query answering. Finally, Section 2.6 explores the relationship between the description logics $\mathcal{ALC}$ and $\mathcal{ALCI}$, and other important logics such as first-order logic, and modal logics. The section will also explain the extension of $\mathcal{ALC}$ and $\mathcal{ALCI}$ with fixpoint operators, an extension that, as we will show in the next chapters, can be used to obtain finite representations of forgetting views.

2.1 Knowledge Representation and Description Logics

One important way of Knowledge representation is the use of logical languages. That is, the languages that allow for constructing different logical formalisms [vHLP08]. The word formalism means that if two users have access to the encoded knowledge and an agreement on the semantics of the logical operators of the language in which

the knowledge is encoded, then they will comprehend the exact same meaning from the encoded knowledge. For instance, consider the following statements encoding definitions of rectangles and squares.

$$\mathit{Rectangle} \quad \mathit{ClosedQuadrilateral} \sqcap \mathit{ObjectWithFourRightAngles} \tag{2.1}$$

$$\mathit{Square} \quad \mathit{ClosedQuadrilateral} \sqcap \mathit{ObjectWithFourRightAngles} \sqcap \mathit{Equilateral} \tag{2.2}$$

There is only one meaning to the statement (2.1); every rectangle is a closed quadrilateral, and every rectangle has four right angles. In the same way, (2.2) can only be understood as that every square is a closed quadrilateral, every square has four right angles sides, and every square has equal side lengths. There cannot be other meanings to the two statements.

The word logical in our description means that the encoded information can be reasoned to derive more information that may or may not be obvious in real life. For instance, we can reason over the two statements (2.1) and (2.2), and infer that squares are types of equilateral rectangles.

Logical languages differ on their expressivity, some languages can be more expressive than others. Complex knowledge requires languages with sufficient expressivity to capture the intended meaning. However, the more expressive the language is, the harder it becomes to reason over. It is easy to get in the so-called undecidable languages. That is, the languages for which it is impossible to design an algorithm for deciding whether a particular information does follow from the encoded information or not. For instance, many of the ideas used nowadays in KR languages have been developed in the language KL-ONE [Bra77, BS85, BN07]. KL-ONE, however, turned out to be undecidable [SS89].

One of the most successful KR families is the family of Description Logics (DLs). Not only do they allow for unambiguous representation of knowledge, but also they offer a simplified syntax, and the ability to reason over the encoded knowledge and infer implicit information that is not explicitly obvious. The SNOMED-CT ontology is one important example of the successful journey of description logics in this regard. The *Systematised NOMenclature of MEDicine-Clinical Terms (SNOMED-CT)* ontology is a large scale ontology founded by the joint work of the health care systems in 47 countries [Int23]. SNOMED-CT contains over 360,000 interrelated health care concepts encoding a comprehensive set of terminologies about clinical findings, symptoms, diagnoses, procedures, body structures, organisms, substances, pharmaceuticals,

devices and specimens. It has the purposes of unifying the health care terminologies, and allowing for consistent patient record reporting.

In addition, description logics have been used in developing the fast-growing framework of *Ontology Based Data Access (ODBA)* where information about the database entities, as well as the interrelations between them, is formalized and encoded in a description logic ontology [CGL+18, XCK+18, Pan21]. The context given by the ontology allows for storing only a subset of the data in the database, and retrieving the omitted data by reasoning over the ontology and the incomplete data [KRMZ13, Bie16]. OBDA systems allow developing efficient and cost effective database management systems as only small sets of data would be stored and managed. In a way we contribute to this framework through the work presented in Chapter 5. Our contribution allows for replacing the ontology component of an OBDA system with a compact forgetting view, which optimizes the reasoning, and consequently reduces the time of data retrieval.

The tremendous success of description logics has led the World Wide Web Consortium (W3C) [Con23] to adopt them as basis to the specifications of the Web Ontology Language (OWL). The success of description logics has grown bigger when the W3C issued OWL-2 with three new language profiles based on description logics [HPSvH03, HPS11].

The description logic $\mathcal{ALC}$ is the central logic in the family of description logics. $\mathcal{ALC}$ stands for *Attributive Language with Complements*. It was first introduced in [SSS91]. There are several properties of $\mathcal{ALC}$ which give it its unique status among other description logics.

1. The high expressivity of the description logic $\mathcal{ALC}$ allows encoding complex knowledge of applications. For instance, the Systematised NOMenclature of MEDicine—Clinical Terms (SNOMED CT) ontology- the landmark biomedical ontology, uses only a portion of the $\mathcal{ALC}$ expressivity.

2. All description logics can be driven from $\mathcal{ALC}$ by adding or removing some constructs. So, many of the challenges seen with the description logics are rooted in the description logic $\mathcal{ALC}$.

3. The connection between $\mathcal{ALC}$ and other logics, such as the connection to the two-variable fragment of the first-order logic [Bor96] and the connection to the standard modal logic [Sch91], allowed for transferring many established results and reasoning methods to the description logic $\mathcal{ALC}$.

Building our research in the context of the description logic $\mathcal{ALC}$ was motivated by the first and second properties above. The first property implies that the forgetting methods designed in this thesis are applicable to a substantial number of real-life ontologies. The second property implies that the ideas and methods developed in our research can be extended to cover more expressive languages from the family of the description logic $\mathcal{ALC}$, or simplified to provide computationally optimized solutions for less expressive languages.

2.2 The Description Logic ALC

We present the syntax and the semantics of the description logic $\mathcal{ALC}$.

The most elementary building blocks of description logic ontologies are *concept names* and *roles*. Concept names represent sets of elements in the application knowledge that we want to encode. Roles represent binary relations between elements in the application knowledge. For example we may encode the following.

$$\mathit{Father}(\mathit{John}) \tag{2.3}$$

$$\mathit{Boy}(\mathit{Bob}) \tag{2.4}$$

$$\mathit{isParentOf}(\mathit{John}; \mathit{Bob}) \tag{2.5}$$

The symbol *Father* is a concept name. It groups and represents all fathers in the application knowledge. *John* is one of those fathers. In the same way, *Boy* is a concept name representing all boys, and (2.4) encodes that *Bob* is a boy. The symbol *isParentOf* is a role. It relates two elements together. In 2.5, the role *isParentOf* allows encoding that John is a parent of Bob. The symbols *John* and *Bob* are individuals. In this section we will focus on the syntax and semantics of concepts and roles. Individuals and statements such as those in (2.4) and (2.5) will be defined in section 2.4.

Denote by N_c the set of concept names and by N_r the set of roles. We assume that N_c and N_r are disjoint. Let $\mathsf{S} = N_c \cup N_r$ denote the set of concept names and roles. The set $\mathcal{ALC}(\mathsf{S})$ is the set of $\mathcal{ALC}$ concepts that can be constructed over S. It is defined inductively as follows.

1. The concepts $\top$ and $\bot$ are in $\mathcal{ALC}(\mathsf{S})$.
2. If $A \in N_c$ is a concept name, then $A \in \mathcal{ALC}(\mathsf{S})$.
3. If $C \in \mathcal{ALC}(\mathsf{S})$, then $\neg C \in \mathcal{ALC}(\mathsf{S})$.

4. If $C; D \in \mathcal{ALC}(S)$, then $C \sqcup D \in \mathcal{ALC}(S)$.

5. If $C; D \in \mathcal{ALC}(S)$, then $C \sqcap D \in \mathcal{ALC}(S)$.

6. If $C \in \mathcal{ALC}(S)$ and $r \in N_r$ is a role, then $\exists r.C \in \mathcal{ALC}(S)$.

7. If $C \in \mathcal{ALC}(S)$ and $r \in N_r$ is a role, then $\forall r.C \in \mathcal{ALC}(S)$.

where the symbol $\neg$ is the negation operator, $\sqcup$ is the disjunction operator, $\sqcap$ is the conjunction operator, $\exists$ is the existential role restriction operator, and $\forall$ is the universal role restriction operator. We call the concept C below the role restrictions in $\exists r.C$ and $\forall r.C$ an r filler. Throughout this thesis, we will use capital letters for concepts, and small letters for roles. An *atomic concept* is a concept of the forms defined by 1 and 2 above. A concept C is a *literal* if C is an atomic concept, or the negation of an atomic concept. A literal is *top-level* in an axiom if it does not occur below role restriction in the axiom.

$\mathcal{ALC}$ concepts can be formally understood by means of interpretations. An interpretation I is a pair $(\Delta^I; \cdot^I)$, where Δ^I is a set of elements and $\cdot^I$ is an interpretation function. We call Δ^I the domain of interpretation of I, or domain of interpretation, for short. The interpretation I is an $\mathcal{ALC}$ interpretation over S if $\cdot^I$ maps each concept name in S to a subset of Δ^I, every role in S to a subset of the Cartesian product $\Delta^I \times \Delta^I$, and every concept in $\mathcal{ALC}(S)$ to a subset of Δ^I as follows.

$$\top^I = \Delta^I \tag{2.6}$$
$$\bot^I = \emptyset \tag{2.7}$$
$$(\neg C)^I = \Delta^I \cap C^I \tag{2.8}$$
$$(C \sqcap D)^I = C^I \setminus D^I \tag{2.9}$$
$$(C \sqcup D)^I = C^I \cup D^I \tag{2.10}$$
$$(\exists r.C)^I = \{d \in \Delta^I \mid \exists e \in \Delta^I \text{ where } (d; e) \in r^I \text{ and } e \in C^I\} \tag{2.11}$$
$$(\forall r.C)^I = \{d \in \Delta^I \mid \forall e \in \Delta^I \text{ if } (d; e) \in r^I \text{ then } e \in C^I\} \tag{2.12}$$

We say d is in *the extension of* C *in* I if $d \in C^I$. If I is known from the context, then we drop the notion of I and just say that d is in the extension of C.

The following logical equivalences directly follow from the above semantics.

$$\exists r.\bot \quad \equiv \quad \bot \tag{2.13}$$

$$\forall r.\top \quad \equiv \quad \top \tag{2.14}$$

$$C \sqcup D \quad \equiv \quad \neg(\neg C \sqcap \neg D) \tag{2.15}$$

$$\forall r.C \quad \equiv \quad \neg\exists r.\neg C \tag{2.16}$$

Let us consider some $\mathcal{ALC}$ concepts and interpret them. Consider the following concepts.

$$C_1 = Boy \sqcap Student \tag{2.17}$$

$$C_2 = Student \sqcup \exists isParentOf.Student \tag{2.18}$$

$$C_3 = \forall isParentOf.Student \sqcap \exists isParentOf.\neg Student \tag{2.19}$$

The $\mathcal{ALC}$ concept C_1 in (2.17) represents all boys who are also students. In an interpretation $\mathcal{I}$, C_1 is the intersection of the extensions of Boy and Student. That is, $C_1^{\mathcal{I}} = Boy^{\mathcal{I}} \cap Student^{\mathcal{I}}$. The concept C_2 in (2.18) represents entities which are either students, parents of students, or both. Suppose $\mathcal{I}$ is an interpretation with an interpretation function that has the following mappings.

$$\Delta^{\mathcal{I}} = \{John, Marry, Peter, Bob\} \tag{2.20}$$

$$Student^{\mathcal{I}} = \{Peter, Bob\} \tag{2.21}$$

$$isParentOf^{\mathcal{I}} = \{(Marry, Bob), (Peter, Bob)\} \tag{2.22}$$

Peter and *Bob* are in the extension of C_2 because they are students. Furthermore, *Marry* is in the extension of C_2 because *Marry* is a parent of *Bob* who is a student. *John* is not in the extension of C_2 be cause *John* is not a student, and is not a parent of student. The concept C_3 in (2.19) represents the entities who are only parents of students, and who are parents of non-student entities at the same time. Obviously no entity can have the two criteria at the same time, so the extension of C_3 is the empty set. We can reach the same conclusion through a different method. Let us rewrite the second conjunct of C_3 using the equivalence in (2.16). With this rewrite C_3 would be the following concept.

$$\forall isParentOf.Student \sqcap \neg\forall isParentOf.Student \tag{2.23}$$

In other words, C_3 is equivalent to the conjunction of the concept $\exists isParentOf:Student$ and its negation. Having a domain element in the extension of C_3 means that the element is in the extension of the concept $\exists isParentOf:Student$, and its negation at the same time, which is impossible.

2.3 Extensions of the Description Logic ALC

The description logic $\mathcal{ALC}$ is a basic logic for a wide family of expressive description logics. We use a naming scheme, where extensions of $\mathcal{ALC}$ are obtained by adding appropriate letters or symbols to the word $\mathcal{ALC}$. Each appended letter or symbol indicates that a particular construct or set of constructs are allowed. Table 2.1 shows common extensions of $\mathcal{ALC}$.

The extension $\mathcal{I}$ allows the use of inverse roles. For instance, while $\exists teach:\top$ is an $\mathcal{ALC}$ concept that represents the the set of all domain elements that *teach* something, inverse roles would allow expressing the concept $\exists teach^{-}:\top$ to denote the set of elements which are taught. We will discuss this extension in more details shortly.

The extension $\mathcal{O}$ allows the use of nominals, or singleton concepts. This extension allows encoding information that cannot be encoded in $\mathcal{ALC}$. For instance, the concept $\exists isParentOf:\{John\}$, denotes the parents of John.

The extension $\mathcal{Q}$ allows qualified number restrictions. That is, concepts where the number of role successors of a domain element is restricted. With qualified number restrictions, the parents who have at most two daughters can be represented by the concept $\leq 2 isParentOf:Girl$, and the individuals who published three or more journal papers can be represented by the concept $\geq 3 hasPublication:JournalPaper$.

Related to the extension $\mathcal{Q}$ are the extension $\mathcal{N}$, which denotes unqualified number restriction, or simply number restriction, and the extension $\mathcal{F}$, which denotes functional roles. The extension $\mathcal{N}$ is useful in situations where the role filler is not important, for instance, if we want to represent the parents who are parents of two or less, then we can use the concept $\leq 2 isParentOf$ which has the same meaning as $\leq 2 isParentOf:\top$. The extension $\mathcal{F}$ allows formulas of the form $func(r)$ which indicates that each domain element can have at most one successor through the role r.

Role hierarchy, the extension denoted by $\mathcal{H}$, allows subsumption and equivalence role axioms. We will return back to this extension when the notions axioms and ontologies are defined.

The extensions denoted by $(u);(t);(:)$ allow expressions with role conjunction,

Symbol	Extension	Constructs
I	Inverse role	r^-
O	Nominal	$\{john\}$
Q	Qualified number restriction	$\geq nr.C;\ \leq nr.C$
N	Number restriction	$\geq n.r,\ \leq n.r$
F	Functional role	$func(r)$
H	Role hierarchy	$r \sqsubseteq s$
$(\sqcap)$	Role conjunction	$r \sqcap s$
$(\sqcup)$	Role disjunction	$r \sqcup s$
$(\neg)$	Role negation	$\neg r$
$(\tilde{N})$	Universal role	$\tilde{N}$
S	Transitive role	$trans(r)$
μ	Fixpoints	$\mu X.C[X]; \nu X.C[X]$

Table 2.1: Extensions in the family of the description logic ALC.

disjunction, and negation, respectively. For instance, parents who are also teachers of their own children can be represented by the concept $\exists(isParentOf \sqcap teach).\top$. Note that is is different from the concept $\exists isParentOf.\top \sqcap \exists teach.\top$ which represents the set of parents who are also teachers, but possibly not teachers of their own children.

The universal role extension enhances the vocabulary with the role $\tilde{N}$ which is interpreted in any model I as the set $\Delta^I \times \Delta^I$.

The extension S denotes the extension of ALC with transitive roles. It does not follow the naming scheme of the ALC family. That is, while we write $ALCI$ to denote the extension to ALC with inverse roles, the symbol S on its own (not $ALCS$) represents the ALC extension with transitive roles. The extension is different from the other ones as it defines the S sub-family of the ALC family.

Finally, the extension μ denotes the extension to ALC with fixpoint operators. We explain this extension in detail in Section 2.7.

2.3.1 The Description Logic ALCI

The description logic $ALCI$ extends ALC with inverse roles. The inverse of a role r is r^-, and the inverse of r^- is r. So, we do not allow constructs such as r^{--}. As before, the building blocks of an $ALCI$ ontology are concept names and roles. Let N_c denote a set of concept names, N_r denote the set of role names. We require that if $r \in N_r$, then $r^- \in N_r$.

Let Σ denote the union of N_c and N_r. The set $ALCI(\Sigma)$ of $ALCI$ concepts over Σ

is defined inductively as follows.

1. If $C \in \mathcal{ALC}(\mathsf{S})$ then $C \in \mathcal{ALCI}(\mathsf{S})$.

2. If $C \in \mathcal{ALCI}(\mathsf{S})$ and $r \in \mathsf{N}_r$ is a role, then $\exists r^{-}.C \in \mathcal{ALCI}(\mathsf{S})$.

3. If $C \in \mathcal{ALCI}(\mathsf{S})$ and $r \in \mathsf{N}_r$ is a role, then $\forall r^{-}.C \in \mathcal{ALCI}(\mathsf{S})$.

An interpretation $\mathcal{I}$ is an $\mathcal{ALCI}$ interpretation over S if the function $^{\mathcal{I}}$ maps every concept in S to a subset of $\Delta^{\mathcal{I}}$, every role in S to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, where the mappings from (2.6) to (2.12) are respected, and for every role $r \in \mathsf{S}$, we have the following.

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \tag{2.24}$$
$$\bot^{\mathcal{I}} = \emptyset \tag{2.25}$$
$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \tag{2.26}$$
$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \tag{2.27}$$
$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} \tag{2.28}$$
$$r^{-\mathcal{I}} = \{(y,x) \mid (x,y) \in r^{\mathcal{I}}\} \tag{2.29}$$
$$(\exists r^{-}.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}} \text{ where } (e,d) \in r^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}}\} \tag{2.30}$$
$$(\forall r^{-}.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \forall e \in \Delta^{\mathcal{I}} \text{ if } (e,d) \in r^{\mathcal{I}} \text{ then } e \in C^{\mathcal{I}}\} \tag{2.31}$$

The equivalences from (2.13) to (2.16) remain valid in $\mathcal{ALCI}$.

Consider the following sentences in $\mathcal{ALCI}$.

$$C_1 = \mathit{Student} \sqcap \exists \mathit{isParentOf}^{-}.\neg \mathit{Student} \tag{2.32}$$
$$C_2 = \exists \mathit{isParentOf}^{-}.\mathit{Student} \sqcup \exists \mathit{isParentOf}^{-}.\neg \mathit{Student} \tag{2.33}$$
$$C_3 = \forall \mathit{isParentOf}^{-}.\mathit{Student} \sqcap \exists \mathit{isParentOf}^{-}.\neg \mathit{Student} \tag{2.34}$$

The $\mathcal{ALCI}$ concept C_1 in (2.32) represents all individuals who are students, and whose parents are student. The concept C_2 in (2.33) represents the individuals whose children are students, or whose parents are students, or both. The concept C_3 in (2.34) represents the individuals whose children, if any, are students, and whose parents are students. Suppose $\mathcal{I}$ is an interpretation whose interpretation function has the following

mappings.

$$\Delta^{\mathcal{I}} = \{Marry, Peter, Bob\} \tag{2.35}$$

$$Student^{\mathcal{I}} = \{Peter, Bob\} \tag{2.36}$$

$$isParentOf^{\mathcal{I}} = \{(Marry, Bob), (Peter, Bob)\} \tag{2.37}$$

Peter is in the extension of C_1 because *Peter* is a student, and a parent of a student, *Bob*. We also have *Peter* and *Marry* are in the extension of C_2 because they are parents of *Bob* who is a student. At the same time, *Bob* is in the extension of C_2 because his parent, *Peter* is a student. We can also see that *Bob* is in the extension of C_3, because his parent, *Peter*, is a student. *Marry* and *Peter* are not in the extension of C_3, because their parents are not known in the interpretation.

2.4 Ontologies and Knowledge Bases

An *ontology*, or a *TBox*, over Σ is a set of *axioms*, which are formulas over *concepts*. Let $\mathcal{L}$ be a description logic. We say that an axiom α is an $\mathcal{L}$ *subsumption* axiom over Σ if it has the form $C \sqsubseteq D$ and $C, D \in \mathcal{L}(\Sigma)$, where $\mathcal{L}(\Sigma)$ is the set of all concepts over Σ in $\mathcal{L}$. In this case, we say that D *subsumes* C, and C is a *subset* of D. The axiom α is said to be an $\mathcal{L}$ *equivalence* axiom over Σ if it has the form $C \equiv D$, and $C, D \in \mathcal{L}(\Sigma)$. In this case, we say that C is equivalent to D, or C and D are equivalent. If the set Σ, and the logic $\mathcal{L}$ are known from the context, then we drop their notions and just say that α is a subsumption axiom, or α is an equivalence axiom. Note that $C \equiv D$ is a shorthand for the two subsumption axioms $C \sqsubseteq D$ and $D \sqsubseteq C$.

We say that an axiom $C \sqsubseteq D$ is true in an interpretation $\mathcal{I}$, in symbols $\mathcal{I} \models C \sqsubseteq D$, if and only if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and $C \equiv D$ is true in $\mathcal{I}$, in symbols $\mathcal{I} \models C \equiv D$, if and only if $C^{\mathcal{I}} = D^{\mathcal{I}}$. The interpretation $\mathcal{I}$ is said to be a model of an ontology $\mathcal{O}$, in symbols $\mathcal{I} \models \mathcal{O}$, if $\mathcal{I} \models \alpha$ for every axiom $\alpha \in \mathcal{O}$.

Following the above definitions we can construct the following ontology encoding knowledge about families.

$$Father \sqcup Mother \sqsubseteq \exists isParentOf.Child \tag{2.38}$$

$$Father \sqcap Mother \sqsubseteq \bot \tag{2.39}$$

$$Child \sqsubseteq \exists isParentOf^{-}.Father \sqcap \exists isParentOf^{-}.Mother \tag{2.40}$$

The axiom (2.38) says that every father and every mother is a parent of a child. The axiom (2.39) says that no father is a mother, and no mother is a father. The axiom (2.40) says that each child must have a father and a mother. Note that (2.38) and (2.39) are $\mathcal{ALC}$ axioms, whereas (2.40) is an $\mathcal{ALCI}$ axiom.

The word TBox stands for *Terminology Box*, it contains *terminological* knowledge. This knowledge is global in the sense that it applies to sets of objects. For instance, the axiom (2.38) encode information about every father and every mother, but not a particular father or a particular mother.

Information about certain individuals is encoded in *Assertion Box*, or *ABox* for short. Let N_i be a set of symbols, such that $N_i \setminus N_c = \emptyset$ and $N_i \setminus N_r = \emptyset$. An ABox over S is a set of assertions on the following forms.

$$A(a) \tag{2.41}$$

$$r(a,b) \tag{2.42}$$

where $A \in \mathsf{S}$ is a concept name, $r \in \mathsf{S}$ is a role, and $a,b \in N_i$ are individuals.

The semantics of an ABox $\mathcal{A}$ is defined as follows. Let $\mathcal{I}$ be an interpretation over S as before. The interpretation $\mathcal{I}$ is a model of the ABox, in symbols $\mathcal{I} \models \mathcal{A}$, if $^{\mathcal{I}}$ maps every individual in N_i to a domain element in $\mathsf{D}^{\mathcal{I}}$, and the following are true.

$$A(a) \in \mathcal{A} \quad \text{implies} \quad a^{\mathcal{I}} \in A^{\mathcal{I}} \tag{2.43}$$

$$r(a,b) \in \mathcal{A} \quad \text{implies} \quad (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}} \tag{2.44}$$

A *knowledge base* over S is a pair $(\mathcal{O},\mathcal{A})$ consisting of an ontology $\mathcal{O}$ over S and an ABox $\mathcal{A}$ over S. An interpretation $\mathcal{I}$ is a model of $(\mathcal{O},\mathcal{A})$ if $\mathcal{I}$ is a model of $\mathcal{O}$ and $\mathcal{I}$ is a model of $\mathcal{A}$.

We defined only TBoxes and ABoxes. However, some resources in the literature may also include RBoxes [KS15a, KS15b]. An RBox is a set of axioms of the following forms:

1. Subsumption axioms $r \sqsubseteq s$,
2. Equivalence axioms $r \equiv s$,
3. Transitivity axioms $\mathit{trans}(r)$, and
4. Functional role axioms $\mathit{func}(r)$,

where $r, s \in N_r$ are roles. The first two forms are only permitted in the $\mathcal{H}$ extension, for instance $\mathcal{ALCH}$ or $\mathcal{ALCIH}$, the extensions of $\mathcal{ALC}$ and $\mathcal{ALCI}$, respectively, with role hierarchy. The third form is permitted only in the description logic $\mathcal{S}$ or its extensions. The fourth form is permitted only in the $\mathcal{F}$ extension. Since none of these extensions will be considered, RBoxes will not be used in this thesis.

Let $\mathcal{O}$ be an ontology, and $\alpha \in \mathcal{O}$ an axiom. We say α is *satisfiable* with respect to $\mathcal{O}$ if there is a model $\mathcal{I}$ of $\mathcal{O}$, where $\mathcal{I} \models \alpha$. We say α is *entailed* by $\mathcal{O}$, in symbols $\mathcal{O} \models \alpha$, if $\mathcal{I} \models \alpha$ for every model $\mathcal{I}$ of $\mathcal{O}$. For example, consider the ontology consisting of the two axioms (2.38) and (2.39), and let α be the following axiom.

$$\mathit{Mother} \sqsubseteq \forall \mathit{isParentOf}.\mathit{Child} \tag{2.45}$$

To show that α is satisfiable with respect to $\mathcal{O}$, we construct a model $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}_1})$ with $\cdot^{\mathcal{I}_1}$ defined as follows.

$$\Delta^{\mathcal{I}_1} = \{\mathit{Marry}, \mathit{Bob}\} \tag{2.46}$$
$$\mathit{Mother}^{\mathcal{I}_1} = \{\mathit{Marry}\} \tag{2.47}$$
$$\mathit{Child}^{\mathcal{I}_1} = \{\mathit{Bob}\} \tag{2.48}$$
$$\mathit{isParentOf}^{\mathcal{I}_1} = \{(\mathit{Marry}, \mathit{Bob})\} \tag{2.49}$$

We can see that $\mathcal{I}_1$ is a model of $\mathcal{O}$ because the two axioms (2.38) and (2.39) are true in $\mathcal{I}$. The axiom (2.45) is also true in $\mathcal{I}$, because Marry is a parent of Bob, and Bob is a child. So (2.45) is satisfiable with respect to $\mathcal{O}$.

The axiom (2.45), however, is not entailed by $\mathcal{O}$ because the following interpretation $\mathcal{I}_2$ is a model of $\mathcal{O}$, and (2.45) is false in $\mathcal{I}_2$.

$$\Delta^{\mathcal{I}_2} = \{\mathit{Marry}, \mathit{Bob}, \mathit{Dan}\} \tag{2.50}$$
$$\mathit{Mother}^{\mathcal{I}_2} = \{\mathit{Marry}\} \tag{2.51}$$
$$\mathit{Child}^{\mathcal{I}_2} = \{\mathit{Bob}\} \tag{2.52}$$
$$\mathit{isParentOf}^{\mathcal{I}_2} = \{(\mathit{Marry}, \mathit{Bob}), (\mathit{Marry}, \mathit{Dan})\} \tag{2.53}$$

The axiom (2.45) is false in $\mathcal{I}_2$ because Marry is a parent of Dan, and Dan is not a child.

We say α is *valid*, in symbols $\models \alpha$, if it is entailed by every ontology. In particular, α is valid if it is entailed by the empty ontology. The axiom $A \sqsubseteq A$, for example, is

valid, because it is true in every ontology. An ontology $\mathcal{O}$ is *satisfiable* if it has a model $\mathcal{I}$, otherwise it is *unsatisfiable*. Satisfiability and entailment are PSPACE-COMPLETE and EXPTIME-COMPLETE in $\mathcal{ALC}$ respectively [SSS91].

The *signature* of a concept C is the set of concept names and roles used in the concept. Likewise, the *signature* of an axiom is the set of concept names and roles used in the axiom, and the *signature* of an ontology is the concept names and roles used in the ontology. For an ABox, the *signature* is the concept names and roles occurring in the ABox. Individuals are not considered part of the ABox signature. We use the function symbol *sig* to denote the signature, for instance $sig(C)$ is the signature of the concept C. The individuals occurring in an ABox $\mathcal{A}$, is denoted by the function $ind(\mathcal{A})$.

We refer to concepts within axioms by their positions. Let $\alpha = C \quad D$ be an axiom where $\quad \in \{\sqsubseteq, \equiv\}$, then the positions of C and D relative to α are 1 and 2 respectively, in symbols: $\alpha|1 = C$ and $\alpha|2 = D$.

Consider a concept γ that occurs in the axiom α. Let $pos(\gamma)$ be the set of positions of the concepts occurring in γ. A concept ϕ in γ is referred to by $\alpha|i:p$ where i is position of γ relative to the axiom α and $p \in pos(\gamma)$ is the position of ϕ relative to γ. The definition of $pos(\gamma)$ is given inductively as follows:

1. $i:p \in pos(\gamma)$, if $\gamma = \phi_1 \quad \phi_2 \qquad \phi_n$ where $\quad \in \{\sqcap, \sqcup\}$, $p \in pos(\phi_i)$ and $1 \quad i \quad n$.

2. $1:p \in pos(\gamma)$, if γ takes any of the forms: $\neg\phi$, $\exists r.\phi$, or $\forall r.\phi$ and $p \in pos(\phi)$.

Example 2.4.1. *Let* α *be the following axiom.*

$$A \sqcap \exists r.B \sqsubseteq \neg D \sqcap (C_1 \sqcup \forall r.C_2) \tag{2.54}$$

The following table lists the subconcepts of α *and their positions relative to* α.

Concept	*Position*	*Concept*	*Position*
$A \sqcap \exists r.B$	$\alpha\|1$	$\neg D \sqcap (C_1 \sqcup \forall r.C_2)$	$\alpha\|2$
A	$\alpha\|1:1$	$\exists r.B$	$\alpha\|1:2$
$\neg D$	$\alpha\|2:1$	$C_1 \sqcup \forall r.C_2$	$\alpha\|2:2$
B	$\alpha\|1:2:1$	D	$\alpha\|2:1:1$
C_1	$\alpha\|2:2:1$	$\forall r.C_2$	$\alpha\|2:2:2$
$\exists r.C_2$	$\alpha\|2:2:2:1$	C_2	$\alpha\|2:2:2:1:1$

We write $\alpha[i{:=}\gamma]$ to denote the axiom generated by replacing the concept ϕ at position i relative to α with the concept γ.

The *polarity* of a concept occurring at position i in a subsumption axiom α is denoted by $pol(\alpha, i)$, where $pol(\alpha, \epsilon) = 1$, $pol(\alpha, 2) = 1$, and:

1. $pol(\alpha, i{:}\rho) = pol(\alpha, i)$ if $\alpha|_i$ is a conjunction, disjunction, or a concept of the forms $\exists r.C$ or $\forall r.C$.

2. $pol(\alpha, i{:}\rho) = -pol(\alpha, i)$ if $\alpha|_i$ is a concept of the form $\neg C$.

A concept is *positive* in an axiom if it occurs with polarity 1, and *negative* if it occurs with polarity -1.

Example 2.4.2. *Consider the following axiom* α *from Example 2.4.1. The following table lists the polarity of the concepts in* α *and their positions.*

Position	*Polarity*	*Position*	*Polarity*
$\alpha\|_1$	1	$\alpha\|_2$	1
$\alpha\|_{1:1}$	1	$\alpha\|_{1:2}$	1
$\alpha\|_{2:1}$	1	$\alpha\|_{2:2}$	1
$\alpha\|_{1:2:1}$	1	$\alpha\|_{2:1:1}$	1
$\alpha\|_{2:2:1}$	1	$\alpha\|_{2:2:2}$	1
$\alpha\|_{2:2:2:1}$	1	$\alpha\|_{2:2:2:1:1}$	1

2.5 Query Answering

Query answering is an important reasoning task that we consider in Chapter 5. We define it here in detail.

Let $\mathcal{K} = (\mathcal{O}, \mathcal{A})$ be a knowledge base. A conjunctive query $q(\vec{x})$ is a first order formula of the following form

$$\exists \vec{y}.\, \phi(\vec{x}, \vec{y}), \tag{2.55}$$

where $\vec{x} = x_1, x_2, \ldots, x_k$ and $\vec{y} = y_1, y_2, \ldots, y_m$ are two finite sets of variables with $k \geq 0$ and $m \geq 0$, ϕ is a conjunction of atoms of the forms $A(u)$ and $r(u, v)$, $A \in \mathsf{N}_C$ is a concept name, $r \in \mathsf{N}_R$ is a role, and $u, v \in \mathsf{N}_I \setminus \vec{x} \setminus \vec{y}$.

Assume $John\ 2\ N_i$ is an individual. The following two queries are conjunctive queries.

$$q_1 = 9y{:}isParentOf(John; y) \wedge Student(y) \quad (2.56)$$

$$q_2(x_1; x_2) = isParentOf(x_1; John) \wedge isParentOf(John; x_2) \quad (2.57)$$

The query q_1 represents the question *is John a parent of a student?*, and q_2 is asking about all pairs $(x_1; x_2)$ of individuals such that x_1 is a parent of John, and x_2 is a child of John.

We call $\vec{x}$ the *answer variables*. If q does not have answer variables then q is called a Boolean conjunctive query, or Boolean query for short. For instance, q_1 in (2.56) is Boolean query, and $x_1; x_2$ are the answer variables of q_2 in (2.57). An answer to a query $q(\vec{x})$ in the knowledge base K is $\vec{a} = a_1; a_2; \ldots; a_k$ with the same cardinality as $\vec{x}$ such that the individuals $a_1; \ldots; a_k$ occur in the ABox A, and $K \models q(\vec{a})$. That is, for every I such that $I \models K$ we have $I \models q(\vec{a})$ (we will define this very shortly). An answer to a Boolean conjunctive query is *yes* if $I \models q$ for every model I of K, and *no* otherwise. An answer $\vec{a}$ to a conjunctive query $q(\vec{x})$ is true in a model I of a knowledge base K, in symbols $I \models q(\vec{a})$, if there is a homomorphism π such that the following conditions are true.

1. If x_j is the j-th element in $\vec{x}$, and a_j is the j-th element in $\vec{a}$, then $\pi(x_j) = a_j^I$;
2. If $A(u)$ is an atom in q, then $\pi(u) \in A^I$; and
3. If $r(u; v)$ is an atom in q, then $(\pi(u); \pi(v)) \in r^I$.

The signature of a query $q(x)$, in symbols $sig(q)$ is the set of concept names and roles used in q. That is, individuals and variables are not part of the signature.

Query answering with respect to a knowledge base K is the problem of retrieving all answers to a given conjunctive query q against the knowledge base K.

Consider the ontology O comprising the following axioms.

$$Mother \sqsubseteq 8isParentOf{:}Child \quad (2.58)$$

$$9isParentOf{:}\top \sqsubseteq Mother \quad (2.59)$$

Let A be the following ABox.

$$isParentOf(Marry; John) \quad (2.60)$$

Consider the following conjunctive queries.

$$q_1 = \exists y.\mathit{Mother}(y) \tag{2.61}$$

$$q_2(x) = \mathit{isParentOf}(\mathit{Marry}, x) \wedge \mathit{Child}(x) \tag{2.62}$$

q_1 is a Boolean conjunctive query. It checks if there is an individual in the knowledge base who is a mother. q_2 is a conjunctive query, which asks for the children of Marry.

In every model of the knowledge base $\mathcal{K} = (\mathcal{O}, \mathcal{A})$, we know that Marry is a parent of John. We also know from (2.59) that the first element in any tuple in the interpretation of *isParentOf* is a *Mother*. Therefore, there is a homomorphism that maps y from q_1 to $\mathit{Marry}^{\mathcal{I}}$ in any model $\mathcal{I}$ of $\mathcal{K}$.

For q_2, we know from above that Marry is a mother, and we know from (2.58) that mothers are parents to children only. So John must be a child. More precisely, the homomorphism that maps x from q_2 to $\mathit{John}^{\mathcal{I}}$ is valid for any model $\mathcal{I}$ of $\mathcal{K}$. Thus, John is the answer to q_2.

2.6 Translation to First-Order logic

Another way of interpreting a description logic ontology is translating it to a first-order formulas. Description logics are decidable fragments of first-order logic [BCM$^+$07]. The description logic $\mathcal{ALC}$ coincides with the first-order fragment of two variables and one free variable [Bor96].

The translation defined below of $\mathcal{ALC}$ concepts and axioms to first-order formulas is standard [GHS04, BCM$^+$07]. Let C be an $\mathcal{ALC}$ concept. Interpret concept names in C as first-order unary predicates, and roles as first-order binary predicates. Then C is translated as follows. Let p_x and p_y be two translation functions that inductively translate the concepts in C to first-order formulas with the free variables x and y, respectively. Note that x is free in p_x but bound in p_y and y is free in p_y but bound in p_x.

$\mathsf{p}_x(\top) = \mathsf{p}_y(\top) = \text{TRUE}$	$\mathsf{p}_x(\bot) = \mathsf{p}_y(\bot) = \text{FALSE}$
$\mathsf{p}_x(A) = A(x)$	$\mathsf{p}_y(A) = A(y)$
$\mathsf{p}_x(\neg C) = \neg\mathsf{p}_x(C)$	$\mathsf{p}_y(\neg C) = \neg\mathsf{p}_y(C)$
$\mathsf{p}_x(C \sqcap D) = \mathsf{p}_x(C) \wedge \mathsf{p}_x(D)$	$\mathsf{p}_y(C \sqcap D) = \mathsf{p}_y(C) \wedge \mathsf{p}_y(D)$
$\mathsf{p}_x(C \sqcup D) = \mathsf{p}_x(C) \vee \mathsf{p}_x(D)$	$\mathsf{p}_y(C \sqcup D) = \mathsf{p}_y(C) \vee \mathsf{p}_y(D)$

$$\mathsf{p}_x(\exists r{:}C) = \exists y\,(r(x;y) \wedge \mathsf{p}_y(C)) \qquad \mathsf{p}_y(\exists r{:}C) = \exists x\,(r(y;x) \wedge \mathsf{p}_x(C))$$
$$\mathsf{p}_x(\forall r{:}C) = \forall y\,(r(x;y) \rightarrow \mathsf{p}_y(C)) \qquad \mathsf{p}_y(\forall r{:}C) = \forall x\,(r(y;x) \rightarrow \mathsf{p}_x(C))$$

For $\mathcal{ALCI}$ concepts we need to extend the above translations with the following four translations.

$$\mathsf{p}_x(\exists r^{-}{:}C) = \exists y\,(r(y;x) \wedge \mathsf{p}_y(C)) \qquad \mathsf{p}_y(\exists r^{-}{:}C) = \exists x\,(r(y;x) \wedge \mathsf{p}_x(C))$$
$$\mathsf{p}_x(\forall r^{-}{:}C) = \forall y\,(r(y;x) \rightarrow \mathsf{p}_y(C)) \qquad \mathsf{p}_y(\forall r^{-}{:}C) = \forall x\,(r(y;x) \rightarrow \mathsf{p}_x(C))$$

Let C and D be any $\mathcal{ALC}$ or $\mathcal{ALCI}$ concepts. Define the translation function p which translates axioms to first-order formulae with one free variable.

$$\mathsf{p}(C \sqsubseteq D) = (\mathsf{p}_x(C) \rightarrow \mathsf{p}_x(D)) \tag{2.63}$$
$$\mathsf{p}(C \equiv D) = (\mathsf{p}_x(C) \leftrightarrow \mathsf{p}_x(D)) \tag{2.64}$$

An ontology $\mathcal{O}$, or a set of axioms, is translated as:

$$\mathsf{p}(\mathcal{O}) = \bigwedge_{C \sqsubseteq D \in \mathcal{O}} \mathsf{p}(C \sqsubseteq D) \wedge \bigwedge_{C \equiv D \in \mathcal{O}} \mathsf{p}(C \equiv D) \tag{2.65}$$

Let us see how the above translations can compute first order formulas from $\mathcal{ALC}$ and $\mathcal{ALCI}$ axioms. Consider the following axiom a.

$$A \sqcap \exists r^{-}{:}\exists r{:}B \sqsubseteq \forall r{:}C \tag{2.66}$$

We have the following.

$$\mathsf{p}(\mathsf{a}) \tag{2.67}$$
$$\mathsf{p}_x(A \sqcap \exists r^{-}{:}\exists r{:}B) \rightarrow \mathsf{p}_x(\forall r{:}C) \tag{2.68}$$
$$\mathsf{p}_x(A) \wedge \mathsf{p}_x(\exists r^{-}{:}\exists r{:}B) \rightarrow \neg\,\mathsf{p}_x(\forall r{:}C) \tag{2.69}$$
$$A(x) \wedge \exists y\,(r(y;x) \wedge \mathsf{p}_y(\exists r{:}B)) \rightarrow \neg\,(\forall y\,(r(x;y) \rightarrow \mathsf{p}_y(C))) \tag{2.70}$$
$$A(x) \wedge \exists y\,(r(y;x) \wedge \exists x\,(r(y;x) \wedge B(x)) \rightarrow \neg\,(\forall y\,(r(x;y) \rightarrow C(y)))) \tag{2.71}$$

2.7 Fixpoint Operators

Extensions of the description logics $\mathcal{ALC}$ and $\mathcal{ALCI}$ can be obtained by allowing fixpoint operators. The description logic $\mathcal{ALC}$ extended with fixpoint operators is denoted by $\mathcal{ALC}\mu$, and the description logic $\mathcal{ALCI}$ extended with fixpoint operators is denoted by $\mathcal{ALCI}\mu$. In these extensions, concepts of the following forms are allowed.

$$\mu X{:}C[X]; \tag{2.72}$$

$$\nu X{:}C[X]; \tag{2.73}$$

where μ denotes the *least fixpoint operator*, and ν denotes the *geatest fixpoint operator*. If we are in the context of the extension $\mathcal{ALC}\mu$, then C is an $\mathcal{ALC}$ concept, and C is an $\mathcal{ALCI}$ concept if we are in the context of the description logic $\mathcal{ALCI}\mu$. The symbol X is called a *concept variable*.

The description logic $\mathcal{ALC}\mu$ and $\mathcal{ALCI}\mu$ are not fragments of the first-order logic, but they are decidable fragments of the second-order logic [LPW10].

In mathematics, the fixpoints of a function f are the set of values x where $f(x) = x$. The correspondence between the algebraic understanding of fix points and fixpoint concepts is as follows. View the concept $C[X]$ as a function where the permissible values of X are concepts D such that the following is true [Tar55, NS98, Koo15].

$$C[D] \quad D \tag{2.74}$$

Let us explain through the following example. Suppose $C[B]$ is the following concept.

$$\exists r{:}B \tag{2.75}$$

The fixpoints of C are the concepts D such that $D \quad C[B{=}D]$, where $C[B{=}D]$ is the result of substituting D for B in C. As such, we can see that the following two concepts are fixpoints of C.

1. The bottom concept $\bot$, and
2. The infinite concept $\exists r{:}\exists r{:}\exists r{:}\exists r{:}\exists r{:}\exists r{:}\cdots$.

The bottom concept is a fixpoint because $\bot \quad C[\bot] \quad \exists r{:}\bot$. The infinite concept $\exists r{:}\exists r{:}\exists r{:}\exists r{:}\exists r{:}\exists r{:}\cdots$ is also a fixpoint of C, because when substituted in C for B, the result is also an infinite concept of the same form [Koo15].

The possible values of D are collectively the fixpoints of $C[X]$. If X occurs only positively in $C[X]$, then we say that C is monotone with respect to X [Sch94]. In our algebraic correspondence, a function f is monotone if for every x and y we have $x \quad y$ implies $f(x) \quad f(y)$, where $\quad$ is a partial order. The analogy also applies to the realm of concepts. That is, two fixpoints D_1 and D_2 of a concept C can be ordered such that $D_1 \quad D_2$ if $\models D_1 \sqcup D_2$ [Sch94, NS98].

According to the fixpoint theorem of Knaster and Tarski, monotone functions always have least and greatest fixpoints [Tar55]. The Knaster-Tarski theorem also states that the least and greatest fixpoints of any monotone function f are unique. Furthermore, the least fixpoint is simply the intersection of all fixpoints of f, and the greatest fixpoint is the union of all fixpoints of f. Recall that in the realm of concepts, we use $\mu X.C[X]$ to denote the least fixpoint of C with respect to X, and $\nu X.C[X]$ to denote the greatest fixpoint. Given the algebraic analogy, the least fixpoint of the concept C from the above example is the conjunction of the bottom concept and the infinite concept, which is just $\bot$. The greatest fixpoint is the disjunction of the two concepts. In other words, it is the infinite concept $\exists r.\exists r.\exists r.\exists r.\exists r.\exists r.\ldots$ [Koo15].

Least fixpoint concepts can be alternatively computed through the following formula.

$$\mu X.C[X] = \bigsqcup_{i \in \omega} C^i[\bot] \tag{2.76}$$

where ω is an ordinal, $C^i[\bot] = C[X/C^{i-1}[\bot]]$, and $C^0[\bot] = C[X/\bot]$.

Note that greatest and least fixpoint concepts are related by the following equality [Tar55].

$$\nu X.C[X] = \neg(\mu X.(\neg C[\neg X])) \tag{2.77}$$

As such, the formula (2.76) allows finding the least fixpoint concept as well as the greatest fixpoint concept.

Let us use (2.76) to compute the least and greatest fixpoint concepts of the concept $C[B]$ defined by (2.75). We have $\mu B.C[B] \quad \bigsqcup_{i \in \omega} C^i[\bot]$.

$$C^0[\bot] \quad \exists r.\bot \tag{2.78}$$

$$\bot \tag{2.79}$$

$$C^1[\bot] \quad C[B/C^0[\bot]] \tag{2.80}$$

$$\exists r.\bot \tag{2.81}$$

$$\bot \tag{2.82}$$

$$C^2[?] \quad C[B{=}C^1[?]] \tag{2.83}$$
$$\exists r{:}? \tag{2.84}$$
$$? \tag{2.85}$$
$$\vdots$$

Thus, $\mu B{:}C[B] \quad ? \sqcup ? \sqcup \quad ?$. In the same way, the greatest fixpoint concept can be computed. From (2.77), we have $\nu B{:}C[B] = \neg(\mu\, \neg B{:}(\neg C[\neg B])) = \neg(\bigsqcap_{i<\omega}(\neg C)^i[?])$. To simplify the notation, let $D = \neg C$. So, $D[\neg B] \quad \forall r{:}\neg B$. We have:

$$D^0[?] \quad \forall r{:}? \tag{2.86}$$
$$? \tag{2.87}$$
$$D^1[?] \quad D[\neg B{=}D^0[?]] \tag{2.88}$$
$$\forall r{:}? \tag{2.89}$$
$$D^2[?] \quad D[\neg B{=}D^1[?]] \tag{2.90}$$
$$\forall r{:}\forall r{:}? \tag{2.91}$$
$$D^3[?] \quad D[\neg B{=}D^2[?]] \tag{2.92}$$
$$\forall r{:}\forall r{:}\forall r{:}? \tag{2.93}$$
$$\vdots$$

Thus, $\nu B{:}C[B] \quad \neg(? \sqcup \forall r{:}? \sqcup \forall r{:}\forall r{:}? \sqcup \forall r{:}\forall r{:}\forall r{:}? \sqcup \vdots\vdots\vdots)$, which is equivalent to the concept $\top \sqcap \forall r{:}\top \sqcap \forall r{:}\forall r{:}\top \sqcap \vdots\vdots\vdots$, which is equivalent to the infinite concept $\forall r{:}\forall r{:}\forall r{:}\vdots\vdots\vdots$ obtained earlier.

Chapter 3

Background on Forgetting

The study of forgetting goes as far back as to Boole [Boo54]. Since then, forgetting has been a key engineering tool in several reasoning applications. In this chapter we build a background on forgetting. We will explain the different meanings, applications, approaches, and methods of forgetting, and review the literature related to forgetting in different logics.

Section 3.1 gives a literature review of forgetting. We discuss the progression in our understanding of forgetting, and explain the different notions of forgetting that have been introduced in the literature. Since our work is built on top of description logics, attention will be paid to the literature of forgetting in the context of description logics as well as the closely related logics.

Section 3.2 discusses the relation between forgetting and other notions in the literature such the notion of second-order quantifier elimination, and the notion of uniform interpolation.

Section 3.3 describes two state of the art forgetting methods in the literature which are closely related to the methods discussed in this thesis.

Section 3.4 discusses some of the applications that are solved or partly solved by forgetting. Many of the discussed applications are modern. So, another benefit of this discussion is picturing the modern directions in the research and the central role of forgetting played in them.

Finally, Section 3.5 groups together the definitions as well as the basic lemmas that are required for the presentation in the following chapters.

3.1 Literature Review

3.1.1 Forgetting in Propositional and Predicate Logic

The earliest known reference to forgetting [EKI19] goes as far back as to Boole who referred to it by *elimination of the middle terms* [Boo54].

In proposition logic, forgetting has been studied under different names. For instance, the name *projection* [Hoo91], the name *marginalisation* [KHM99], and the name *variable independence* [LLM03]. Although the three cited papers discuss one problem, which is forgetting in the propositional logic, they use fundamentally different methods.

The method in [LLM03] uses a resolution approach to perform propositional variable and literal forgetting. Forgetting a propositional variable eliminates the variable from the computed formula. Literal forgetting eliminates only the positive occurrences, or only the negative occurrences of a propositional variable. Resolution in [LLM03] is performed based on the following idea. Given a propositional formula f, a proposition p is eliminated from from f.

$$ELIMINATE(f; p) = f^{\top}_{p} \_ f^{?}_{p}; \tag{3.1}$$

where $f^{\top}_{p}$ denotes the formula obtained by replacing all occurrences of p in f with $\top$, and $f^{?}_{p}$ denotes the formula obtained by replacing all occurrences of p with $?$.

The method of [KHM99] makes an interesting connection between forgetting and the Davis-Putnam algorithm [DP60]. Recall that the Davis-Putnam algorithm checks the unsatisfiability of a first-order formula by eliminating, or *forgetting*, the symbols occurring in the formula. By adapting the algorithm such that only a subset of the symbols are eliminated, the forgetting method in [KHM99] was obtained.

The name *projection* in [Hoo91] was used in connection to *polyhedral projection* which transforms a propositional logic formula into a set of inequalities describing a polyhedral cone. The forgetting solution would be encoded by the points on the polyhedral cone which satisfy the set of obtained inequalities.

In first-order logic, forgetting has been studied under the name *projection* [Wer12], and *predicate elimination* [KK16]. The term *projection* in [Wer12] is different from that in [Hoo91]. *Projection* in the first-order logic refers to the forgetting of literals, the variant of forgetting considered in [LLM03] but generalized for first-order theories.

The term *Forgetting* was introduced in [LR94b, LR94a]. Two variants of forgetting

were considered in [LR94a]. The first variant is concerned with forgetting grounded atoms. A solution to this variant was provided by a generalization to equation (3.1). The second variant of forgetting is concerned with forgetting predicate symbols.

Predicate symbol forgetting was shown to be closely related to the problem of *second-order quantifier elimination*. The connection between forgetting and second-order quantifier elimination can be established as follows. Let f be a first order formula, and P a predicate symbol occurring in f. We can forget P from f by substituting a fresh predicate symbol R for P in f, where R and P have the same arity. The following theorem from [LR94a] formalizes this process.

Theorem 3.1.1. *(Predicate Forgetting [LR94a]). Suppose $T = \{\mathsf{f}\}$, and P is any n-ary predicate. Then*

$$forget(T;P) = \{(\exists R)\,(\mathsf{f}[P/R])\}; \tag{3.2}$$

where R is an n-ary second-order predicate variable, and $\mathsf{f}[P/R]$ is the result of replacing every occurrence of P in f with R.

The forgetting view in equation (3.2) is not very interesting because it only renames the predicate symbol P. An elimination of the second-order predicate symbol R would be desired. In this way, forgetting can be seen as an instance of the second-order quantifier elimination problem.

The correspondence between forgetting and second-order quantifier elimination allows for transferring known results from the latter problem to the former. One such a result is the undecidability of second-order quantifier elimination [Ack35, GO92, GSS08]. Transferring this result implies that first-order forgetting views of first-order theories cannot always be computed.

Other transferables are the methods and techniques of second-order quantifier elimination. Since these are closely related to our work, we discuss them in more details in Section 3.2.1.

Lin and Reiter used a model equivalence notion to define forgetting [LR94a]. A formula y is a forgetting view of a formula f with respect to a set of symbols S if for every model of y there is a model of f such that the two models coincide on the interpretations of S symbols, and vice versa. A formalization of this definition is as follows.

Definition 3.1.2. *(Forgetting [LR94a]). Let f be a first-order formula, and S a set of predicate symbols occurring in f. The formula y is a* forgetting view *of f with respect*

to S *if for every interpretation* I *the following is true.*

$$I \models \gamma \text{ if and only if there is a model } J \text{ of } \phi, \text{ such that } I \text{ and } J \text{ agree on all elements except, possibly, the elements in } S. \tag{3.3}$$

Interestingly, Definition 3.1.2 does not explicitly require γ to be free of the symbols from S, a condition that is generally required in the context of first-order and description logics.

Motivated by the undecidability results of forgetting in first-order logic, Zhou and Zhang [ZZ10] introduce a weaker notion of forgetting.

Definition 3.1.3. *Let* T_1 *and* T_2 *be two first-order theories, and* F *a set of predicate symbols occurring in* T_1*. We say that* T_2 *is a* forgetting view *of* T_1 *with respect to* F *if and only if the following two conditions are true.*

1. $sig(T_2) \subseteq sig(T_1) \setminus F$*, and*
2. *for every first-order theory* T_0 *where* $sig(T_0) \subseteq sig(T_1) \setminus F$*, we have* $T_1 \models T_0$ *if and only if* $T_2 \models T_0$*.*

Definition 3.1.3 is often referred to with *weak forgetting*, whereas Definition 3.1.2 is referred to with *strong forgetting*. As the names imply, strong forgetting is stronger than weak forgetting. If a forgetting view obtained by strong forgetting is expressible in first-order logic then there is a logically equivalent forgetting view obtainable by weak forgetting. But, the inverse is not true. That is, a forgetting view obtained by weak forgetting does not necessarily imply the existence of a forgetting view that adheres to the criteria of strong forgetting. Every forgetting view obtained by weak forgetting is representable in first-order logic. However, they may require infinite representations. Different proposals have been made to obtain finite representation. Since they are also applicable to forgetting in description logics, we will discuss them in the next section.

3.1.2 Forgetting in Description Logics

Strong forgetting has been studied in the context of description logics with the name *semantic forgetting*, and weak forgetting has been studied with the name *uniform interpolation* [TCMV06]. In this thesis, we call weak forgetting *deductive forgetting*. The name highlights the property that a deductive forgetting view is closed under all possible deductions of the input ontology over the non-forgotten symbols.

The notion of *inseparability*, first introduced in [Hen63], plays an important role in our understanding of forgetting [KWZ08, KWZ10, LW10, KLWW13, BKR+15, BKR+16, BKL+17]. Forgetting is the process whose inputs are an ontology $\mathcal{O}$ and a forgetting signature $\mathbf{F}$, and whose output $\mathcal{V}$ is an ontology where symbols from $\mathbf{F}$ are not used, and $\mathcal{V}$ and $\mathcal{O}$ are inseparable with respect to $sig(\mathcal{O}) \cap \mathbf{F}$. Three definitions of inseparability have been presented in the literature [KWW09, LW10, KLWW13], *deductive inseparability*, *query inseparability*, and *model inseparability*. We note them in the following

Two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ expressed in a logic $\mathcal{L}$ are deductively inseparable with respect to a signature $\mathbf{S}$ if they coincide on all entailed $\mathcal{L}$ axioms over $\mathbf{S}$. A formal definition of deductive inseparability is as follows.

Definition 3.1.4. *Let $\mathcal{L}$ be a logic, $\mathbf{S}$ a set of concept names and roles, and $\mathcal{O}_1$ and $\mathcal{O}_2$ two ontologies in $\mathcal{L}$. We say $\mathcal{O}_1$ and $\mathcal{O}_2$ are* deductively inseparable with respect to a signature $\mathbf{S}$ *if for every $\mathcal{L}$ axiom α over $\mathbf{S}$, we have $\mathcal{O}_1 \models \alpha$ if and only if $\mathcal{O}_2 \models \alpha$.*

The two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ are *query inseparable with respect to* $\mathbf{S}$ if they yield the same answers to all $\mathbf{S}$-conjunctive queries against all $\mathbf{S}$-ABoxes. More formally:

Definition 3.1.5. *Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two ontologies, and $\mathbf{S}$ a set of concept names and roles. The two ontologies are* query inseparable with respect to $\mathbf{S}$ *if for every conjunctive query $q(\vec{x})$ such that $sig(q) \subseteq \mathbf{S}$, and every ABox $\mathcal{A}$ such that $sig(\mathcal{A}) \subseteq \mathbf{S}$ we have $(\mathcal{O}_1, \mathcal{A}) \models q(\vec{a})$ if and only if $(\mathcal{O}_2, \mathcal{A}) \models q(\vec{a})$ where $\vec{a}$ is an answer to $q(\vec{x})$ comprising individuals from $\mathcal{A}$.*

Observe that Definition 3.1.5 does not require $\mathcal{O}_1$ and $\mathcal{O}_2$ to be expressed in the same logic.

The two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ are said to be *model inseparable with respect to* $\mathbf{S}$, if every model of $\mathcal{O}_1$ has a corresponding model of $\mathcal{O}_2$ such that the two models coincide on the interpretations of $\mathbf{S}$ symbols, and vice versa. A formal definition of model inseparability is as follows.

Definition 3.1.6. *Let $\mathcal{O}_1$ and $\mathcal{O}_2$ two ontologies, and $\mathbf{S}$ a set of concept names and roles. The two ontologies are* model inseparable with respect to $\mathbf{S}$ *if for every interpretation $\mathcal{I}$ we have $\mathcal{I} \models \mathcal{O}_1$ implies the existence of a model $\mathcal{J}$ of $\mathcal{O}_2$ such that $p^{\mathcal{I}} = p^{\mathcal{J}}$ for every $p \in \mathbf{S}$, and vice versa.*

Each notion of forgetting requires a notion of inseparability. Deductive forgetting requires the input ontology and the forgetting view to be at least deductively inseparable with respect to the non-forgetting symbols. Likewise, query forgetting requires the

input and the forgetting view to be at least query inseparable, and semantic forgetting requires them to be model inseparable with respect to the non-forgetting signature. The following three examples explain the three types of forgetting.

Example 3.1.7. *Consider the following ontology $\mathcal{O}$.*

$$A_1 \sqsubseteq \forall r.B \tag{3.4}$$

$$A_2 \sqsubseteq \forall r.(\neg B \sqcup C); \tag{3.5}$$

and let $\mathbf{F} = \{B\}$ be a forgetting signature. A deductive forgetting view $\mathcal{V}^d$ of $\mathcal{O}$ with respect to $\mathbf{F}$ is the ontology consisting of the following axiom,

$$A_1 \sqcap A_2 \sqsubseteq \forall r.C; \tag{3.6}$$

The ontologies $\mathcal{O}$ and $\mathcal{V}^d$ are deductively inseparable with respect to the remaining symbols $\{A_1, A_2, r, C\}$ because (3.6) is the only $\mathcal{ALC}$ consequence of (3.4) and (3.5) over $\{A_1, A_2, C, r\}$.

Example 3.1.8. *Continuing with Example 3.1.7, consider the following ABox $\mathcal{A}$ over $\{A_1, A_2, r\}$.*

$$A_1(a_1) \tag{3.7}$$

$$A_2(a_2) \tag{3.8}$$

$$r(a_1, c) \tag{3.9}$$

$$r(a_2, c) \tag{3.10}$$

Suppose we have a conjunctive query $q(x) = C(x)$. We can see that $(\mathcal{O}, \mathcal{A}) \models C(c)$ because:

1. *The axiom (3.4), and the assertions (3.7), and (3.9) imply that c is an instance of B.*
2. *The axiom (3.5), and the assertions (3.8), and (3.10) imply that c is not an instance of B, or c is an instance of C.*
3. *The previous two conclusions imply that c is an instance of C.*

We can also see that $(\mathcal{V}^d, \mathcal{A}) \not\models C(c)$ because a_1 is not an instance of the conjunction $A_1 \sqcap A_2$, and a_2 is not an instance of the same conjunction. So, $\mathcal{O}$ and $\mathcal{V}^d$ are not query

inseparable with respect to $\{A_1, A_2, C, r\}$. *Consider the ontology* $\mathcal{V}^q$ *consisting of the following* $\mathcal{ALCI}$ *axiom.*

$$\exists r^-.A_1 \sqcap \exists r^-.A_2 \sqsubseteq C \tag{3.11}$$

We can see that $(\mathcal{V}^q, A) \models C(c)$ *because* c *is an instance of* $\exists r^-.A_1$ *(from (3.9)), and* c *is an instance of* $\exists r^-.A_2$ *(from (3.10)). We will prove in Chapter 5 that* $\mathcal{O}$ *and* $\mathcal{V}^q$ *are query inseparable with respect to* $\{A_1, A_2, C, r\}$. *Moreover, it can be shown that the two ontologies are model inseparable with respect to* $\{A_1, A_2, C, r\}$. *So, not only is* $\mathcal{V}^q$ *a query forgetting view of* $\mathcal{O}$ *with respect to* $\mathbf{F}$, *but also a semantic forgetting view.*

Example 3.1.9. *Consider the ontology* $\mathcal{O}$ *comprising the following axioms.*

$$A \sqsubseteq \exists r.B \tag{3.12}$$

$$A \sqsubseteq \exists r.\neg B \tag{3.13}$$

Let $\mathbf{F} = \{B\}$ *be a forgetting signature. The ontology* $\mathcal{V}$ *consisting of the following axiom is a deductive forgetting view, and a query forgetting view of* $\mathcal{O}$ *with respect to* $\mathbf{F}$.

$$A \sqsubseteq \exists r.\top \tag{3.14}$$

However, $\mathcal{V}$ *is not a semantic view of* $\mathcal{O}$ *with respect to* $\mathbf{F}$, *because* $\mathcal{O}$ *encodes the information that elements in the extension of* A *are related via* r *to elements in two disjoint sets. An information that is not preserved in* $\mathcal{V}$.

Using inseparability based definition of forgetting allows tailoring the forgetting towards particular applications. For instance, *classification* is a reasoning task whereby a taxonomy of the concepts occurring in a given ontology is obtained. An essential operation in a classification task is checking the entailment of subsumption axioms $C \sqsubseteq D$. Suppose $\mathcal{O}$ is an ontology, and $\mathcal{V}$ a forgetting view of $\mathcal{O}$. If a $\mathcal{V}$ is used in an application where classification is required, then the concept classification obtained using the ontology $\mathcal{V}$ should be indistinguishable from the concept classification obtained if $\mathcal{O}$ is used. At least deductive inseparability is therefore needed, otherwise the entailment checking inferences may compute incorrect answers when $\mathcal{V}$ is used.

In the same way, if the forgetting view is going to be used in ontology based data access applications (see Section 3.4), then deductive inseparability is not sufficient, because it does not guarantee obtaining the same answers to queries when the forgetting view is used as those that would be obtained if the original ontology is used. Instead of

deductive inseparability, it is better to rely on query inseparability, where the answers of the original ontology to queries and the answers of the forgetting view to the same queries would coincide.

Model inseparability, as noted in [BKL$^+$17], implies that the original ontology and the forgetting view coincide on all entailments expressed in any language up to second-order language. Thus, semantic forgetting views subsume query and deductive forgetting views, and can be used in all applications where they are used. However, they are often harder to compute, and may contain more information than needed to perform the reasoning tasks.

The theoretical properties of deductive forgetting in the context of the description logic $\mathcal{ALC}$ have been investigated in [WWTP08, LW11]. Deductive forgetting views of $\mathcal{ALC}$ ontologies are always representable in $\mathcal{ALC}$. However, there are cases where infinite representation is required. In these cases, we say that the deductive forgetting view does not exist. Recall that deductive forgetting is synonym to weak forgetting. Thus, the existence properties of $\mathcal{ALC}$ forgetting view coincide with those of first-order forgetting views obtained by weak forgetting. We show an example where infinite representation is needed. Consider the ontology $\mathcal{O}$ consisting of the following axioms.

$$A \sqsubseteq B \tag{3.15}$$

$$B \sqsubseteq \exists r.B \tag{3.16}$$

A deductive forgetting view $\mathcal{V}$ of $\mathcal{O}$ with respect to the forgetting signature $\Sigma = \{B\}$ is the ontology comprising infinite number of axioms of the form $A \sqsubseteq (\exists r)^n.\top$, where $n \geq 1$, and $(\exists r)^n$ is a shorthand for the following expression.

$$\underbrace{\exists r.\exists r.\ldots.\exists r}_{\text{n times}}$$

The existence of $\mathcal{ALC}$ deductive forgetting views, or in other words the existence of a finite representation, is decidable in double exponential time. Furthermore, when the forgetting view is finitely expressible, its size can be triple exponential in the size of the input ontology in the worst case [LW11]. There are three proposals in the literature to obtain a finite representation of the deductive forgetting view when an infinite representation would be normally required.

The first proposal, often denoted by *bounded forgetting*, restricts the size of the

concepts in the deductive forgetting view, so that a finite representation is obtainable. The proposal was first investigated in [WWTP08] in the context of the description logic $\mathcal{ALC}$, and generalized later to first-order logic in [ZZ11]. The obtained forgetting view would be weaker than the deductive forgetting view, and can be seen as an approximation. An example of this approach is the ontology $\mathcal{V}_b$ the finite set of axioms $A \sqsubseteq (\exists r)^m.\top$ where $m \geq l, l \geq 1$, and $m, l \in \mathbb{N}$. The ontology $\mathcal{V}_b$ is an approximation to the infinite deductive forgetting view $\mathcal{V}$ described above. $\mathcal{V}_b$ is weaker that $\mathcal{V}$ because $\mathcal{V}_b \not\models A \sqsubseteq (\exists r)^k.\top$ for every $k \geq l + 1$.

The second proposal extends the language of the forgetting view with fixpoint operators, a proposal that was suggested by the second-order quantifier elimination method in [NS98]. An example of this proposal on the above example is the forgetting view $\{A \sqsubseteq \nu X.(X \sqcap \exists r.X)\}$ where ν is the greatest fixpoint operator, and X is a concept variable. As noted in [NS98], first-order logic extended with fixpoints have polynomial time model checking over finite domains. Thus, a forgetting view expressed using fixpoints can be practical in some applications such as model checking. But, as far as we know, there is only one reasoner that accepts ontologies with fixpoints [GHH+23]. So the usability of a forgetting view with fixpoints remains limited in applications that cannot use the cited reasoner.

The third proposal extends the language of the forgetting view with second-order predicate symbols [LR94a, KS13c]. Although the proposal may be counter-intuitive as the aim is to the eliminate the second-order symbols in the forgetting view, the below properties can be obtained by this approach. Some of them will be established in Chapter 4.

1. The forgetting view is expressed in $\mathcal{ALC}$ without any new operators.

2. The forgetting view is deductively inseparable from the input ontology with respect to the non-forgotten symbols.

3. The introduced second-order symbols are often fewer than the forgetting symbols.

The first property allows access to a wide range of $\mathcal{ALC}$ reasoning methods and applications. It is particularly important to users who require their data to be in encoded in the $\mathcal{ALC}$ language. The second property guarantees the completeness of the forgetting view. The third property suggests that forgetting views represented with this proposal can be practical. In Chapter 7 we evaluate the number of the second-order symbols

that are introduced by this proposal against the number of forgetting symbols in forgetting tasks with different complexities. Our findings suggest that the second-order symbols that would be used are often significantly fewer than the forgotten symbols. The forgetting methods in Chapters 4, 5, and 6 follow this proposal to obtain a finite representation of the semantic forgetting view. This proposal was also used in the forgetting methods in [Koo15], where the second-order symbols were referred to by the name *definers*, see also Section 3.3.1. The same name is adopted in the forgetting methods presented in this thesis.

Several deductive forgetting methods have been developed for the description logics in the $\mathcal{ALC}$ family.- A comprehensive list of these methods is presented in Table 3.1. The method [KS13c] in the third row of the table is the basis for all the methods [KS13b, KS14, KS15a, KS15b] that come after it in the table. Since there is an overlap between this method and the one presented in Chapter 4 of this thesis, a detailed discussion of this method will be presented in Section 3.3.1.

Yizheng and Schmidt developed a series of semantic forgetting methods which we list Table 3.2. Note that there are cases where the semantic forgetting views of the methods in Table 3.2 are not complete [Zha18], a result that may be expected by recalling that semantic forgetting is stronger than deductive forgetting. As such, theoretical existence results of deductive forgetting transfer directly to semantic forgetting. Since deductive forgetting views of $\mathcal{ALC}$ ontologies may not exist, incompleteness of the methods in Table 3.2 would then be expected to follow. The method [ZS15] is the basis for all the methods in Table 3.2. A detailed description of it will be presented in Section 3.3.2.

A common first step of the forgetting methods in Tables 3.1 and 3.2 is transforming the input ontology to a normal form. As noted in [DP60], normal forms cut down structural complexity and eventually simplifies the forgetting method. For instance, when the ontology is expressed in a disjunctive normal form, forgetting can be distributed over disjunctions [LR94a, KWW08, ZZ10]. As such, forgetting is simplified to eliminating forgetting symbols from the individual disjuncts. When the ontology is expressed in a conjunctive normal form, forgetting can be performed by resolving together the clauses where the forgetting symbol occur. This process that can be performed iteratively to eliminate the forgetting symbols one after the other.

Despite the simplifications offered by the disjunctive normal form, only the deductive forgetting method in [WWTP08] uses it. Other forgetting methods in Tables 3.1 and 3.2 use conjunctive normal forms. The reason of this bias is perhaps what was

Publication	Language	Description
[WWT+09]	$\mathcal{ALC}$	A forgetting method for $\mathcal{ALC}$ ontologies.
[LK14]	$\mathcal{ALC}$	A forgetting method for $\mathcal{ALC}$ ontologies.
[KS13c]	$\mathcal{ALC}$	A forgetting method for $\mathcal{ALC}$ ontologies. The method was developed simultaneously but independently of [LK14].
[KS13b]	$\mathcal{ALCH}$	A forgetting method for ontologies in the extension $\mathcal{ALC}$ with role hierarchy ($\mathcal{H}$).
[KS14]	$\mathcal{SHQ}$	A forgetting method for ontologies in the extension of $\mathcal{ALC}$ with transitive roles ($\mathcal{S}$), role hierarchy, and qualified number restrictions ($\mathcal{Q}$).
[KS15a]	$\mathcal{SIF}$	A forgetting method for ontologies in the extension of $\mathcal{ALC}$ with transitive roles, inverse roles ($\mathcal{I}$), and functional roles ($\mathcal{F}$).
[KS15b]	$\mathcal{ALC}$ KB	A forgetting method for $\mathcal{ALC}$ knowledge bases.

Table 3.1: Deductive forgetting methods for the description logics in the $\mathcal{ALC}$ family.

noted by Davis and Putnam, that disjunctive normal forms lead to exponential growth in the size of the formula, and is unfeasible in general [DP60]. In their comment, Davis and Putnam are making the note that a first-order theory is translated to a disjunctive normal form by taking the conjunction of its formulas and translating it in one big step whereas it can be translated to conjunctive normal form by translating each formula independently. In the ontology world, we add to the above note that when the input of forgetting is translated to disjunctive normal form, the output is often expressed in disjunctive normal form, and a further translation to conjunctive normal form will be required to express it as a set of axioms. This extra transformation is expensive in terms of time and computation resources. Furthermore, it can be viewed as redundant since it would not be required if a conjunctive normal form was used in the first step. Evaluations of the method in [WWTP08] were not provided, and so the performance implications of the disjunctive normal form are not known.

Forgetting methods have been developed for ontologies in extensions of the description logics $\mathcal{EL}$ and DL-Lite. Table 3.3 summarizes the developed methods. A semantic forgetting method was developed in [WWTP08] for ontologies in the DL-Lite description logic. The method is resolution based. It was shown that when forgetting is only performed for concept names, then a semantic forgetting view of a DL-Lite

Publication	Language	Description
[ZS15]	$\mathcal{ALCOI}$	A forgetting method for ontologies in the extension of $\mathcal{ALC}$ with nominals($\mathcal{O}$) and inverse roles ($\mathcal{I}$).
[ZS16]	$\mathcal{ALCOIH}\mu^{+}(\tilde{\mathrm{N}}, u)$	A forgetting method for ontologies in the extension of $\mathcal{ALCOI}$ with role hierarchy, top role ($\tilde{\mathrm{N}}$), and least fixpoint operator (μ)
[ZS17]	$\mathcal{ALCOQH}(\tilde{\mathrm{N}})$	A forgetting method for ontologies in the extension of $\mathcal{ALC}$ with nominals, qualified number restriction, role hierarchy, and top role.
[ZS18]	$\mathcal{ALCOQ}(:, u, t)$	A forgetting method for ontologies in the extension of $\mathcal{ALC}$ with nominals, qualified number restriction, role negation, role conjunction, and role disjunction.

Table 3.2: Semantic forgetting methods for the description logics in the $\mathcal{ALC}$ family.

Publication	Language	Description
[WWTP08]	DL-Lite	Semantic forgetting method for DL-Lite ontologies.
[KWW09]	$\mathcal{EL}$	Deductive and query forgetting methods for ontologies in extensions of the description logic $\mathcal{EL}$.

Table 3.3: Forgetting methods for the description logics in the $\mathcal{EL}$ and DL-Lite families.

ontology $\mathcal{O}$ always exist in DL-Lite, and forgetting a single concept name is polynomial in the size of $\mathcal{O}$. Although no complexity result was shown for forgetting a set of concept names of arbitrary size, we can deduce that the size of the forgetting view is exponential in the size of $\mathcal{O}$ because if forgetting a single symbol produces an output of size $\|\mathcal{O}\|^n$ where $\|\mathcal{O}\|$ means the size of $\mathcal{O}$ then forgetting k symbols produces an output which is $\|\mathcal{O}\|^{n^{\cdot^{\cdot^{\cdot^{n}}}}}$ (k times) where the exponent is the multiplication of n by itself k times. The value of this is just $\|\mathcal{O}\|^{n^k}$.

Deductive and query forgetting methods for ontologies in extensions of the description logic $\mathcal{EL}$ were developed in [KWW09]. Two flavors of query forgetting were considered. The first is *instance query forgetting*, a special case of query forgetting where

the forgetting view and the input ontology coincide on answers to instance queries but not necessarily other forms of conjunctive queries. The second is conjunctive query forgetting, the notion described before by query inseparability where the forgetting view and the input ontology coincide on answers to conjunctive queries against arbitrary ABoxes. Forgetting views of the three types of forgetting, that is, deductive, instance, and conjunctive query forgetting, were shown to always exist under some acyclicity conditions. The size of the forgetting view was shown to be exponential in the size of the input ontology in the worst case.

3.2 Related Notions

3.2.1 Second Order Quantifier Elimination

Second-order quantifier elimination is the problem concerned with eliminating quantified second-order predicates, and obtaining an equivalent first-order formula which is free of the eliminated symbols [Ack35]. Although undecidable, algorithms were developed to eliminate second-order quantifiers [Ack35, GO92, GSS08]. There are two approaches generally followed in the developed algorithms. These are the resolution approach, and the approach based on applications of a result of Ackermann which is commonly referred to as Ackermann's lemma [Ack35].

The resolution approach have been followed in SCAN [GO92, Eng96, GSS08]. The input to the SCAN tool is first converted to a clausal form, where first-order quantifiers are eliminated, negations are applied only to predicate symbols, and formulas are rewritten in conjunctive normal form. Then, resolution is performed on the clauses where the second-order predicate occurs. When all resolvents have been captured, the clauses where the predicate occurs are removed. Finally, the first-order quantifiers which have been eliminated in the first step are restored back. If the last step is successful, the desired first-order formula would be obtained, otherwise the algorithm fails. This general approach has been followed in many forgetting methods including the ones presented in Chapters 4 and 5 and the ones in [HM08, LK14, KS13c, KS13b, KS14, KS15a, KS15b, KK16].

The approach based on Ackermann's lemma have been followed in DLS and other methods [DŁS97, Sch12, AS19]. The general flow of the approach comprises the following three steps performed one after the other.

1. Rewriting the formulas of the input theory in a form suitable for performing the

transformation of Ackermann's lemma.

2. Performing the transformation of the Ackermann's lemma.

3. Simplifying the output.

Ackermann's lemma has been first introduced in [Ack35]. It is formulated in this thesis through the following lemma.

Lemma 3.2.1. *([Ack35]). Let $\bar{u};\bar{v}$ be two ordered sets of variable, Q a predicate variable of arity $|\bar{u}|$, $F(\bar{u};\bar{v})$ a formula in which Q does not occur, $G^+[Q]$ a formula in which Q occurs only positively, and $G^-[Q]$ a formula in which Q occurs only negatively. Then, the following equivalences hold.*

$$\exists Q.\forall \bar{u}.((\neg Q(\bar{u}) \vee F(\bar{u};\bar{v})) \wedge G^+[Q] \quad G^+[Q/F(\bar{u};\bar{v})] \tag{3.17}$$

$$\exists Q.\forall \bar{u}.(Q(\bar{u}) \vee F(\bar{u};\bar{v})) \wedge G^-[Q] \quad G^-[Q/\neg F(\bar{u};\bar{v})]; \tag{3.18}$$

where the variables of $\bar{u}$ in $G^+[Q/F(\bar{u};\bar{v})]$ are renamed to the respective arguments of Q in $G^+[Q]$, and the variables of $\bar{u}$ in $G^-[Q/F(\bar{u};\bar{v})]$ to the respective arguments of Q in $G^-[Q]$.

The transformations in the Ackermann's lemma can be performed in the two directions, from left to right, and from right to left. The transformations from right to left are referred to as *structural transformation*. An operation that is often performed in normal form transformations to reduce the size of the normal form [NW01, BUL01]. Structural transformation is used in our fine-grained forgetting framework presented in Chapter 4, and the forgetting methods in [KS13c, KS13b, KS14, KS15a, KS15b].

The transformations from left to right eliminate the second-order quantified predicate symbol Q. They are restricted by the syntactic form of the formulas on the left hand sides of (3.17) and (3.18). In (3.17) the predicate symbol Q is required to occur negatively in one conjunct of the formula, and only positively in the remainder of the conjuncts. In (3.18) the predicate symbol Q is required to occur positively in one conjunct of the formula, and only negatively in the remainder of the conjuncts. The two transformations can be seen as a form of resolution, where the literal $\neg Q(\bar{u})$ is resolved with the positive occurrences of Q in G^+, and the literal $Q(\bar{u})$ is resolved with the negative occurrences of Q in G^-. The syntactic restrictions described above emphasize the rewrite operations performed in the first step of the flow, which is perhaps what distinguishes this approach from the resolution approach followed by SCAN.

One restriction in Lemma 3.2.1 is that the formulas F^+ and F^- are free of Q. A generalization of the Ackermann's lemma which removed this restriction has been provided in [NS98].

Lemma 3.2.2. *([NS98]). Let $\bar{u}$ be an ordered sets of variable, Q a predicate variable of arity $|\bar{u}|$, $F^+[Q]$ and $G^+[Q]$ two formulas where Q occurs only positively, and $F^-[Q]$ and $G^-[Q]$ two formulas where Q occurs only negatively. Assume Q is not grounded in $G^+[Q]$ and $G^-[Q]$, that is, it occurs only with variables as arguments but not with terms. Then,*

$$\exists X.\forall \bar{u}(\neg Q(\bar{u}) \vee F^+[Q]) \wedge G^+[Q] \quad G^+[Q=\nu Q(\bar{u}).F^+[Q]]; \tag{3.19}$$

$$\exists X.\forall \bar{u}(Q(\bar{u}) \vee F^-[Q]) \wedge G^-[Q] \quad G^-[Q=\mu Q(\bar{u}).F^-[Q]]; \tag{3.20}$$

where ν is the greatest fixpoint operator, μ is the least fixpoint operator, and the arguments of Q bound by the fixpoint operators are mapped to the actual variables of the substituted predicates.

The transformations in the generalized Ackermann's lemma allow the predicate Q to occur in F^+ and F^-. In compensation, the fixpoint operators are used in the formulas on the right hand sides of (3.19) and (3.20). The generalized lemma has been used in the DLS* algorithm [NS98]. It has also been used to obtain finite representation of the deductive forgetting views obtained by the methods in [KS13c, KS13b, KS14, KS15a, KS15b], and the semantic forgetting method in [ZS16].

3.2.2 Uniform Interpolation

The two notions of *deductive forgetting* and *uniform interpolation* [Vis96, TCMV06, GLW06] coincide. We explain uniform interpolation, and its correspondence to deductive forgetting.

Uniform interpolation is a stronger variant of *Craig interpolation* [Cra57]. Let us recall Craig interpolation by the following definition.

Theorem 3.2.3. *(Craig Interpolation [Cra57]). Let Σ_1 and Σ_2 be two first-order formulas such that $\Sigma_1 \models \Sigma_2$. Let Σ be the common symbols of Σ_1 and Σ_2. There is a first-order formula β over Σ, such that the following is true.*

$$\Sigma_1 \models \beta \tag{3.21}$$

$$\beta \models \Sigma_2 \tag{3.22}$$

The formula β *is called the* Craig interpolant *of* Γ_1 *and* Γ_2.

That is, given knowledge of Γ_1, Γ_2, and the common symbols between them, there is a formula β expressed over the common symbols such that the conditions (3.21) and (3.22) are true. Note that in its original form, the syntactic notion of derivability, $\vdash$, was used in (3.21) and (3.22) in place of the semantic notion of entailment, $\models$. The use of the entailment is correct due to the completeness theorem, $\Gamma \models A$ if and only if $\Gamma \vdash A$, of Gödel assuming a sound and complete derivation calculus [Göd30].

Uniform interpolation is obtained from Craig's interpolation by omitting the knowledge of Γ_2 in the hypothesis.

Definition 3.2.4. *(Uniform interpolation). A logic* $\mathcal{L}$ *has has the* uniform interpolation *property if and only if for any formula* Γ_1 *in* $\mathcal{L}$ *and any set* $S \subseteq sig(\Gamma_1)$ *of symbols, there exists a formula* β *such that for every* $\mathcal{L}$ *formula* Γ_2 *containing only symbols in* S *we have,*

$$\Gamma_1 \models \Gamma_2 \text{ if and only if } \beta \models \Gamma_2 \tag{3.23}$$

The formula β *is referred to by the* uniform interpolant *of* Γ_1 *with respect to* S.

We can see by comparing definitions 3.2.4 and 3.1.3 that the two notions of deductive forgetting and uniform interpolation coincide. Forgetting a set of symbols F from a theory T_1 is equivalent to computing the uniform interpolant of T_1 with respect to $S = sig(T_1) \setminus F$.

3.3 Forgetting Methods

The forgetting methods in [KS13c, KS13b, KS14, KS15a, KS15b] have been implemented in one system, *Lethe* [KS13a]. They are extensions to the method in [KS13c], which performs forgetting in the context of the description logic $\mathcal{ALC}$.

Likewise, the forgetting methods in [ZS15, ZS16, ZS17, ZS18] have been implemented in one system, Fame [Zha18]. They are extensions to the method in [ZS15], which performs forgetting in the context of the description logic $\mathcal{ALCOI}$, the extension of $\mathcal{ALC}$ with nominals and inverse roles. Empirical evaluations suggest that Lethe and Fame are the most practical forgetting systems for expressive description logics in the family of $\mathcal{ALC}$ [KS13a, ZFA+18].

There are common features between the forgetting method of [KS13c] and the deductive customization of the forgetting framework presented in Chapter 4. Moreover,

the forgetting methods developed in Chapters 4 and 6 are evaluated against Lethe and Fame.

In this section, we explain the forgetting methods of presented in [KS13c] and [ZS15]. To simplify the presentation, we shall refer to the method of [KS13c] by the method of Lethe, and the method of [ZS15] by the method of Fame.

3.3.1 The Method of Lethe

The method of Lethe is a resolution based method. It combines aspects from the two approaches of SCAN and Ackermann explained in Section 3.2.1, which we will explain in the following. The method runs in three stages.

The input of the first stage is the original ontology. The output is a representation of the input ontology in a normal form, which is inspired by the normal form of SCAN. Each axiom is transformed independently to the normal form by a pipeline of three consecutive transformations. The output of one transformation is input to the next.

The first transformation in the pipeline rewrites the axiom as a clause in negation normal form with negations applied directly to concept names. The second transformation flattens existential and universal role restrictions by extracting the role fillers in new clauses. The third transformation, converts the clauses from the previous transformation to a conjunctive normal form.

The second transformation of the described pipeline uses the transformation in (3.17) in the direction from right to left. Recall that in this direction, a formula F is extracted by introducing a new second-order quantified symbol and adding a new conjunct to the ontology. This operation is often referred to as structural transformation, and the second-order symbols are referred to by the name *definers*. Definers must be fresh. In other words, they cannot be already in use when they are introduced. We explain the transformation to normal form through the following example.

Example 3.3.1. *Consider the ontology $\mathcal{O}$ comprising the following axioms.*

$$A \sqsubseteq \forall r.(B \sqcap C) \tag{3.24}$$

$$E \sqsubseteq \exists r.\top \sqsubseteq B \tag{3.25}$$

Let us transform the axiom (3.24) to the normal form. First we rewrite it in negation

normal form as follows.

$$:A \sqsubseteq \forall r{:}(B \sqcap C) \tag{3.26}$$

Second we flatten (3.26) by a structural transformation operation. This extracts the role filler $B \sqcap C$ in a new clause and introduces a new definer D_1.

$$:A \sqsubseteq \forall r{:}D_1 \tag{3.27}$$

$$:D_1 \sqsubseteq (B \sqcap C); \tag{3.28}$$

where D_1 is a definer. Finally, the two clauses (3.27) and (3.28) are transformed to conjunctive normal form. The clause (3.27) is already in conjunctive normal form, so no change is done. The clause (3.28) is transformed to $(:D_1 \sqsubseteq B) \sqcap (:D_1 \sqsubseteq C)$ which can be rewritten as two separate clauses.

In the same way the axiom (3.25) is transformed, and the normal form representation of $\mathcal{O}$ would be the ontology comprising the following clauses.

$$:A \sqsubseteq \forall r{:}D_1 \tag{3.29}$$

$$:D_1 \sqsubseteq B \tag{3.30}$$

$$:D_1 \sqsubseteq C \tag{3.31}$$

$$:E \sqsubseteq \exists r{:}D_2 \tag{3.32}$$

$$:D_2 \sqsubseteq :B \tag{3.33}$$

The second stage of the method of Lethe eliminates the symbols of the forgetting signature from the normal form representation obtained in the first stage. The method uses resolution. As such, it follows the approach of SCAN. The calculus comprises the three resolution rules listed in Figure 3.1, which are applied to the clauses of the normal form representation until saturation [Koo15]. The Lethe method restricts its calculus rules in the sense that they are used only when their conclusions have at most one negative definer. If the application of any of the rules would lead to two distinct negative definers in the conclusion, then the method will prevent it from performing. When all possible inferences have been performed, the clauses where a forgetting symbol occur are removed. Through out the process, standard optimizations such as subsumption elimination, purification, and tautology deletion are eagerly performed.

Example 3.3.2. *Continuing with Example 3.3.1. The two clauses (3.30) and (3.33)*

Resolution

$$\frac{C_1 \sqsubseteq B; \quad C_2 \sqsubseteq \neg B}{C_1 \sqsubseteq C_2}$$

where B is a forgetting symbol or a definer, and C_1 and C_2 are concepts.

Role Propagation

$$\frac{C_1 \sqsubseteq \forall r{:}D_1; \quad C_2 \sqsubseteq Q\, r{:}D_2}{C_1 \sqsubseteq C_2 \sqsubseteq Q\, r{:}D_3; \quad \neg D_3 \sqsubseteq D_1; \quad \neg D_3 \sqsubseteq D_2}$$

where $Q \in \{\exists, \forall\}$, C_1 and C_2 are concepts, D_1 and D_2 are definers, and D_3 is a fresh definer.

Existential Elimination

$$\frac{\mathcal{O} \cup \{C \sqsubseteq \exists r{:}D; \quad \neg D\}}{\mathcal{O} \cup \{C\}}$$

where C is a concepts.

Figure 3.1: The resolution calculus rules of the method of Lethe.

cannot be resolved together on the symbol B because the conclusion would contain the two negative definers $\neg D_1$ *and* $\neg D_2$.

Instead, a Role Propagation *inference can be performed on the two clauses (3.29) and (3.32). The conclusion comprises the following three clauses.*

$$\neg A \sqsubseteq \neg E \sqsubseteq \exists r{:}D_3 \tag{3.34}$$

$$\neg D_3 \sqsubseteq D_1 \tag{3.35}$$

$$\neg D_3 \sqsubseteq D_2 \tag{3.36}$$

where D_3 *is a fresh definer.*

Next, the clause (3.35) is resolved with the clauses (3.30) and (3.31), and the clause (3.36) is resolved with the clause (3.33). We obtain:

$$\neg D_3 \sqsubseteq B \tag{3.37}$$

$$\neg D_3 \sqsubseteq C \tag{3.38}$$

$$\neg D_3 \sqsubseteq \neg B \tag{3.39}$$

The two clauses (3.37) and (3.39) are then resolved on B, and we obtain the following

clause.

$$\neg D_3 \tag{3.40}$$

The clause (3.40) subsumes the clauses (3.35), (3.36), and (3.38). The subsumed clauses are removed.

Next, we perform an Existential Elimination *inference with the input clauses (3.34) and (3.40). The inference removes (3.34) and (3.40) and computes the following clause.*

$$\neg A \sqcup \neg E \tag{3.41}$$

Since (3.41) subsumes (3.34), we remove (3.34).

At this point, no further inferences can be performed. We remove all clauses where the symbol B occurs, and obtain the following ontology comprising (3.29), (3.31), (3.32), and (3.41).

$$\neg A \sqcup \forall r.D_1$$
$$\neg D_1 \sqcup C$$
$$\neg E \sqcup \exists r.D_2$$
$$\neg A \sqcup \neg E$$

Let us stop for a moment at the *Role Propagation* inference performed in the above example. The inference was performed using the premises $\neg A \sqcup \forall r.D_1$ (3.29) and $\neg E \sqcup \exists r.D_2$ (3.32). The conclusion of the inference was the three clauses $\neg A \sqcup \neg E \sqcup \exists r.D_3$ (3.34), $\neg D_3 \sqcup D_1$ (3.35), and $\neg D_3 \sqcup D_2$ (3.36). In the original description of the method, there is nothing preventing another *Role Propagation* inference from performing on the premises (3.29) and (3.34). Suppose we perform this inference, the clauses $\neg A \sqcup \neg E \sqcup \exists r.D_4; \neg D_4 \sqcup D_1$, and $\neg D_4 \sqcup D_3$ will be generated. We notice here that there is not any difference between the concept descriptions of D_3 and D_4. The two clauses (3.34) and $\neg A \sqcup \neg E \sqcup \exists r.D_4$ are identical modulo the definer, and the two clauses $\neg D_4 \sqcup D_1$ and $\neg D_4 \sqcup D_3$ can be rewritten as $\neg D_4 \sqcup (D_1 \sqcap D_3)$ which is logically equivalent to $\neg D_4 \sqcup D_3$ because from (3.35) D_3 is a subset of D_1. In this sense, a *Role Propagation* inference between the conclusion and any of the premises of a previous *Role Propagation* inference is meaningless. Furthermore, such inferences, if allowed, will lead to infinite series of *Role Propagation* inferences and will prevent the termination of the method. The implementation of Lethe [KS13a] maintains a subsumption hierarchy of definers, and disallows a *Role Propagation* inference if one definer in the

premises subsumes the other. That is, if D_1 in the *Role Propagation* rule in Figure 3.1 subsumes D_2 or vice versa. In the example, the inference over (3.29) and (3.34) is not performed because D_1 subsumes D_3.

The third stage of the method of Lethe eliminates definers from the ontology obtained by the second stage. The elimination is performed using the Ackermann's approach explained in Section 3.2.1. That is, the clauses $:D \sqsubseteq C_1; \ldots; :D \sqsubseteq C_n$ where the definer D occurs negatively are rewritten in a form that resembles the first conjunct of (3.17) or (3.19). In this case, we would rewrite $:D \sqsubseteq C_1; \ldots; :D \sqsubseteq C_n$ as $:D \sqsubseteq (C_1 \sqcap \ldots \sqcap C_n)$. Then, the transformation of (3.17) or (3.19) is performed from left to right to eliminate the definer.

Example 3.3.3. *Continuing with Example 3.3.1 and 3.3.2, the definer D_1 can be eliminated directly using the transformation in (3.17). We obtain the following clause.*

$$:A \sqsubseteq \forall r.C \tag{3.42}$$

The definer D_2 occurs with only one polarity, so it can be eliminated by replacing it with $\top$. The final forgetting view would be obtained as the following ontology.

$$:A \sqsubseteq \forall r.C$$
$$:E \sqsubseteq \exists r.\top$$
$$:A \sqsubseteq :E$$

We summarize the above description of the method of Lethe in the following.

1. The method of Lethe uses a conjunctive clausal normal form enhanced with structural transformation which may introduce second-order quantifiers, or definers.

2. In this normal form and throughout the forgetting process, clauses can use just one definer with negative polarity. The restriction can be seen as an implicit filtering to the information that is preserved. Each restricted inference corresponds to some information that is filtered out.

3. The calculus of the method is resolution based. It may introduce new definers dynamically through role propagation inferences.

4. The method combines the resolution approach and the approach based on Ackermann's lemma. The latter approach is realized by using the transformations

in (3.17) and (3.19) for the operations of structural transformation and definer elimination.

We highlight some shortcomings of the method of Lethe. These shortcomings will be addressed in the fine-grained forgetting framework and the deductive customization that will be proposed in Chapter 4.

1. *The normal form:* Many of the definers introduced by structural transformation could have been avoided by simple syntactic rewriting. For instance, the clause $A \sqcap \exists r.B \sqcap \exists r.C$ is structurally transformed to the three clauses $A \sqcap \exists r.D_1 \sqcap \exists r.D_2; \neg D_1 \sqcap B; \neg D_2 \sqcap C$. Here the output uses two definers D_1 and D_2 but only one is sufficient. To see this, let us rewrite the clause as $A \sqcap \exists r.(B \sqcap C)$ which is logically equivalent to the original clause. The new clause is then structurally transformed to the two clauses $A \sqcap \exists r.D_3; \neg D_3 \sqcap B \sqcap C$ where only one definer D_3 is used. Introducing definers sparingly can improve the performance of the method by eliminating the time required to eliminate them.

2. *Omitting of information:* The method of Lethe performs an implicit omitting of information by preventing the calculus rules from computing clauses where two negative definers occur. This omitting of information is performed without notifying the user of what information has been filtered out.

3. *Dynamic definers:* Some definers are introduced dynamically by *Role Propagation* inferences. For example, the clauses (3.34), (3.35), and (3.36) in Example 3.3.2 have the definer D_3 which is introduced by a *Role Propagation* inference. There are two complexities associated with the dynamic introduction of definers. (1) Being introduced as subsets of other definers can lead to non-termination as discussed above, and requires maintaining a subsumption hierarchy between the definers to guarantee termination. (2) The way these definers are introduced puts the ontology in a state where the forgetting symbols can not be eliminated and requires extra processing to recover from this state. For example, when the *Role Propagation* inference was performed in Example 3.3.2, three resolution steps where performed to eliminate positive occurrences of D_1 and D_2 from (3.35) and (3.36). The elimination of the forgetting symbol B could not resume until the three noted resolutions were performed.

3.3.2 The Method of Fame

The method of Fame follows the approach based on the Ackermann's lemma explained in Section 3.2.1. It was originally introduced for the description logic $\mathcal{ALCOI}$ [ZS15], the description logic extending $\mathcal{ALC}$ with nominals and inverse roles. Since we are interested only in $\mathcal{ALC}$ forgetting methods, we present a simplified version of the method. In this simplified version, we consider only the part of the method that operates on $\mathcal{ALC}$ axioms. The other part dealing with nominals is not considered. A run of the simplified version on $\mathcal{ALC}$ ontologies would compute the same result as the run of the full version of the method.

The method of Fame is a pipeline of four phases, the clausification, the normalization, the central, and the simplification phases. The output of one phase is input to the next one in the pipeline.

The first phase is the clausification phase. A process whereby the input ontology is rewritten in a clausal form using the following rules.

$$C \equiv D \quad \rightarrow \quad \neg C \sqcup D, \neg D \sqcup C \tag{3.43}$$

$$C \sqsubseteq D \quad \rightarrow \quad \neg C \sqcup D \tag{3.44}$$

The second stage is the normalization phase. In this phase, two further transformations are performed on the clauses obtained from the clausification phase. The first transforms the clauses to negation normal form (nnf) where negations are applied only to concept names. The second transforms the nnf clauses to conjunctive normal form.

The two phases together transform the input ontology to a normal form. By comparing this normal form to the normal form of the method of Lethe, we can see that they only differ on the structural transformation component. The method of Fame does not perform structural transformation, whereas the method of Lethe cannot be performed without it.

The following example illustrates the two explained phases of the method of Fame.

Example 3.3.4. *Consider an ontology $\mathcal{O}$ comprising the following axioms.*

$$A \equiv B \sqcup (C \sqcap D) \tag{3.45}$$

$$\forall r.A \sqsubseteq \forall r.B \tag{3.46}$$

The first equivalence axiom is transformed to two subsumption axioms $A \sqsubseteq B \sqcup (C \sqcap D)$ and $B \sqcup (C \sqcap D) \sqsubseteq A$. Then, each of the two subsumption axioms is rewritten in negation

normal form. The first subsumption axiom is transformed to the clause (3.47) below, and the second subsumption axiom is transformed to the clause (3.48).

$$:A \sqcup B \sqcup (C \sqcap D) \tag{3.47}$$

$$(:B \sqcap (:C \sqcup :D)) \sqcup A \tag{3.48}$$

In the same way, the axioms (3.46) is transformed to the following clause in negation normal form.

$$\exists r.:A \sqcup \forall r.B \tag{3.49}$$

The three clauses are then transformed to conjunctive normal form. The clause (3.49) is already in conjunctive normal form. The clauses (3.47) and (3.48) are converted to the following clauses in conjunctive normal form.

$$:A \sqcup B \sqcup C \tag{3.50}$$

$$:A \sqcup B \sqcup D \tag{3.51}$$

$$:B \sqcup A \tag{3.52}$$

$$:C \sqcup :D \sqcup A \tag{3.53}$$

The second phase of the method of Fame is the central phase. In this phase, symbols from the forgetting signature are eliminated one after the other. Each round eliminates one forgetting symbol, called the pivot. The output of one round is the input to the following one. In each round, the method performs two steps. The first step attempts to transform every pivot clause to a pivot-reduced form. A pivot clause is a clause in the ontology where the pivot occurs. A clause is in pivot reduced form if the pivot occurs only as a top level, possibly negated, literal. That is, the clause is in pivot reduced form if the pivot does not occur below role restrictions in the clause. For instance, Let the pivot be the concept name A, and consider the following clauses.

$$A \sqcup B \tag{3.54}$$

$$:A \sqcup B \tag{3.55}$$

$$B \sqcup \forall r.\forall r.A \tag{3.56}$$

$$B \sqcup \exists r.\forall r.A \tag{3.57}$$

Surfacing

$$\frac{C \sqsubseteq \forall r{:}D}{\forall r\ {:}C \sqsubseteq D}$$

where A is a forgetting symbol, A does not occur in C, and A occurs in D.

Figure 3.2: The rewrite rules of the method of Fame.

The clauses (3.54) and (3.55) are in A-reduced form because the pivot A occurs as a top level literal in the two clauses. The clauses (3.56) and (3.57) are not in A-reduced form because the pivot A occurs below role restrictions in the two clauses. The method of Fame uses the *Surfacing* rule in Figure 3.2 to convert pivot clauses to the pivot reduced form. The *Surfacing* rule can be used to transform a clause to pivot reduced form if the pivot occurs below universal role restriction. If the pivot occurs below nested role restrictions then the clause can be transformed to the pivot reduced form only if all the role restrictions in the nesting are universal. For example, the clause (3.56) can be transformed to A reduced form by two *Surfacing* operations. The output of the first one is the clause $\forall r\ {:}B \sqsubseteq \forall r{:}A$, and the output of the second one is $\forall r\ {:}\forall r\ {:}B \sqsubseteq A$, which is a clause in A reduced form. If a pivot occurs below an existential role restriction, then the *Surfacing* rule cannot be used, and the clause cannot be transformed to pivot reduced form. The second side condition of the rule prevents falling in infinite loops. To explain consider the following clause.

$$A \sqsubseteq \forall r{:}A \tag{3.58}$$

The clause (3.58) cannot be transformed to A reduced form, because the concept A occurs in the first disjunct, and below role restriction in the second disjunct. By ignoring the second side condition, the following clauses would be obtained $\forall r\ {:}A \sqsubseteq A$, which is not in A reduced form. This transformation can be repeated infinitely without reaching the required A reduced form. To prevent such situation, the second side condition of the rule must be enforced.

Example 3.3.5. *Consider the ontology $\mathcal{O}$ comprising the following two clauses.*

$$C \sqsubseteq \exists r{:}A \tag{3.59}$$

$$D \sqsubseteq \forall r{:}(\neg A \sqsubseteq B) \tag{3.60}$$

Suppose the concept name A is the pivot. The aim is to eliminate A from the two clauses above. The two clauses are not in A reduced form. Moreover, the Surfacing *rule cannot be performed on the first clause, because A occurs below existential role restriction. It can, however, be applied on the second clause, and compute the following clause,*

$$\forall r.\neg D \sqcup \neg A \sqcup B; \tag{3.61}$$

which is in A reduced form.

An ontology is said to be in pivot reduced form if one of the following is true.

1. All positive occurrences of the pivot are top level disjuncts in the clauses of the ontology.

2. All negative occurrences of the pivot are top level disjuncts in the clauses of the ontology.

If an ontology is in a pivot reduced form, then the second step of the phase can be performed. If the pivot occurs only positively in the ontology then we eliminate it by replacing it with $\top$. If the pivot occurs only negatively in the ontology then we eliminate it by replacing it with $\bot$. The two operations together are often referred to by *purification*.

If the pivot occurs with positive polarity in in some clauses, and negative polarity in other clauses then we eliminate the pivot as follows. Suppose the concept name A is the pivot, and the ontology is in A reduced form such that 1 is true. As such, all positive occurrences of A are top level disjuncts in the clauses of the ontology. More precisely, the concept A occurs with positive polarity only in clauses of the form $A \sqcup C_i$ where $i \quad 1$. The concept A is eliminated by the transformation (3.17) of the Ackermann's lemma applied from left to right. The forgetting view is then the following ontology where all occurrences of A are replaced with the concept $(\neg C_1 \sqcup \quad \sqcup \neg C_n)$.

$$\mathcal{O}[A=(\neg C_1 \sqcup \quad \sqcup \neg C_n)] \tag{3.62}$$

Alternatively, suppose the ontology is in A reduced form such that 2 is true. As such, all negative occurrences of A are top level disjuncts in the clauses of the ontology. More precisely, the concept A occurs with negative polarity only in clauses of the form $\neg A \sqcup C_i$ where $i \quad 1$. The concept A is eliminated by the transformation (3.18) of the Ackermann's lemma applied from left to right. The forgetting view is then the

following ontology where all occurrences of A are replaced with $(:C_1 \sqcap \cdots \sqcap :C_n)$.

$$\mathcal{O}[A=(:C_1 \sqcap \cdots \sqcap :C_n)] \tag{3.63}$$

Example 3.3.6. *Continuing with Example 3.3.5. The ontology $\mathcal{O}$ has the following clauses.*

$$C \sqcap \exists r.A \tag{3.64}$$
$$\forall r.\neg D \sqcap \neg A \sqcap B \tag{3.65}$$

where (3.65) is obtained by the Surfacing *rule. The ontology $\mathcal{O}$ is in A reduced form, because A occurs negatively only in (3.65) where it is a top level disjunct. The concept A is eliminated by substituting $\forall r.\neg D \sqcap B$ for A in all clauses of the ontology. Consequently, we obtain:*

$$C \sqcap \exists r.(\forall r.\neg D \sqcap B) \tag{3.66}$$
$$(\forall r.\neg D \sqcap B) \sqcap \neg(\forall r.\neg D \sqcap B) \tag{3.67}$$

Since the clause (3.67) is a tautology, it is removed. The semantic forgetting view of $\mathcal{O}$ with respect to the forgetting symbol $\{A\}$ is therefore the ontology consisting of the clause (3.66).

There are cases where the ontology cannot be rewritten in pivot reduced form, and the transformations of the Ackermann's lemma cannot be consequently performed. There are two cases where this can occur. The first is when the *Surfacing* rule cannot be applied by the second side condition. The second is when the pivot occurs in at least two clauses, where the pivot occurs with negative polarity below existential role restriction in one of the clauses, and with positive polarity below existential role restriction in the other one. In these two cases, the method of Fame fails to write the ontology in pivot reduced form, and the fails to obtain the semantic forgetting view. This is however inevitable and expected, given the existence results of the forgetting view that we discussed earlier.

3.4 Applications of Forgetting

We explain some of the modern applications where forgetting has been used.

Ontology Creation: Ontologies are usually developed incrementally. An ontology engineer would start with a snapshot of the ontology in its most recent state. Then, the new axioms are developed, and a set of validations would run on the new axioms together with the snapshot to make sure they update the meaning of the ontology in the intended way [LLA+21]. When validations have passed correctly, a new version of the ontology comprising the original snapshot as well as the new axioms is published. If the snapshot of the ontology used throughout this process is large, validations can become a bottleneck to the process [CAS+19, ASDG21, DSG22, DPSG+23]. Forgetting allows for extracting the subset of the knowledge that is sufficient to perform the required validations. By eliminating the remaining knowledge, validations and other reasoning tasks become feasible and more efficient. For instance, when developing new axioms in the SNOMED-CT ontology regarding dental procedures, it would be more efficient to use a snapshot where the knowledge regarding knee procedures has been eliminated.

Ontology Reuse: Ontology-based systems often reuse existing ontologies, rather than building their ontologies from scratch [GHKS07, GHKS08, SSZ09, KLWW09, KWZ10, GKW14]. Obvious savings in time and cost would be gained by reusing ontologies. However, as only subsets of the existing ontologies might be relevant to the applications addressed by an ontology-based system, much of the knowledge of encoded by the ontology would be irrelevant. Not only will reasoning slow down because more data would be unnecessarily processed, storage and maintenance costs will be incurred. Forgetting offers a solution to the problem. By eliminating the irrelevant symbols from the ontology, and preserving the knowledge over the relevant ones, only the relevant subset of knowledge can be reused by the system.

Information Hiding: Data privacy is a challenging requirement for systems that encodes confidential information in their ontology or knowledge bases. Examples of such systems are ontology-based medical and financial systems. When a view of the ontology or the knowledge base is shared or queried, confidential information should not be revealed to users with restricted privileges [MS04, Swe02, CG10, CGM10]. One possible solution to the problem is using a forgetting view where the confidential information has been forgotten.

Abduction: Abductive reasoning computes a set of hypotheses that explain a given observation in the context of a background ontology. The problem is often parameterized with a signature S, such that only the hypotheses that can be expressed over this signature are computed. A forgetting-based method has been proposed for abduction

reasoning in the context of first-order logic [Lin00, DŁS01]. Let $\mathcal{O}$ be an ontology, and ψ an observation. The aim is finding the hypotheses ϕ such that the following is true,

$$\mathcal{O} \cup \phi \models \psi; \tag{3.68}$$

In other words, the observations should follow from the union of the ontology and the hypotheses. The cited methods relied on the following formula, where the negation of the required hypotheses is entailed by the union of the ontology and the negation of the observation.

$$\mathcal{O} \cup \neg\psi \models \neg\phi \tag{3.69}$$

Forgetting was used to eliminate the symbols of $\mathcal{O} \cup \neg\psi$ that were not in $\mathbf{S}$. Since the output of forgetting is the strongest consequence of $\mathcal{O} \cup \neg\psi$ that does not use symbols from $\mathbf{S}$, the formula $\neg\phi$ is necessarily encoded in the forgetting view. It was found however that irrelevant information that does not explain the observation is also encoded in the forgetting view. Distilling the correct hypotheses from the forgetting view is a problem on its own right. Recently, forgetting-based abduction systems were introduced in the context of description logics [DPS19a, DPS19b, DP22]. The systems integrate the forgetting process with a filtering method, and obtain just the wanted hypotheses.

Updating agent beliefs: The beliefs of an agent, in multi-agent environments, are continuously updated. For instance, existing beliefs of an agent are often updated after learning about factual changes, and new beliefs can be acquired through the communication with other agents.

Updating existing beliefs requires forgetting old beliefs and learning new ones [LLM03, DHLM08]. In this process we can see that forgetting is a central component.

Acquiring new beliefs in multi-agent environment has been considered in [TSP22, DŁS01]. Suppose agent 1 needs to communicate information to agent 2 over the set of symbols $\mathbf{S}_1$. If agent 2 only understands a small subset $\mathbf{S}_2$ of the symbols in $\mathbf{S}_1$, then communicating the information becomes a challenging task. There are two solutions to this problem, and in both of them forgetting is central. One solution is forgetting the symbols from $\mathbf{S}_1 \cap \mathbf{S}_2$, and communicating the forgetting view to agent 2. Although the forgetting view would not be as strong as the original information, it may suffice in many cases [DŁS01]. The second solution is explaining the unknown concepts to

agent 2. One approach to explain the concepts was suggested in [TSP22]. The approach uses forgetting to compute the explanation by utilizing the notions of strongest necessary conditions and weakest sufficient conditions.

3.5 Formal Background and Definitions

In this section, we group together the basic definitions and lemmas that we use in the presentations and discussions in the remainder of the thesis. The proofs of the lemmas in this section could not be found in other sources. As such, they are a contribution of this thesis.

We use standard syntactic and semantic definitions of entailment.

Definition 3.5.1. *Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two ontologies. The following statements are equivalent.*

1. $\mathcal{O}_1$ entails $\mathcal{O}_2$, *in symbols* $\mathcal{O}_1 \models \mathcal{O}_2$.
2. $\mathcal{O}_1 \models \alpha$ *for every axiom* $\alpha \in \mathcal{O}_2$.
3. $\mathcal{I} \models \mathcal{O}_1$ *implies* $\mathcal{I} \models \mathcal{O}_2$, *for every interpretation* $\mathcal{I}$.

Two ontologies are logically equivalent if each entails the other.

Definition 3.5.2. *Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two ontologies. We say $\mathcal{O}_1$ and $\mathcal{O}_2$ are* logically equivalent, *in symbols $\mathcal{O}_1 \equiv \mathcal{O}_2$ if and only if $\mathcal{O}_1 \models \mathcal{O}_2$ and $\mathcal{O}_2 \models \mathcal{O}_1$.*

Next, we define the three notions of inseparability which are model inseparability, query inseparability, and deductive inseparability. The provided definitions are inline with definitions provided in [LW10, BKL+17]. First, we define a model coincidence relation which we will use after to define model inseparability.

Definition 3.5.3. *Two models $\mathcal{I}$ and $\mathcal{J}$* S-coincide *if and only if $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ and $p^{\mathcal{I}} = p^{\mathcal{J}}$ for every concept or role symbol $p \in \mathbf{S}$.*

We use the S-coincidence relation to define *model inseparability* as follows.

Definition 3.5.4. *Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two ontologies and $\mathbf{S}$ be a set of concept names and roles. The two ontologies are* model inseparable with respect to $\mathbf{S}$, *in symbols $\mathcal{O}_1 \overset{M}{\equiv}_{\mathbf{S}} \mathcal{O}_2$, if for every interpretation $\mathcal{I}$ the following is true.*

$\mathcal{I} \models \mathcal{O}_1$ *if and only if there is a model $\mathcal{J}$ of $\mathcal{O}_2$, such that $\mathcal{I}$ and $\mathcal{J}$* S-coincide. (3.70)

If $\mathcal{I}$ is a model of $\mathcal{O}_1$ such that (3.70) is not satisfied, then the two ontologies are model separable*, and $\mathcal{I}$ is a* witness*.*

Two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ are *query inseparable with respect to* $\mathbf{S}$ if they coincide on all answers to all conjunctive queries over $\mathbf{S}$ against arbitrary ABoxes.

Definition 3.5.5. *Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two ontologies, and $\mathbf{S}$ be a set of concept names and roles. The two ontologies are* query inseparable with respect to $\mathbf{S}$*, in symbols $\mathcal{O}_1 \overset{Q}{\equiv}_{\mathbf{S}} \mathcal{O}_2$ if for every conjunctive query $q(\vec{x})$ such that $sig(q) \subseteq \mathbf{S}$, and every ABox $\mathcal{A}$ such that $sig(\mathcal{A}) \subseteq \mathbf{S}$ the following is true.*

$$(\mathcal{O}_1, \mathcal{A}) \models q(\vec{a}) \text{ if and only if } (\mathcal{O}_2, \mathcal{A}) \models q(\vec{a}); \tag{3.71}$$

where $\vec{a}$ is an answer to $q(\vec{x})$ comprising individuals from $\mathcal{A}$. If an answer to a query q is obtainable from one of the ontologies together with the ABox, but not obtainable from the knowledge base consisting of the other ontology and the ABox, then $\mathcal{O}_1$ and $\mathcal{O}_2$ are query separable*, and q is a* witness*.*

The two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ are deductively inseparable with respect to a signature $\mathbf{S}$ if they coincide on all $\mathcal{L}$ axioms over $\mathbf{S}$.

Definition 3.5.6. *Let $\mathcal{L}$ be a logic, $\mathbf{S}$ a set of concept names and roles, and $\mathcal{O}_1$ and $\mathcal{O}_2$ two ontologies in $\mathcal{L}$. The two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ are* deductively inseparable with respect to a signature $\mathbf{S}$*, in symbols $\mathcal{O}_1 \overset{C}{\equiv}_{\mathbf{S}} \mathcal{O}_2$, if for every $\mathcal{L}$ axiom α over $\mathbf{S}$, we have:*

$$\mathcal{O}_1 \models \alpha \text{ if and only if } \mathcal{O}_2 \models \alpha \tag{3.72}$$

If α is a consequence of one of the two ontologies, but not a consequence of the other, then the two ontologies are deductively separable *and α is a* witness*.*

We prove a set of lemmas about inseparability. The lemmas will be useful in the proofs presented throughout the thesis.

Lemma 3.5.7. *The relations $\overset{M}{\equiv}$, $\overset{Q}{\equiv}$, and $\overset{C}{\equiv}$ are equivalence relations.*

Proof. To show that the relations $\overset{M}{\equiv}$, $\overset{Q}{\equiv}$, and $\overset{C}{\equiv}$ are equivalence relations, we need to prove their symmetry, transitivity, and reflexivity properties. The proofs given here are for model inseparability. The same proofs are extendable in the straightforward way to deductive and query inseparability.

Firstly, we show the symmetry property of model inseparability. Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two ontologies and S is a signature. Suppose $\mathcal{O}_1 \overset{M}{\equiv}_{\mathsf{S}} \mathcal{O}_2$ but $\mathcal{O}_2 \not\overset{M}{\equiv}_{\mathsf{S}} \mathcal{O}_1$. From the latter assumption one of the following must be true.

1. There is a model $\mathcal{I}$ of $\mathcal{O}_2$, such that for every model $\mathcal{J}$ of $\mathcal{O}_1$ we have that $\mathcal{I}$ and $\mathcal{J}$ do not S-coincide.

2. There is a model $\mathcal{I}$ of $\mathcal{O}_1$, such that for every model $\mathcal{J}$ of $\mathcal{O}_2$ we have that $\mathcal{I}$ and $\mathcal{J}$ do not S-coincide.

If any of the two conditions is true, then the first assumption that $\mathcal{O}_1 \overset{M}{\equiv}_{\mathsf{S}} \mathcal{O}_2$ is incorrect. Therefore, $\mathcal{O}_1 \overset{M}{\equiv}_{\mathsf{S}} \mathcal{O}_2$ implies $\mathcal{O}_2 \overset{M}{\equiv}_{\mathsf{S}} \mathcal{O}_1$. The same argument shows that $\mathcal{O}_2 \overset{M}{\equiv}_{\mathsf{S}} \mathcal{O}_1$ implies $\mathcal{O}_1 \overset{M}{\equiv}_{\mathsf{S}} \mathcal{O}_2$. So, $\overset{M}{\equiv}$ is symmetric.

Secondly, we show the transitivity property of model inseparability. Let $\mathcal{O}_1$, $\mathcal{O}_2$, and $\mathcal{O}_3$ be three ontologies and S is a signature. Suppose the following is true.

$$\mathcal{O}_1 \overset{M}{\equiv}_{\mathsf{S}} \mathcal{O}_2 \tag{3.73}$$

$$\mathcal{O}_2 \overset{M}{\equiv}_{\mathsf{S}} \mathcal{O}_3 \tag{3.74}$$

We now have the following.

1. (3.73) implies that every model $\mathcal{I}$ of $\mathcal{O}_1$, has a corresponding model $\mathcal{J}$ of $\mathcal{O}_2$ such that $\mathsf{z}^{\mathcal{I}} = \mathsf{z}^{\mathcal{J}}$ for every $\mathsf{z} \in \mathsf{S}$.

2. (3.74) implies that every model $\mathcal{J}$ of $\mathcal{O}_2$, has a corresponding model $\mathcal{M}$ of $\mathcal{O}_3$ such that $\mathsf{z}^{\mathcal{M}} = \mathsf{z}^{\mathcal{J}}$ for every $\mathsf{z} \in \mathsf{S}$.

3. Adding the conclusions of the two previous points together we get that every model $\mathcal{I}$ of $\mathcal{O}_1$, has a corresponding model $\mathcal{M}$ of $\mathcal{O}_3$ such that $\mathsf{z}^{\mathcal{I}} = \mathsf{z}^{\mathcal{M}}$ for every $\mathsf{z} \in \mathsf{S}$. In the same way, the reverse of this statement can be obtained. So, we have $\mathcal{O}_1 \overset{M}{\equiv}_{\mathsf{S}} \mathcal{O}_3$. Thus, model inseparability is a transitive relation.

Thirdly, we prove the reflexivity property of model inseparability. Let $\mathcal{O}$ be an ontology, and S a signature. The requirement is to prove that $\mathcal{O} \overset{M}{\equiv}_{\mathsf{S}} \mathcal{O}$. This is true if for every model $\mathcal{I}$ of $\mathcal{O}$ there is a model $\mathcal{J}$ such that $\mathcal{I}$ and $\mathcal{J}$ S coincide. But, this is always true because we can take $\mathcal{J}$ to be the model $\mathcal{I}$ itself. So model inseparability is a reflexive relation.

□

Lemma 3.5.8. *Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two $\mathcal{ALC}$ ontologies, and let S a subset of the symbols in $\mathcal{O}_1$ and $\mathcal{O}_2$. We have, $\mathcal{O}_1 \equiv \mathcal{O}_2$ implies $\mathcal{O}_1 \equiv^{M}_{\mathsf{S}} \mathcal{O}_2$.*

Proof. We prove the lemma by contradiction. Assume $\mathcal{O}_1 \equiv \mathcal{O}_2$, and $\mathcal{O}_1 \not\equiv^{M}_{\mathsf{S}} \mathcal{O}_2$. The latter assumption is true if one of the following is true.

1. There is a model $\mathcal{I}$ of $\mathcal{O}_1$ such that for every model $\mathcal{J}$ of $\mathcal{O}_2$, $\mathcal{I}$ and $\mathcal{J}$ do not S-coincide.

2. There is a model $\mathcal{I}$ of $\mathcal{O}_2$ such that for every model $\mathcal{J}$ of $\mathcal{O}_1$, $\mathcal{I}$ and $\mathcal{J}$ do not S-coincide.

Suppose 1 is true. Since $\mathcal{O}_1 \equiv \mathcal{O}_2$, we have that $\mathcal{I} \models \mathcal{O}_2$. We choose $\mathcal{J}$ to be the model $\mathcal{I}$. The condition 1 is then incorrect because $\mathcal{I}$ coincides with itself on all interpretations for any signature S. In the same way, we see that the condition 2 is incorrect. Thus, $\mathcal{O}_1 \equiv \mathcal{O}_2$ implies $\mathcal{O}_1 \equiv^{M}_{\mathsf{S}} \mathcal{O}_2$. □

The following two lemmas shows that model inseparability is a stronger notion than deductive and query inseparability.

Lemma 3.5.9. *Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two $\mathcal{ALC}$ ontologies, and let S a set of symbols. We have, $\mathcal{O}_1 \equiv^{M}_{\mathsf{S}} \mathcal{O}_2$ implies $\mathcal{O}_1 \equiv^{C}_{\mathsf{S}} \mathcal{O}_2$.*

Proof. We prove the lemma by contradiction. Suppose $\mathcal{O}_1 \equiv^{M}_{\mathsf{S}} \mathcal{O}_2$, and let C and D be two $\mathcal{ALC}$ concepts over S such that one of the two following conditions is true.

1. $\mathcal{O}_1 \models C \sqsubseteq D$, and $\mathcal{O}_2 \not\models C \sqsubseteq D$.

2. $\mathcal{O}_2 \models C \sqsubseteq D$, and $\mathcal{O}_1 \not\models C \sqsubseteq D$.

Assume 1 is true. Since $\mathcal{O}_1 \models C \sqsubseteq D$, we have that $\mathcal{I} \models C \sqsubseteq D$ for every model $\mathcal{I}$ of $\mathcal{O}_1$. Since $\mathcal{O}_1 \equiv^{M}_{\mathsf{S}} \mathcal{O}_2$, $sig(C) \subseteq \mathsf{S}$, and $sig(D) \subseteq \mathsf{S}$, we have $\mathcal{I} \models C \sqsubseteq D$ for every model $\mathcal{I}$ of $\mathcal{O}_2$. Therefore $\mathcal{O}_2 \models C \sqsubseteq D$.

In the same way we can show that if the second condition above is true, then $\mathcal{O}_1 \models C \sqsubseteq D$. Altogether, $\mathcal{O}_1 \equiv^{M}_{\mathsf{S}} \mathcal{O}_2$ implies $\mathcal{O}_1 \equiv^{C}_{\mathsf{S}} \mathcal{O}_2$. □

Lemma 3.5.10. *Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two $\mathcal{ALC}$ ontologies, and let S a set of symbols. We have, $\mathcal{O}_1 \equiv^{M}_{\mathsf{S}} \mathcal{O}_2$ implies $\mathcal{O}_1 \equiv^{Q}_{\mathsf{S}} \mathcal{O}_2$.*

Proof. Assume $\mathcal{O}_1 \equiv^{M}_{\mathsf{S}} \mathcal{O}_2$ and $\mathcal{O}_1 \not\equiv^{Q}_{\mathsf{S}} \mathcal{O}_2$. The latter assumption is true if one of the following is true.

$$(\mathcal{O}_1, \mathcal{A}) \models q(\vec{a}), \text{ and } (\mathcal{O}_2, \mathcal{A}) \not\models q(\vec{a}); \tag{3.75}$$

$$(\mathcal{O}_2, \mathcal{A}) \models q(\vec{a}), \text{ and } (\mathcal{O}_1, \mathcal{A}) \not\models q(\vec{a}); \tag{3.76}$$

where $\mathcal{A}$ is an arbitrary ABox, $q(\vec{x})$ is a conjunctive query, $\vec{x} = x_1, x_2, \ldots, x_k$ are the answer variables, $\vec{a} = a_1, a_2, \ldots, a_k$ are individuals in $\mathcal{A}$, and $k \geq 0$.

Suppose (3.76) is correct. Let $\mathcal{I}$ be a model of $(\mathcal{O}_1, \mathcal{A})$ such that $\mathcal{I} \not\models q(\vec{a})$. We have $\mathcal{I} \models \mathcal{O}_1$ and $\mathcal{I} \models \mathcal{A}$. Since $\mathcal{O}_1 \equiv^{M}_{\mathsf{S}} \mathcal{O}_2$, there is a model $\mathcal{J}$ of $\mathcal{O}_2$ such that $\mathcal{I}$ and $\mathcal{J}$ S-coincide. This model $\mathcal{J}$ must also be a model of $\mathcal{A}$ because $sig(\mathcal{A}) \subseteq \mathsf{S}$. So, $\mathcal{J}$ is a model of the knowledge base $(\mathcal{O}_2, \mathcal{A})$. Since $(\mathcal{O}_1, \mathcal{A}) \not\models q(\vec{a})$, there must be a mapping p that maps the variables in q to domain elements in $\mathcal{I}$ such that $\mathsf{p}(x_i) = a_i^{\mathcal{I}}$ for $1 \leq i \leq k$, and at least one of the following two conditions is true.

1. $A(u)$ is an atom in q, and $\mathsf{p}(u) \notin A^{\mathcal{I}}$.
2. $r(u, v)$ is an atom in q, and $(\mathsf{p}(u), \mathsf{p}(v)) \notin r^{\mathcal{I}}$,

where A is a concept name in S, and r is a role in S. Since $\mathcal{I}$ and $\mathcal{J}$ S-coincide, one of the above two conditions must be true in $\mathcal{J}$. But then $q(\vec{a})$ is not true in $\mathcal{J}$. The following two contradicting statements are now obtained.

1. $(\mathcal{O}_2, \mathcal{A}) \models q(\vec{a})$ by assumption, and
2. $(\mathcal{O}_2, \mathcal{A}) \not\models q(\vec{a})$ because $\mathcal{J} \models (\mathcal{O}_2, \mathcal{A})$ and $\mathcal{J} \not\models q(\vec{a})$.

From the contradiction we conclude that if $\mathcal{O}_1 \equiv^{M}_{\mathsf{S}} \mathcal{O}_2$ then $(\mathcal{O}_2, \mathcal{A}) \models q(\vec{a})$ implies $(\mathcal{O}_1, \mathcal{A}) \models q(\vec{a})$. In the same way we can conclude that if $\mathcal{O}_1 \equiv^{M}_{\mathsf{S}} \mathcal{O}_2$ then $(\mathcal{O}_1, \mathcal{A}) \models q(\vec{a})$ implies $(\mathcal{O}_2, \mathcal{A}) \models q(\vec{a})$. Altogether, we see that $\mathcal{O}_1 \equiv^{M}_{\mathsf{S}} \mathcal{O}_2$ implies $\mathcal{O}_1 \equiv^{Q}_{\mathsf{S}} \mathcal{O}_2$. □

Lemma 3.5.11. *Deductive forgetting and query forgetting are not related.*

Proof. We showed in Examples 3.1.7 and 3.1.8 that deductive forgetting with respect to a signature S does not imply query forgetting with respect to the same signature. We show in the following example that query forgetting does not imply deductive forgetting.

Consider the ontology $\mathcal{O} = \{A \sqsubseteq B, B \sqsubseteq C \sqcup D\}$, and let $\mathsf{F} = \{B\}$ be a forgetting signature. The empty ontology $\mathcal{V}^q = \{\}$ is a query forgetting views of $\mathcal{O}$ with respect to F, and its correctness can be seen as follows. Let $\mathcal{A}$ be an arbitrary ABox over

$A;C$, and D. Since A can only have positive concept assertions, inferences between A and O can only be performed with respect to the negative concepts in O that are not in F. That is, the concept A. Suppose $A = fA(a)g$ is an ABox, the only interesting conjunctive queries are $q_1(x) = C(x)$ and $q_2(x) = D(x)$. But neither of them has an answer in $(O;A)$, which is also true for $(V^q;A)$. So, V^q is a query forgetting view of O with respect to F. But, V^q is not a deductive forgetting view of O with respect to F because $V^q \not\models A \vee C \sqcup D$ whereas $O \models A \vee C \sqcup D$. □

Lemma 3.5.12. *Let O_1 and O_2 be two ontologies, and S_1 and S_2 two sets of symbols such that $S_1 \subseteq S_2$. We have, $O_1 \equiv^M_{S_2} O_2$ implies $O_1 \equiv^M_{S_1} O_2$.*

Proof. Suppose $O_1 \equiv^M_{S_2} O_2$. From Definition 3.5.4 we have that for model I of O_1 there is a model J of O_2 and vice versa such that $z^I = z^J$ for every $z \in S_2$. Since $S_1 \subseteq S_2$, the above must also be true for every $z \in S_1$. Therefore, we must have $O_1 \equiv^M_{S_1} O_2$. □

Lemma 3.5.13. *Let O_1 and O_2 be two ontologies, and S_1 and S_2 two sets of symbols such that $S_1 \subseteq S_2$. We have, $O_1 \equiv^C_{S_2} O_2$ implies $O_1 \equiv^C_{S_1} O_2$*

Proof. Suppose $O_1 \equiv^C_{S_2} O_2$. Denote by G_2 the set of axioms $C \sqsubseteq D$ over S_2. Since $O_1 \equiv^C_{S_2} O_2$, we have for every $a \in G_2$:

$$O_1 \models a \qquad \text{if and only if} \qquad O_2 \models a \tag{3.77}$$

Let G_1 be the set of axioms $C' \sqsubseteq D'$ over S_1. The set G_1 is a subset of the set G_2 because $S_1 \subseteq S_2$. So, (3.77) is true for every $a \in G_1$. In other words, $O_1 \equiv^C_{S_1} O_2$. □

Now that we have defined inseparability and stated important properties of them, we define the three notions of forgetting that will be used in the remainder of this thesis.

Definition 3.5.14. *Let O be an ALC ontology, and let $F \subseteq sig(O) \setminus N_c$ be a forgetting signature. An ontology V is a* deductive forgetting view, *or* deductive view *for short, of O with respect to F if and only if the following two conditions are true.*

1. *V does not use symbols from F, in symbols $sig(V) \subseteq sig(O) \cap F$.*
2. *O and V are deductively inseparable with respect to the symbols in $sig(O) \cap F$.*

Definition 3.5.15. *Let O be an ALC ontology, and let $F \subseteq sig(O) \setminus N_c$ be a forgetting signature. An ontology V is a* query forgetting view, *or* query view *for short, of O with respect to F if and only if the following two conditions are true.*

1. $\mathcal{V}$ *does not use symbols from* $\mathbf{F}$*, in symbols* $sig(\mathcal{V}) \cap sig(\mathcal{O}) \cap \mathbf{F}$.
2. $\mathcal{O}$ *and* $\mathcal{V}$ *are query inseparable with respect to the symbols in* $sig(\mathcal{O}) \cap \mathbf{F}$.

Definition 3.5.16. *Let* $\mathcal{O}$ *be an* $\mathcal{ALC}$ *ontology, and let* $\mathbf{F} \subseteq sig(\mathcal{O}) \setminus N_c$ *be a forgetting signature. An ontology* $\mathcal{V}$ *is a* semantic forgetting view*, or* semantic view *for short, of* $\mathcal{O}$ *with respect to* $\mathbf{F}$ *if and only if the following two conditions are true.*

1. $\mathcal{V}$ *does not use symbols from* $\mathbf{F}$*, in symbols* $sig(\mathcal{V}) \cap sig(\mathcal{O}) \cap \mathbf{F}$.
2. $\mathcal{O}$ *and* $\mathcal{V}$ *are model inseparable with respect to the symbols in* $sig(\mathcal{O}) \cap \mathbf{F}$.

As noted earlier, the forgetting methods in Chapters 4, 5, and 6 may use second-order symbols, or definers, to obtain a finite form of the forgetting views. As such, the first condition of Definitions 3.5.14, 3.5.15, and 3.5.16, would not be satisfied by the output of the methods if definers were used. In this case, the computed ontology is not, strictly speaking, a forgetting view, but a representation of it. Suppose $\mathcal{O}$ is an ontology, and $\mathbf{F}$ is a forgetting signature. A representation of the forgetting view is a finite ontology $\mathcal{V}$ such that $\mathbf{F} \subseteq sig(\mathcal{V})$, and the set of consequences of $\mathcal{V}$ over $sig(\mathcal{O}) \cap \mathbf{F}$ is inseparable from the, possibly infinite, forgetting view of $\mathcal{O}$ with respect to $\mathbf{F}$.

Chapter 4

Fine-grained Forgetting

We build a basic fine-grained forgetting framework that will be the basis of our research. In its basic form, the framework accepts two inputs, an $\mathcal{ALC}$ ontology, and a forgetting signature consisting of the concept names to be forgotten.

The framework computes a representation of the semantic forgetting views of the input ontology with respect to the forgetting signature. The novel form of this representation allows customization processes to extract forgetting views of customized content. In this chapter, we present a deductive customization process that consumes the representation of the semantic forgetting view and generates a deductive forgetting view of the input ontology with respect to the forgetting signature. The customization also computes auxiliary axioms indicating the difference between the content of the obtained deductive forgetting view and the semantic forgetting views of the input ontology with respect to the forgetting signature. We describe a process though which users can increment the final deductive forgetting view with some of the auxiliary axioms to improve the informativeness of the final forgetting views according to their requirements, a process that reveals a spectrum of fine-grained forgetting views in-between the deductive and the semantic forgetting views. In the next chapter, we will present another customization process that consumes the representation of the semantic forgetting view and generates a query forgetting view of the input ontology with respect to the forgetting signature.

Recall from Chapter 3 that deductive forgetting views of $\mathcal{ALC}$ ontologies may not exist. Moreover, it is not possible to capture a deductive forgetting view in some cases because of the high complexity of forgetting. In these situations, our framework will generate an approximation to the deductive forgetting views of the input ontology with respect to the forgetting signature.

Two main research topics will be addressed in this chapter.

1. The auxiliary information obtained when extracting the deductive forgetting view will aid our understanding of the information not preserved by deductive forgetting views.

2. The separation of the customization processes from the elimination of the forgetting symbols allows the users to eliminate the forgetting symbols at one time and extract the required forgetting view at a later time. This addresses the use case explained in Section 1.2 where forgetting is needed but the variant of forgetting is only known in a later stage.

3. By computing fine-grained forgetting views in-between the deductive and the semantic forgetting views, we allow a new paradigm of forgetting where the user has control over the content of the forgetting view. This is different from the existing usage-based paradigm where the user can only specify the reasoning tasks where the forgetting view is used.

Throughout this work, We assume that the input ontology is consistent.

The Chapter is structured as follows. Section 4.1 outlines the forgetting framework as a process of three consecutive stages. Section 4.2 describes the normal form used by our framework. The three stages of our framework will be developed in Sections 4.3, 4.4, and 4.5 respectively. Cases, where an approximation to the deductive forgetting views is obtained, are presented and discussed in Section 4.6. Section 4.7 discusses the output obtained in the three stages, and compares our representation of the semantic forgetting views against the output of the method of [LR94a]. Finally, Section 4.8 shows the extraction of fine-grained views using our forgetting framework.

4.1 A Three-Stage Forgetting Framework

Our forgetting framework runs in three stages presented in Figure 4.1.

The first stage is the *Resolution* stage. The inputs of the stage are an ontology $\mathcal{O}$, and a forgetting signature $\mathbf{F}$ consisting of concept names. The output of the stage is an $\mathcal{ALC}$ ontology $\mathcal{O}^{int}$ called the *intermediate ontology*. $\mathcal{O}^{int}$ is a representation of the semantic forgetting views of $\mathcal{O}$ with respect to $\mathbf{F}$. It does not use the forgetting symbols from $\mathbf{F}$, and it is model-inseparable from the input ontology $\mathcal{O}$ with respect to the symbols $sig(\mathcal{O}) \setminus \mathbf{F}$. If $\mathcal{O}^{int}$ is expressed over the non-forgotten vocabulary $sig(\mathcal{O}) \setminus \mathbf{F}$,

then $\mathcal{O}^{int}$ is a semantic forgetting view of $\mathcal{O}$ with respect to $\mathcal{F}$ by Definition 3.5.16. However, it is known from the literature that the semantic forgetting views of $\mathcal{ALC}$ ontologies may not exist [LR94a, ZZ09, LW11]. To ensure a representation, $\mathcal{O}^{int}$ is expressed in a normal form that allows for introducing auxiliary concept names, called definers, in $\mathcal{O}^{int}$.

The second and third stages together are responsible for customizing the final forgetting view. The second stage is the *Deductive Reduction* stage. The input of this stage is the intermediate ontology $\mathcal{O}^{int}$. The outputs are two sets of clauses, a reduced ontology $\mathcal{O}^{d}$, and a set D^{d}. The reduced ontology $\mathcal{O}^{d}$ is deductively inseparable from the $\mathcal{O}^{int}$ with respect to the non-forgotten vocabulary $sig(\mathcal{O}) \cap \mathcal{F}$. The set D^{d} represents the content difference between $\mathcal{O}^{int}$ and $\mathcal{O}^{d}$.

The third stage is the *Definer Elimination* stage. The input to this stage is the reduced ontology $\mathcal{O}^{d}$. The purpose of this stage is to eliminate the definers from $\mathcal{O}^{d}$, and obtain a final deductive forgetting view $\mathcal{V}^{d}$ in $\mathcal{ALC}$. The ontology $\mathcal{V}^{d}$ is model inseparable from $\mathcal{O}^{d}$, and deductively inseparable from $\mathcal{O}^{int}$ and $\mathcal{O}$, with respect to the signature $sig(\mathcal{O}) \cap \mathcal{F}$. If definers have been completely eliminated from $\mathcal{V}^{d}$, then $\mathcal{V}^{d}$ is a deductive forgetting view of $\mathcal{O}$ with respect to $\mathcal{F}$. When the deductive forgetting views of $\mathcal{O}$ with respect to $\mathcal{F}$ do not exist, or when it is impractical to capture them due to space and computation complexities, some definers may remain in $\mathcal{V}^{d}$. In this case, $\mathcal{V}^{d}$ is an approximation of the deductive forgetting views of $\mathcal{O}$ with respect to $\mathcal{F}$. We will show that in this case every model of $\mathcal{V}^{d}$ is also a model of a deductive view of $\mathcal{O}$ with respect to $\mathcal{F}$.

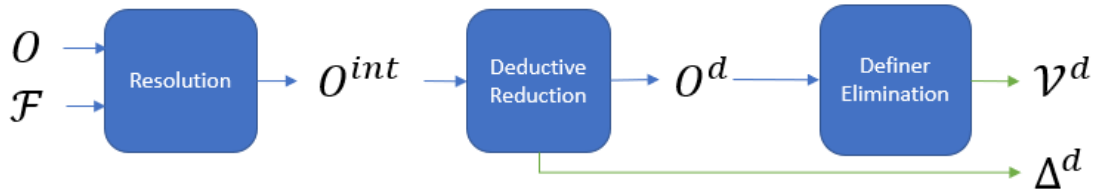

Figure 4.1: Fine-grained forgetting framework.

4.2 The Normal Form

Our method requires normalizing the input ontology in a normal form which we specify here. The output of this normalization is an ontology $\mathcal{O}^{clausal}$.

An ontology is in normal form if all its axioms where a forgetting symbol occurs are in normal form. An axiom that contains a forgetting symbol B is in normal form if it takes the form $\top \sqsubseteq B \sqcup C$ or the form $\top \sqsubseteq \neg B \sqcup C$ where C is a concept. That is, if B occurs in the axiom as a top-level literal. This makes forgetting symbols accessible to the calculus used to obtain $\mathcal{O}^{int}$.

Suppose α is an $\mathcal{ALC}$ axiom. A normal form of α is obtained by applying the following in sequence.

1. Negation Normal Form (NNF).
2. Prenexing
3. Structural transformation
4. Conjunctive Normal Form (CNF).

The above transformations are performed using the recursive rules in Table 4.1.

The rules $(nnf\ 1)$ and $(nnf\ 2)$ convert the axioms, where forgetting symbols occur, to clauses of the form $\top \sqsubseteq C$ where C is a concept. The remaining NNF rules are applied on concepts rather than axioms. They push negation inwards, such that if a forgetting symbol B occurs negatively in an axiom, then it occurs directly under a single negation. Our NNF transformation is non-standard in that it only applies to concepts (or axioms) where a forgetting symbol is used. For example, if B is a forgetting symbol and A does not use a forgetting symbol, we transform the axiom $\exists r.A \sqcap \exists r.B \sqsubseteq \bot$ to the clause $\top \sqsubseteq \neg \exists r.A \sqcup \forall r.\neg B$. Furthermore, an axiom $\exists r.A \sqsubseteq C$ will not be transformed if it does not contain a forgetting symbol. The axioms obtained in this example do not follow the negation normal form defined in classic sources, for instance [BUL01], but it suffices our purpose and avoids useless operations.

Prenexing optimizes the structural transformation step. It is convenient to explain it after introducing structural transformation.

Structural transformation is an important flattening technique in the applications which transform its input to conjunctive normal form. It effectively keeps the size of the clauses polynomial in the size of the input ontology which makes the application more efficient as small input is processed [PG86, dlT92, NW01]. The two structural transformation rules in Figure 4.1 transform a clause C, if a forgetting symbol $B \in F$ occurs under role restriction in C. They require occurrence of concepts of the forms $\exists r.E$ or $\forall r.E$ at a position i, where a forgetting symbol B occurs in E. When these conditions are satisfied, structural transformation renames the concept E to D, and

Negation Normal Form (NNF)			
$C \equiv E$	$\rightarrow$	$\top \sqsubseteq \neg C \sqcup E; \top \sqsubseteq \neg E \sqcup C$	$(nnf\ 1)$
$C \sqsubseteq E$	$\rightarrow$	$\top \sqsubseteq \neg C \sqcup E$	$(nnf\ 2)$
$\neg(C \sqcup E)$	$\rightarrow$	$\neg C \sqcap \neg E$	$(nnf\ 3)$
$\neg(C \sqcap E)$	$\rightarrow$	$\neg C \sqcup \neg E$	$(nnf\ 4)$
$\neg \exists r.E$	$\rightarrow$	$\forall r.\neg E$	$(nnf\ 5)$
$\neg \forall r.E$	$\rightarrow$	$\exists r.\neg E$	$(nnf\ 6)$
$\neg\neg E$	$\rightarrow$	E	$(nnf\ 7)$
where C and E are concepts, $sig(C) \setminus F \neq \emptyset$, or $sig(E) \setminus F \neq \emptyset$.			
Prenexing			
$\exists r.C \sqcup \exists r.E$	$\rightarrow$	$\exists r.(C \sqcup E)$	$(pnx\ 1)$
$\forall r.C \sqcap \forall r.E$	$\rightarrow$	$\forall r.(C \sqcap E)$	$(pnx\ 2)$
where C and E are concepts, $sig(C) \setminus sig(E) \setminus F \neq \emptyset$.			
Structural Transformation			
C	$\rightarrow$	$C[i/\exists r.D]; \top \sqsubseteq \neg D \sqcup E$	$(st\ 1)$
where $C|_i = \exists r.E; B \in sig(E);$ and $B \in F$.			
C	$\rightarrow$	$C[i/\forall r.D]; \top \sqsubseteq \neg D \sqcup E$	$(st\ 2)$
where $C|_i = \forall r.E; B \in sig(E);$ and $B \in F$.			
Conjunctive Normal Form			
$\top \sqsubseteq C_1 \sqcap C_2$	$\rightarrow$	$\top \sqsubseteq C_1; \top \sqsubseteq C_2$	$(cnf\ 1)$
where $sig(C_1) \setminus (F \cup N_d) \neq \emptyset$, or $sig(C_2) \setminus (F \cup N_d) \neq \emptyset$			
$C_1 \sqcup (C_2 \sqcap C_3)$	$\rightarrow$	$(C_1 \sqcup C_2) \sqcap (C_1 \sqcup C_3)$	$(cnf\ 2)$
where $sig(C_2) \setminus (F \cup N_d) \neq \emptyset$, or $sig(C_3) \setminus (F \cup N_d) \neq \emptyset$			

Table 4.1: Normal form translation rules of $\mathcal{ALC}$ ontologies.

adds a *definition* clause $\neg D \sqcup E$ to the ontology. The concept name D must be fresh. That is, it is not being used in the ontology and it has not been introduced in previous structural transformations.

Definition 4.2.1. *We denote by N_d the set of concept names introduced by structural transformations, and call them* definers*.*

A definition *clause of a definer D is a clause $\neg D \sqcup E$ introduced when the concept E was renamed to D.*

Prenexing rewrites the concepts $\exists r.C \sqcup \exists r.E$ and $\forall r.C \sqcap \forall r.E$ to the logically equivalent $\exists r.(C \sqcup E)$ and $\forall r.(C \sqcap E)$, respectively (see Lemma 4.2.3 below). The side condition $sig(C) \setminus sig(E) \setminus F \neq \emptyset$ ensures that the number of definers, that will be introduced by structural transformation, is reduced. For example, the clause $C = \top \sqsubseteq$

$\exists r{:}B_1 \sqcup \exists r{:}B_2$ would be structurally transformed without prenexing to the following three clauses.

$$\exists r{:}D_1 \sqcup \exists r{:}D_2$$
$$\neg D_1 \sqcup B_1$$
$$\neg D_2 \sqcup B_2$$

where $B_1, B_2 \in \mathsf{F}$ are forgetting symbols, and $D_1, D_2 \in \mathsf{N}_d$ are definers. However, with prenexing, C is written as $\exists r{:}(B_1 \sqcup B_1)$, and it will be structurally transformed to the following two clauses where only one definer D_3 is introduced.

$$\exists r{:}D_3$$
$$\neg D_3 \sqcup B_1 \sqcup B_2$$

The rule (cnf 1) separates the conjuncts of a clause in new clauses if they contain a forgetting symbol or a definer. The rule (cnf 2) is the standard distribution law. Both rules are applied if their result allow for having the concepts $B, \neg B, \exists r{:}D, \forall r{:}D$, as a top-level disjuncts in a clause, where $B \in \mathsf{F}$ is a forgetting symbol, and $D \in \mathsf{N}_d$ is a definer. For instance, if $B \in \mathsf{F}$ is a forgetting symbol, the clause $\top \sqsubseteq A \sqcup (\neg B \sqcap C)$ would be transformed to the following two clauses by applying (cnf 2) then (cnf 1).

$$\top \sqsubseteq A \sqcup \neg B$$
$$\top \sqsubseteq A \sqcup C$$

However, the clause $\top \sqsubseteq \neg B \sqcup (A \sqcap C)$ will not be transformed because $\neg B$ already occurs as a top-level disjunct in the clause.

In addition to the rules in Table 4.1 we allow the following standard simplifications.

$$C \sqcap C \Rightarrow C \qquad C \sqcup C \Rightarrow C \qquad C \sqcap \neg C \Rightarrow \bot \qquad C \sqcup \neg C \Rightarrow \top$$
$$C \sqcap \top \Rightarrow C \qquad C \sqcup \top \Rightarrow \top \qquad C \sqcap \bot \Rightarrow \bot \qquad C \sqcup \bot \Rightarrow C$$

where C is a concept. The following example shows the transformation of an ontology $\mathcal{O}$ to $\mathcal{O}^{clausal}$.

Example 4.2.2. *Let $\mathcal{O}$ be the following ontology*

$$\forall r{:}\neg B \sqcup \exists r{:}(A \sqcap C) \tag{4.1}$$

$$A \sqcap C \sqsubseteq E \tag{4.2}$$

$$E \sqsubseteq F \tag{4.3}$$

where $F = \{B; C\}$ *is a forgetting signature.*

Only the axioms (4.1) and (4.2) will be translated to negation normal form because they contain forgetting symbols. Using $(nnf\ 2)$*, the axioms (4.1) and (4.2) are transformed to the following.*

$$\top \sqsubseteq \neg\exists r.\neg B \sqcup \exists r.(A \sqcap C) \tag{4.4}$$

$$\top \sqsubseteq \neg(A \sqcap C) \sqcup E \tag{4.5}$$

(4.4) and (4.5) are not yet in NNF. We transform (4.4) to NNF using $(nnf\ 6)$ *and* $(nnf\ 7)$ *consecutively, and (4.5) using* $(nnf\ 4)$*. This replaces (4.1) and (4.2) by the following two clauses.*

$$\top \sqsubseteq \forall r.B \sqcup \exists r.(A \sqcap C) \tag{4.6}$$

$$\top \sqsubseteq (\neg A \sqcap \neg C) \sqcup E \tag{4.7}$$

(4.6) is prenexed by $(pnx\ 1)$*, and replaced by the following clause.*

$$\top \sqsubseteq \forall r.(B \sqcup (A \sqcap C)) \tag{4.8}$$

Then, (4.8) is structurally transformed by $(st\ 1)$ *to the following two clauses.*

$$\top \sqsubseteq \forall r.D \tag{4.9}$$

$$\neg D \sqcup B \sqcup (A \sqcap C) \tag{4.10}$$

where $D \in N_d$ *is a fresh definer symbol, and the clause (4.10) is the definition of D.*

Finally, (4.7) and (4.10) are transformed to CNF using $(cnf\ 1)$ *and* $(cnf\ 2)$ *to obtain:*

$$\top \sqsubseteq \neg A \sqcup E \tag{4.11}$$

$$\top \sqsubseteq \neg C \sqcup E \tag{4.12}$$

$$\top \sqsubseteq \neg D \sqcup B \sqcup A \tag{4.13}$$

$$\top \sqsubseteq \neg D \sqcup B \sqcup C \tag{4.14}$$

$\mathcal{O}^{clausal}$ is now the following ontology.

$$\top \sqsubseteq \forall r.D \tag{4.15}$$
$$\top \sqsubseteq \neg D \sqcup B \sqcup A \tag{4.16}$$
$$\top \sqsubseteq \neg D \sqcup B \sqcup C \tag{4.17}$$
$$\top \sqsubseteq \neg A \sqcup E \tag{4.18}$$
$$\top \sqsubseteq \neg C \sqcup E \tag{4.19}$$
$$E \sqsubseteq F \tag{4.20}$$

The NNF transformation preserves logical equivalence in first-order logic (FOL). Since $\mathcal{ALC}$ is a fragment of FOL, logical equivalence is preserved in $\mathcal{ALC}$ as well. Prenexing and CNF transformations do not preserve logical equivalence in FOL because they use Skolemization which necessarily breaks logical equivalence [BUL01]. However, when performed in the way described before, they preserve logical equivalence. Lemma 4.2.3 below proves this claim.

Lemma 4.2.3. *Prenexing, and CNF transformation preserve logical equivalence.*

Proof. We perform prenexing using (*pnx* 1) and (*pns* 2), and CNF transformation using (*cnf* 1) and (*cnf* 2). For (*cnf* 1), we show that the ontologies $\mathcal{O}_1 = \{\top \sqsubseteq C_1 \sqcap C_2\}$ and $\mathcal{O}_2 = \{\top \sqsubseteq C_1; \top \sqsubseteq C_2\}$ are logically equivalent. First, we show $\mathcal{O}_1 \models \mathcal{O}_2$. Let $\mathcal{I}$ be a model of $\mathcal{O}_1$. For every domain element $d \in \Delta^{\mathcal{I}}$ we have that $d \in C_1^{\mathcal{I}}$ and $d \in C_2^{\mathcal{I}}$. This implies that $\mathcal{I} \models \top \sqsubseteq C_1$ and $\mathcal{I} \models \top \sqsubseteq C_2$. So, $\mathcal{I} \models \mathcal{O}_2$.

Second we show that $\mathcal{O}_2 \models \mathcal{O}_1$. Let $\mathcal{I}$ be a model of $\mathcal{O}_2$. For every domain element $d \in \Delta^{\mathcal{I}}$ we have that $d \in C_1^{\mathcal{I}}$ and $d \in C_2^{\mathcal{I}}$. So, $d \in (C_1 \sqcap C_2)^{\mathcal{I}}$. This means that $\mathcal{I} \models \top \sqsubseteq C_1 \sqcap C_2$. So, $\mathcal{I} \models \mathcal{O}_1$.

The rule (*cnf* 2) can directly be verified by observing that the Venn diagrams of the concepts $C_1 \sqcup (C_2 \sqcap C_3)$ and $(C_1 \sqcup C_2) \sqcap (C_1 \sqcup C_3)$ are identical. Figure 4.2 shows the Venn diagram of the two concepts.

For (*pnx* 1), we prove that

$$\models \forall r.C \sqcup \forall r.E \quad \equiv \quad \forall r.(C \sqcup E) \tag{4.21}$$

First, we prove that $\models \forall r.C \sqcup \forall r.E \sqsubseteq \forall r.(C \sqcup E)$ by contradiction. Let $\mathcal{I}$ be an interpretation, and $d \in \Delta^{\mathcal{I}}$ such that:

$$d \in (\forall r.C \sqcup \forall r.E)^{\mathcal{I}} \tag{4.22}$$

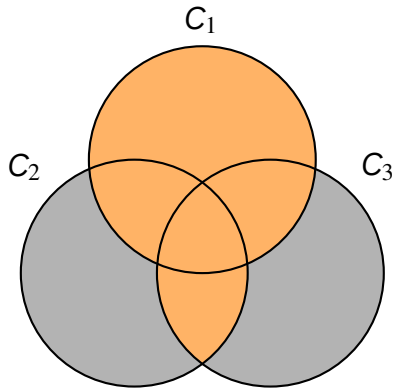

Figure 4.2: Venn diagram of the concepts $C_1 \sqcup (C_2 \sqcap C_3)$ and $(C_1 \sqcup C_2) \sqcap (C_1 \sqcup C_3)$.

$$d \notin (\exists r.(C \sqcup E))^{\mathcal{I}} \tag{4.23}$$

From (4.22), one or both of the following must be true in $\mathcal{I}$.

1. There exists $e_1 \in \Delta^{\mathcal{I}}$ such that $e_1 \in C^{\mathcal{I}}$ and $(d, e_1) \in r^{\mathcal{I}}$.
2. There exists $e_2 \in \Delta^{\mathcal{I}}$ such that $e_2 \in E^{\mathcal{I}}$ and $(d, e_2) \in r^{\mathcal{I}}$.

However, from (4.23), if e is an r-successor of d then $e \notin C^{\mathcal{I}}$ and $e \notin E^{\mathcal{I}}$, which contradicts the two cases above. So, $\models \exists r.C \sqcup \exists r.E \sqsubseteq \exists r.(C \sqcup E)$

Second, we prove by contradiction that $\models \exists r.(C \sqcup E) \sqsubseteq \exists r.C \sqcup \exists r.E$. Let $\mathcal{I}$ be an interpretation, and $d \in \Delta^{\mathcal{I}}$ such that:

$$d \in (\exists r.(C \sqcup E))^{\mathcal{I}} \tag{4.24}$$

$$d \notin (\exists r.C \sqcup \exists r.E)^{\mathcal{I}} \tag{4.25}$$

From (4.24), there must be an $e \in \Delta^{\mathcal{I}}$ such that $(d, e) \in r^{\mathcal{I}}$, where e is in the interpretation of C, E, or both. From (4.25), every r-successor e of d must not be in the interpretations of C and E, which contradicts the previous condition. So, $\models \exists r.(C \sqcup E) \sqsubseteq \exists r.C \sqcup \exists r.E$.

For (*pnx* 2), we prove that $\models \forall r.C \sqcap \forall r.E \equiv \forall r.(C \sqcap E)$. First we show that $\models \forall r.C \sqcap \forall r.E \sqsubseteq \forall r.(C \sqcap E)$ by proving its contrapositive.

$$\neg \forall r.(C \sqcap E) \sqcup \neg (\forall r.C \sqcap \forall r.E) \tag{4.26}$$

By pushing the negations in (4.26) inwards, we need to prove:

$$\exists r.(\neg C \sqcup \neg E) \sqcup \exists r.\neg C \sqcup \exists r.\neg E \tag{4.27}$$

which has been proved by (4.21). The same idea can be used to prove that $\models \forall r.(C \sqcup E) \sqsubseteq \forall r.C \sqcup \forall r.E$. □

Structural transformation only preserves satisfiability because it introduces definers. However, the below lemma shows model-inseparability between $\mathcal{O}$ and $\mathcal{O}^{clausal}$.

Lemma 4.2.4. *Let $\mathcal{O}$ be an $\mathcal{ALC}$ ontology, and $\mathcal{O}^{clausal}$ the normal form translation of $\mathcal{O}$ as described above. Then, $\mathcal{O}$ and $\mathcal{O}^{clausal}$ are model-inseparable with respect to $sig(\mathcal{O})$.*

Proof. As discussed above, the NNF transformation, prenexing, and CNF transformation preserve logical equivalence. By Lemma 3.5.8 they also preserve model-inseparability. It remains to show that model-inseparability with respect to $sig(\mathcal{O})$ is preserved by structural transformation.

Let $\mathcal{O}_1$ be an ontology in NNF, α an axiom in $\mathcal{O}_1$, and $C = Q\,r.E$ be a concept in α at position i, where $Q \in \{\exists, \forall\}$, $B \in sig(E)$, and B is a forgetting symbol. After structural transformation we move to a new ontology $\mathcal{O}_2 = \mathcal{O}_1' \cup \{\top \sqsubseteq \neg C \sqcup D \sqcup E\}$ such that $\mathcal{O}_1'$ is equal to $\mathcal{O}_1$ in everything but replaces C with $Q\,r.D$, where D is a fresh definer.

First, we show that every model $\mathcal{I}$ of $\mathcal{O}_2$ is a model of $\mathcal{O}_1$. We need to show that for every domain element $d \in (Q\,r.D)^{\mathcal{I}}$ we have that $d \in (Q\,r.E)^{\mathcal{I}}$. Suppose $Q = \forall$, and there is no $e \in \Delta^{\mathcal{I}}$ such that $(d,e) \in r^{\mathcal{I}}$, then $d \in (\forall r.E)^{\mathcal{I}}$ is also true. Suppose there is $e \in \Delta^{\mathcal{I}}$ where $(d,e) \in r^{\mathcal{I}}$. Since $\mathcal{I} \models \top \sqsubseteq \neg D \sqcup E$, we have that $e \in E^{\mathcal{I}}$. Altogether, we have $d \in (\forall r.E)^{\mathcal{I}}$. The same argument holds when $Q = \exists$.

Second, we show that every model $\mathcal{I}$ of $\mathcal{O}_1$ can be extended to a model $\mathcal{J}$ of $\mathcal{O}_2$, such that $\mathcal{I}$ and $\mathcal{J}$ $sig(\mathcal{O}_1)$-coincide. Define the model $\mathcal{J}$ such that $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}}$, and $\cdot^{\mathcal{J}} = \cdot^{\mathcal{I}} \cup \{D^{\mathcal{J}}\}$, where $D^{\mathcal{J}}$ is defined as follows.

$$D^{\mathcal{J}} = \{y \in E^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\} \tag{4.28}$$

It directly follows from (4.28) that:

$$\mathcal{J} \models \top \sqsubseteq \neg D \sqcup E \tag{4.29}$$

$$\mathcal{J} \models \exists r.E \sqcup \exists r.D \tag{4.30}$$

Recall that $\mathcal{O}_1$ is in NNF, and $B \in sig(E)$ where B is a forgetting symbol. Therefore, the concept $C = Q\,r.E$ must be occurring positively in $\mathcal{O}_1$. Otherwise, there would be a negation indirectly applied to the forgetting symbol B that occurs in E which violates

our normal form. If additionally $Q = 9$, then $\mathcal{J} \models \mathcal{O}_2$ follows from (4.29), (4.30), and the fact that C is positive in $\mathcal{O}_1$.

To show the same result when $Q = 8$, we need only to show that $\mathcal{J} \models \exists r.E \sqsubseteq \forall r.D$. Consider an arbitrary domain element $d \in (\exists r.E)^{\mathcal{I}}$. If there is no $e \in \Delta^{\mathcal{I}}$ such that $(d,e) \in r^{\mathcal{I}}$, then $d \in (\forall r.E)^{\mathcal{I}}$ is also true. If e exists, and $e \in E^{\mathcal{I}}$, then from (4.28) we have $e \in D^{\mathcal{J}}$. Since d and e are arbitrary, we have $\mathcal{J} \models \exists r.E \sqsubseteq \forall r.D$.

From the above arguments we have $\mathcal{J} \models \mathcal{O}_2$. Also, $\mathcal{I}$ and $\mathcal{J}$ $sig(\mathcal{O}_1)$-coincide by construction. Therefore, $\mathcal{O}_1 \underset{sig(\mathcal{O}_1)}{\equiv_M} \mathcal{O}_2$.

Given an ontology $\mathcal{O}$ in NNF and forgetting signature $\mathbf{F}$, we apply the structural transformation exhaustively until no forgetting symbol is present under role restriction in $\mathcal{O}$. This gives a finite series of ontologies $\mathcal{O}_i$, where $0 \leq i \leq n$, $\mathcal{O}_0 = \mathcal{O}$, and $\mathcal{O}_i$ is generated by applying a single structural transformation step on $\mathcal{O}_{i-1}$. This series is finite and is bounded by the number of role restrictions in $\mathcal{O}$. By the above argument we have:

$$\mathcal{O}_0 \underset{sig(\mathcal{O}_0)}{\equiv_M} \cdots \underset{sig(\mathcal{O}_{n-1})}{\equiv_M} \mathcal{O}_n \tag{4.31}$$

because structural transformation only introduces new symbols, we have $sig(\mathcal{O}_i) \subseteq sig(\mathcal{O}_{i+1})$. Then, (4.31) and Lemma 3.5.12 imply that:

$$\mathcal{O}_0 \underset{sig(\mathcal{O}_0)}{\equiv_M} \cdots \underset{sig(\mathcal{O}_0)}{\equiv_M} \mathcal{O}_n \tag{4.32}$$

Furthermore, from Lemma 3.5.7 we have:

$$\mathcal{O}_0 \underset{sig(\mathcal{O}_0)}{\equiv_M} \mathcal{O}_n \tag{4.33}$$

Finally, $\mathcal{O}_n$ is converted to CNF to obtain $\mathcal{O}^{clausal}$. Since CNF transformation preserves logical equivalence, and $\mathcal{O}_0 \equiv \mathcal{O}$, we have $\mathcal{O} \underset{sig(\mathcal{O})}{\equiv_M} \mathcal{O}^{clausal}$. □

We end this section with a summary of properties of $\mathcal{O}^{clausal}$ that will be useful to our framework. All four properties below are direct consequences of our NNF, structural, and CNF transformation as discussed before.

1. The concepts B and $\neg B$ occur only as top-level disjuncts in a clause.

2. A definer symbol D occurs positively only under role restrictions.

3. Concepts of the forms $\exists r.D$ and $\forall r.D$, where D is a definer, occur only positively in $\mathcal{O}^{clausal}$ as top-level disjuncts in clauses.

4. A definer symbol D occurs negatively only as a top-level disjunct in a clause.

In the remainder of this Chapter and in next Chapters, if an ontology consists only of clauses of the form $\top \sqsubseteq C$, then we will drop the leading '$\top \sqsubseteq$' and write the ontology as a set of concepts C for better readability. For instance, an ontology consisting only of the clauses (4.18) and (4.19) will be written as the set $\{A \sqcup E, \neg C \sqcup E\}$.

4.3 Stage One: Resolution

In this Section we specify the first stage of our forgetting framework. The inputs to this stage are an $\mathcal{ALC}$ ontology $\mathcal{O}$, and a set of concept names $\mathbf{F}$ to be forgotten from $\mathcal{O}$. The output is an ontology $\mathcal{O}^{int}$ with the following properties satisfied:

1. The symbols from $\mathbf{F}$ do not occur in $\mathcal{O}^{int}$, and
2. $\mathcal{O}$ and $\mathcal{O}^{int}$ are model inseparable with respect to the non-forgotten symbols $sig(\mathcal{O}) \cap \mathbf{F}$.

A first step in this stage is to write $\mathcal{O}$ in the normal form specified in Section 4.2. Let $\mathcal{O}^{clausal}$ be the ontology in normal form. We generate $\mathcal{O}^{int}$ in two passes. The first pass iterates over the forgetting symbols and substitutes $\top$ for the forgetting symbols that occur only positively in $\mathcal{O}^{clausal}$, and $\bot$ for the forgetting symbols that occur only negatively in $\mathcal{O}^{clausal}$. The second pass iterates over the forgetting symbols which occur both positively and negatively in $\mathcal{O}^{clausal}$. Each iteration resolves, using binary resolution, pairs of clauses that contain the iterated forgetting symbol in opposite polarities. When all resolvents have been captured, the clauses, that use the forgetting symbol, are removed.

During the forgetting process, we eagerly perform simplifications that remove a clause if it contains a forgetting symbol positively and negatively at the same time, also remove clauses of the form $C \sqcup \top$ which contain $\top$ as a top-level disjunct. We call these clauses *tautology clauses*, and the action of removing them *Tautology Deletion*.

Algorithm 1 describes the method of computing $\mathcal{O}^{int}$ in more detail. The ontology $\mathcal{O}^{int}$ is initialized as the subset of $\mathcal{O}^{clausal}$ consisting only of clauses, and an ontology $\mathcal{O}^{axiom}$ is initialized with the axioms that remain. Recall that only the clauses of $\mathcal{O}^{int}$ can contain forgetting symbols. Line 1 performs tautology deletion to remove tautology clauses that may occur in $\mathcal{O}^{int}$ as a result of the normal form transformation. A description of the TautologyDeletion process is given in Algorithm 2. The lines from 2

Algorithm 1: Algorithm for computing $\mathcal{O}^{int}$

Input: Ontology $\mathcal{O}^{clausal}$ in normal form, forgetting signature $\mathbf{F}$
Output: The intermediate ontology $\mathcal{O}^{int}$
Initialize: $\mathcal{O}^{int} :=$ the subset of clauses in $\mathcal{O}^{clausal}$. $\mathcal{O}^{axiom} := \mathcal{O}^{clausal} \setminus \mathcal{O}^{int}$

```
1  O^int := TautologyDeletion(O^int, F)
   // Positive and negative purification
2  while B ∈ F occurs with single polarity in O^int: do
3      if B occurs only positively in O^int then
4          O^int := O^int[B=⊤] // Replace every occurrence of B in O^int with ⊤
5      else if B occurs only negatively in O^int then
6          O^int := O^int[B=⊥] // Replace every occurrence of B in O^int with ⊥
7  for B ∈ F: do // Binary resolution
8      for C1, C2 ∈ O^int do
9          if B occurs with opposite polarities in C1 and C2 then
10             R := Res(C1, C2)
11             O^int := O^int ∪ TautologyDeletion({R}, F)
12     for C ∈ O^int do
13         if B ∈ sig(C) then
14             O^int := O^int \ {C}
15 O^int := O^int ∪ O^axiom
```

Return: $\mathcal{O}^{int}$

to 6 perform positive and negative purification as described above. The lines from 8 to 11 perform binary resolution on pairs of clauses containing a forgetting symbol in opposite polarity. Line 10 refers to the binary resolution rule *Res* in Figure 4.3 which we will describe shortly. Line 11 performs tautology deletion on the resolvent of C_1 and C_2, so the resolvent is used only if it is not a tautology clause. The lines from 12 to 14 remove the clauses that use the forgetting symbol. Note that by line 12, resolution will have exhaustively been applied on the clauses of $\mathcal{O}^{int}$.

Figure 4.3 specifies the rules used to compute $\mathcal{O}^{int}$. The *Resolution* rule takes two clauses as input. One of the two inputs uses a forgetting symbols B positively, and the other clauses uses B negatively. The output of the rule is the resolvent of the input clauses with respect to B. The *Tautology Deletion 1*, *Tautology Deletion 2*, *Positive Purification*, and *Negative Purification* rules are as described before. The following example clarifies the process of computing $\mathcal{O}^{int}$.

Algorithm 2: Algorithm of the TautologyDeletion process

Input: A set of clauses $\mathcal{O}$ in normal form, forgetting signature $\mathbf{F}$
Output: A subset of $\mathcal{O}$ where tautology clauses are removed
Initialize: $\mathcal{T} := \emptyset$

```
1 for C 2 O do
2     if > occurs in C as top-level disjunct then
3         T := T [ fCg
4 for B 2 F : do
5     for C 2 OnT do
6         if B occurs positively and negatively in C then
7             T := T [ fCg
```

Return: $\mathcal{O}n\mathcal{T}$

Example 4.3.1. *Let $\mathcal{O}$ be the following ontology.*

$$A_1 \sqsubseteq \forall r.(B \sqcup C)$$
$$A_2 \sqsubseteq \forall r.(E \sqcap (\neg B \sqcup \neg C))$$
$$\neg E \sqsubseteq F$$

where $\mathbf{F} = \{B, C, E\}$ is a forgetting signature. We start by generating $\mathcal{O}^{clausal}$ as:

$$\neg A_1 \sqcup \forall r.D_1 \tag{4.34}$$
$$\neg A_2 \sqcup \forall r.D_2 \tag{4.35}$$
$$E \sqcup F \tag{4.36}$$
$$\neg D_1 \sqcup B \sqcup C \tag{4.37}$$
$$\neg D_2 \sqcup E \tag{4.38}$$
$$\neg D_2 \sqcup \neg B \sqcup \neg C \tag{4.39}$$

where D_1, and D_2 are fresh definers. Here, all axioms of $\mathcal{O}^{clausal}$ are clauses. Hence, $\mathcal{O}^{int}$ is initialized with all the axioms of $\mathcal{O}^{clausal}$. The concept E is eliminated first because it occurs only positively in $\mathcal{O}^{int}$. E is eliminated by substituting it with $\top$ in (4.36) and (4.38). This transforms (4.36) and (4.38) to the clauses $\{\top \sqcup F; \neg D_2 \sqcup \top\}$ which are removed in tautology deletion. $\mathcal{O}^{int}$ is now the following ontology.

$$\neg A_1 \sqcup \forall r.D_1 \tag{4.40}$$

Resolution (Res)

$$\frac{C_1 \sqsubseteq B \quad C_2 \sqsubseteq \neg B}{C_1 \sqcap C_2 \sqsubseteq \bot}$$

where $B \in \mathbf{F}$ is a forgetting symbol and C_1, C_2 are concepts.

Tautology Deletion 1

$$\frac{\mathcal{O} \cup \{C \sqsubseteq B \sqcup \neg B\}}{\mathcal{O}}$$

where $B \in \mathbf{F}$ is a forgetting symbol, and C is a concepts.

Tautology Deletion 2

$$\frac{\mathcal{O} \cup \{C \sqsubseteq \top\}}{\mathcal{O}}$$

where C is a concept.

Positive Purification

$$\frac{\mathcal{O}}{\mathcal{O}[B \Rightarrow \bot]}$$

where $B \in \mathbf{F}$ is a forgetting symbol, and B occurs only positively in $\mathcal{O}$.

Negative Purification

$$\frac{\mathcal{O}}{\mathcal{O}[B \Rightarrow ?]}$$

where $B \in \mathbf{F}$ is a forgetting symbol, and B occurs only negatively in $\mathcal{O}$.

Figure 4.3: Calculus rules used to compute $\mathcal{O}^{int}$

$$\neg A_2 \sqsubseteq \forall r.\neg D_2 \tag{4.41}$$

$$\neg D_1 \sqsubseteq B \sqcup C \tag{4.42}$$

$$\neg D_2 \sqsubseteq \neg B \sqcup C \tag{4.43}$$

In the second iteration, we eliminate B or C. The order is not important. Suppose we eliminate B first. The clauses (4.42) and (4.43) use B in opposite polarities. We eliminate B from them using the Resolution *rule in Figure 4.3. The resolvent is the clause* $\neg D_1 \sqcup \neg D_2 \sqcup C \sqcup \neg C$*, which is a tautology clause because it contains C positively and negatively. Therefore, the generated resolvent is removed. The clauses (4.42) and (4.43) are removed as well, because no further resolution inferences can be*

performed. Thereby, $\mathcal{O}^{int}$ is obtained as the following ontology in which B;C, and E has been eliminated.

$$: A_1 \sqcup \forall r.D_1 \tag{4.44}$$

$$: A_2 \sqcup \forall r.D_2 \tag{4.45}$$

Observe that we prioritize forgetting the symbols that occur with only one polarity in $\mathcal{O}^{int}$ over the symbols that occur with two polarities. This is because purification is simpler and runs in linear time, whereas resolution runs in polynomial time. Furthermore, purification does not introduce new clauses. The following example shows an interesting situation where prioritizing purification prevents generation of new clauses that should not be present in $\mathcal{O}^{int}$.

Example 4.3.2. *Let $\mathcal{O}$ be the following ontology.*

$$A \sqsubseteq \forall r.B$$
$$G \sqsubseteq \exists r.(\neg B \sqcup C)$$

where $\mathbf{F} = \{B;C\}$ is a forgetting signature. In normal form, $\mathcal{O}^{clausal}$ is the following ontology.

$$: A \sqcup \forall r.D_1 \tag{4.46}$$

$$: G \sqcup \exists r.D_2 \tag{4.47}$$

$$: D_1 \sqcup B \tag{4.48}$$

$$: D_2 \sqcup \neg B \sqcup C \tag{4.49}$$

The forgetting symbol B occurs positively and negatively in (4.48) and (4.49) respectively, whereas the forgetting symbol C occurs only positively in (4.49).

On the one hand, forgetting the symbol C first will substitute $\top$ for C in (4.49), and generate the tautology clause $: D_2 \sqcup \neg B \sqcup \top$, which will be removed by tautology deletion. In the second iteration, B will occur only positively in (4.48), and will be purified in the same way C was purified.

On the other hand, forgetting the symbol B first, will require resolving (4.48) and (4.49). The resolvent will be the clause $: D_1 \sqcup \neg D_2 \sqcup C$, which will be removed as a result of purifying C in the second iteration.

So, when purification was not prioritized, the generated resolvent $:D_1 \sqcup :D_2 \sqcup C$ *was eliminated by subsequent forgetting steps. This is a redundancy that is avoided when purification is prioritized.*

The termination of the generation process of $\mathcal{O}^{int}$ can be seen from the process described above. Purification is terminating, because it eliminates a forgetting symbol in $\mathcal{O}^{int}$ in each iteration, and does not increase the number of clauses of $\mathcal{O}^{int}$. Tautology deletion is terminating because it only removes clauses from $\mathcal{O}^{int}$. Resolution expands $\mathcal{O}^{int}$ with new resolvents. However, this expansion is temporary, because the premises of the resolution inferences are removed when $\mathcal{O}^{int}$ has been saturated with respect to a forgetting symbol, i.e., all possible resolution inferences has been performed on the forgetting symbol. Furthermore, the number of forgetting symbols in $\mathcal{O}^{int}$ in an iteration is strictly less than their number in previous iterations, which makes the loop at line 8 of Algorithm 1 terminating.

The next lemma proves model-inseparability between the input ontology $\mathcal{O}$ and $\mathcal{O}^{int}$.

Lemma 4.3.3. *Let* $\mathcal{O}$ *be an* $\mathcal{ALC}$ *ontology, and* $\mathbf{F}$ *a set of concept names to be forgotten. Let* $\mathcal{O}^{int}$ *be the intermediate ontology computed as explained earlier. Then,* $\mathcal{O}$ *and* $\mathcal{O}^{int}$ *are model-inseparable with respect to* $sig(\mathcal{O}) \cap \mathbf{F}$.

Proof. Recall that $\mathcal{O} \equiv^{M}_{sig(\mathcal{O})} \mathcal{O}^{clausal}$ from Lemma 4.2.4. Since by Lemma 3.5.7, the relation $\equiv^{M}$ is transitive, we need only to prove that $\mathcal{O}^{clausal} \equiv^{M}_{sig(\mathcal{O}) \cap \mathbf{F}} \mathcal{O}^{int}$.

$\mathcal{O}^{int}$ is computed by a process that performs the following four actions.

1. Purification
2. Tautology Deletion
3. Resolution
4. Removing clauses that use a forgetting symbol following exhaustive resolution on the clauses that contain the forgetting symbol.

Purification substitutes $\top$ for a forgetting symbol if it occurs only positively in $\mathcal{O}^{int}$, and $\bot$ if it occurs only negatively. Because the concepts B and $:B$ can occur only as top-level disjuncts in clauses when $B \sqsubseteq \mathbf{F}$ is a forgetting symbol, purifying a concept $B \sqsubseteq \mathbf{F}$ results in $\top$ being introduced as a top-level disjunct in the clauses which triggers the *Tautology Deletion 2* rule in Figure 4.3. We consider purification as one unit applying

the two rules to remove clauses from $\mathcal{O}^{int}$ if they contain a forgetting symbol B, where B occurs with only one polarity in $\mathcal{O}^{int}$.

Suppose $\mathcal{O}_1^{int} = \mathcal{O}_0^{int} \cup \{C_1 \sqsubseteq \mathsf{b}, \ldots, C_n \sqsubseteq \mathsf{b}\}$ is the intermediate ontology before purification applied, where $\mathsf{b} \in \{B, \neg B\}$, and $C_i \sqsubseteq \mathsf{b}$ are the clauses being purified and removed, with $1 \leq i \leq n$. When purification is complete, we have $\mathcal{O}_2^{int} = \mathcal{O}_0^{int}$ as the intermediate ontology. Let $\mathcal{F}_1 = sig(\mathcal{O}_1^{int})$, $\mathcal{F}_2 = sig(\mathcal{O}_2^{int})$, and $\mathcal{F}_3 = \mathcal{F}_1 \cap (\mathcal{F}_2 \cup \{B\})$. The set $\mathcal{F}_3$ contains the concepts and the roles, excluding B, that are removed as a consequence of removing the clauses $C_i \sqsubseteq \mathsf{b}$. If $\mathcal{F}_3 \neq \emptyset$, we add the axioms $A \sqsubseteq \top$, and $\top \sqsubseteq \forall r.\top$ to $\mathcal{O}_1^{int}$ and $\mathcal{O}_2^{int}$, for every concept $A \in \mathcal{F}_3$ and role $r \in \mathcal{F}_3$. Adding these axioms does not allow inferring new information from $\mathcal{O}_1^{int}$ or $\mathcal{O}_2^{int}$. Moreover, the models of $\mathcal{O}_1^{int}$ are not impacted by adding these axioms. The reason for adding these axioms is to ensure that every model of $\mathcal{O}_2^{int}$ has interpretations of the symbols in $\mathcal{F}_3$.

We show that $\mathcal{O}_1^{int} \equiv_{sig(\mathcal{O}) \setminus \{B\}} \mathcal{O}_2^{int}$. First, we show that every model of $\mathcal{O}_1^{int}$ is a model of $\mathcal{O}_2^{int}$. Suppose $\mathcal{I}$ is a model of $\mathcal{O}_1^{int}$. Because $\mathcal{O}_2^{int} \subseteq \mathcal{O}_1^{int}$ we have that $\mathcal{I} \models \mathcal{O}_2^{int}$. Second, we show that every model of $\mathcal{O}_2^{int}$ can be extended to a model of $\mathcal{O}_1^{int}$ where the interpretations of the symbols from $sig(\mathcal{O}) \setminus \{B\}$ are preserved. Suppose $\mathcal{I}$ is a model of $\mathcal{O}_2^{int}$. We construct $\mathcal{J}$ as the model that extends $\mathcal{I}$ with an interpretation of B. If B occurs positively in $\mathcal{O}_1^{int}$, then $B^{\mathcal{J}} = \Delta^{\mathcal{J}}$. If B occurs negatively in $\mathcal{O}_2^{int}$, then $B^{\mathcal{J}} = \emptyset$. Now the clauses $C_i \sqsubseteq \mathsf{b}$ are true in $\mathcal{J}$, because $(C_i \sqsubseteq \mathsf{b})^{\mathcal{J}} = \top$. Additionally, $\mathcal{J} \models \mathcal{O}_0^{int}$ by construction. So, $\mathcal{J} \models \mathcal{O}_1^{int}$. Furthermore, $\mathcal{I}$ and $\mathcal{J}$ coincide on the interpretations of the symbols from $sig(\mathcal{O}_1^{int}) \setminus \{B\}$, so $\mathcal{O}_1^{int} \equiv_{sig(\mathcal{O}_1^{int}) \setminus \{B\}} \mathcal{O}_2^{int}$

The above argument can be generalized to purifying a set $\mathcal{F}'$ of n concept names that occur with a single polarity. We view purifying a set $\mathcal{F}'$ as an iterative process where each iteration purifies a single forgetting symbol B, and triggers the *Tautology Deletion 2* rule. The input to this process is $\mathcal{O}^{clausal}$, and the output is $\mathcal{O}_n^{int}$ where all symbols from $\mathcal{F}'$ are purified. The output of an iteration i, is an ontology $\mathcal{O}_i^{int}$ which purifies a symbol $B_i \in \mathcal{F}$. We have:

$$\mathcal{O}^{clausal} \equiv_{sig(\mathcal{O}^{clausal}) \setminus \{B_1\}} \mathcal{O}_1^{int} \ldots \equiv_{sig(\mathcal{O}_{i-1}^{int}) \setminus \{B_i\}} \mathcal{O}_i^{int} \ldots \equiv_{sig(\mathcal{O}_{n-1}^{int}) \setminus \{B_n\}} \mathcal{O}_n^{int} \tag{4.50}$$

By Lemmas 3.5.7 and 3.5.12, and the fact that $\mathcal{O}_0^{int} = \mathcal{O}^{clausal}$, we get:

$$\mathcal{O}^{clausal} \equiv_{sig(\mathcal{O}^{clausal}) \setminus \mathcal{F}'} \mathcal{O}_n^{int} \tag{4.51}$$

Therefore, purification preserves model-inseparability with respect to the non-purified

symbols. The same proof above can be used to show that the application of the *Tautology Deletion 1* rule preserves model-inseparability.

Resolution only makes implicit entailments explicit. Extending an ontology with sound consequences that follow from the ontology's axioms preserves logical equivalence and implies preservation of model-inseparability (see Lemma 3.5.8). The concern however is when the clauses used as premises to resolution are removed after exhaustive resolution has been complete. Like before, we consider resolution as one unit that exhaustively resolves a symbol B, and removes the used premises when all resolvents has been produced.

Let $\mathcal{O}_1^{int}$ be the state of $\mathcal{O}^{int}$ before resolution is performed. Consider all clauses $C_i \vee B$ in $\mathcal{O}^{int}$ that use B positively with $1 \leq i \leq n$. We replace these clauses with the single clause $C \vee B$ where $C = \bigvee_{i=1}^{n} C_i$. Observe that the set $\{C_i \vee B\}$ is the CNF transformation of $C \vee B$. We showed in Section 4.2 that CNF transformation preserves logical equivalence. In the same way, if there are m clauses of the form $E_i \vee \neg B$ in $\mathcal{O}^{int}$, we replace them with the single clause $E \vee \neg B$ where $E = \bigvee_{j=0}^{m} E_j$. We have:

$$\mathcal{O}_1^{int} = \mathcal{O}_0^{int} \cup \{C \vee B; E \vee \neg B\} \tag{4.52}$$

where $\mathcal{O}_0^{int}$ represents the clauses and axioms that do not use B.

Let $\mathcal{O}_2^{int}$ be the state of $\mathcal{O}^{int}$ after resolution is performed and premises are removed. We have:

$$\mathcal{O}_2^{int} = \mathcal{O}_0^{int} \cup \{C \vee E\} \tag{4.53}$$

where $C \vee E$ is the resolvent of $C \vee B$ and $C \vee \neg B$.

First, we show that every model $\mathcal{I}$ of $\mathcal{O}_1^{int}$ is a model of $\mathcal{O}_2^{int}$. This can be directly seen because $\mathcal{O}_0^{int} \subseteq \mathcal{O}_1^{int}$, and $\{C \vee B; E \vee \neg B\} \models C \vee E$.

Second, we show that every model $\mathcal{I}$ of $\mathcal{O}_2^{int}$ can be extended to a model of $\mathcal{O}_1^{int}$ where the interpretations of the symbols from $sig(\mathcal{O}_1^{int}) \setminus \{B\}$ are preserved. We construct $\mathcal{J}$ as the model that extends $\mathcal{I}$ with the interpretation of B in the following way.

$$B^{\mathcal{J}} = \{d \in \Delta^{\mathcal{J}} \mid d \in C^{\mathcal{J}} \text{ and } d \in E^{\mathcal{J}}\} \tag{4.54}$$

Since $\mathcal{O}_2^{int} \models C \vee E$, every $d \in \Delta^{\mathcal{J}}$ must be in the interpretations of C, E, or both. From 4.54, we have:

1. If $d \in C^{\mathcal{J}}$ and $d \in E^{\mathcal{J}}$, then $d \in B^{\mathcal{J}}$.
2. If $d \in C^{\mathcal{J}}$ and $d \notin E^{\mathcal{J}}$, then $d \notin B^{\mathcal{J}}$.

3. If $d \in C^{\mathcal{J}}$ and $d \in E^{\mathcal{J}}$, then $d \in B^{\mathcal{J}}$.

In all three cases we get that $\mathcal{J} \models C \sqsubseteq B$, and $\mathcal{J} \models E \sqsubseteq \bot$. Hence, $\mathcal{J} \models \mathcal{O}_1^{int}$. Also, $\mathcal{I}$ and $\mathcal{J}$ coincide on the interpretations of $sig(\mathcal{O}_1^{int}) \cap \{B\}$. Therefore $\mathcal{O}_1^{int} \equiv_{sig(\mathcal{O}_1^{int}) \cap \{B\}} \mathcal{O}_2^{int}$. This result can be generalized as before to the case when a set Γ' of concept names are resolved.

Altogether, we conclude that $\mathcal{O}^{clausal}$ and $\mathcal{O}^{int}$ are model-inseparable with respect to $\Gamma \cup (\mathcal{O}^{clausal}) \cap \Gamma$. From Lemmas 3.5.12, Lemma 3.5.7, and 4.2.4 we get that $\mathcal{O} \equiv_{sig(\mathcal{O}) \cap \Gamma} \mathcal{O}^{int}$. □

4.4 Stage Two: Deductive Reduction

The second stage of our forgetting is Deductive Reduction. The input to this stage is the intermediate ontology $\mathcal{O}^{int}$ computed in the previous section. The outputs are two sets, $\mathcal{O}^d$ and D^d, of axioms. The first set $\mathcal{O}^d$ is called *the deductively reduced* ontology. It coincides with the deductive forgetting views of the input ontology $\mathcal{O}$ with respect to the forgetting signature Γ. That is, the ontologies $\mathcal{O}$ and $\mathcal{O}^d$ are deductively inseparable with respect to the non-forgotten symbols $sig(\mathcal{O}) \cap \Gamma$. The axioms of the second set D^d are consequences of $\mathcal{O}^{int}$ that are not used or entailed by $\mathcal{O}^d$. Furthermore, by taking the union of $\mathcal{O}^d$ and D^d, we can reconstruct $\mathcal{O}^{int}$. In this sense, the set D^d indicates the content difference between $\mathcal{O}^{int}$ and $\mathcal{O}^d$. We will use D^d in the next sections to obtain fine-grained forgetting views with more content than the deductive forgetting views. Important proofs and techniques will be developed in this section to prove deductive inseparability between $\mathcal{O}$ and $\mathcal{O}^d$. The concepts developed in this section will be useful in the next chapter.

The ontology $\mathcal{O}^d$ is mainly obtained from $\mathcal{O}^{int}$ by removing the D^d clauses. But some entailments of $\mathcal{O}^{int}$ over $sig(\mathcal{O}) \cap \Gamma$ may no longer be computable when the D^d clauses are removed. These entailments should be preserved by $\mathcal{O}^d$. So, we compute $\mathcal{O}^d$ in two passes. In the first pass, we compute and preserve the noted consequences. In the second pass, we remove the clauses of D^d from $\mathcal{O}^{int}$. The ontology $\mathcal{O}^d$ is then the remaining axioms of $\mathcal{O}^{int}$ extended with the entailments preserved in the first pass.

4.4.1 The First Pass

The aim of this pass is to compute some implicit entailments of $\mathcal{O}^{int}$, so to allow preserving them later in $\mathcal{O}^d$. To compute these entailments, we use the *Role Propagation*

Role Propagation

$$\frac{P_0 \sqsubseteq C_0,\ \bigsqcup_{j=1} \{P_j \sqsubseteq C_j\},\ E_0 \sqsubseteq Q\, r.D_0,\ \bigsqcup_{i=1} \{E_i \sqsubseteq \forall r.D_i\}}{(\bigsqcup_{i=0} E_i) \sqsubseteq Q\, r.(\bigsqcup_{j=0} C_j)}$$

where $P_0 = \bigsqcup_{i=0} \neg D_i$, P_j is any concept in P_0, $Q \in \{\exists, \forall\}$, and C_0 and C_j do not contain a definer.

Figure 4.4: Role Propagation rule.

rule in Figure 4.4. The rule relies on the following two properties which allow for extracting the premises of the *Role Propagation* inferences.

1. Concepts of the forms $\exists r.D$ and $\forall r.D$, where D is a definer, occur only positively, and occur only as top-level disjuncts in clauses.

2. A definer symbol D occurs negatively only as a top-level disjunct in a clause.

The two properties originate in our normal form. They remain true in O^{int}, because O^{int} is generated by (1) purification and tautology deletion, which only remove clauses, and do not therefore impact the two properties above, and (2) resolution, which creates new clauses whose top-level disjuncts occurs as top-level disjuncts in the premises.

A *Role Propagation* inference uses four premises. The first premise is a clause $P_0 \sqsubseteq C_0$ that satisfies two conditions.

1. The concept C_0 does not contain negative definers.

2. The concept P_0 is a disjunction of $n+1$ negative definers where $n \geq 1$. That is, P_0 takes the form $\neg D_0 \sqcup \neg D_1 \sqcup \cdots \sqcup \neg D_n$ where $D_0, D_1, \ldots, D_n$ are definers.

We call the clause $P_0 \sqsubseteq C_0$ a *trigger clause*, and the definers occurring in P_0 *trigger definers*, because they influence the other premises of the *Role Propagation* rule.

When $\mathcal{O}^{int}$ has several clauses:

$$P_0 \sqsubseteq C_0^1$$
$$P_0 \sqsubseteq C_0^2$$
$$\vdots$$
$$P_0 \sqsubseteq C_0^l$$

with $l \geq 1$, we *combine* these clauses in a single clause:

$$P_0 \sqsubseteq (\bigsqcap_{i=1}^{l} C_0^i)$$

The combined clause is then used as the trigger clause to the *Role Propagation* inference. This ensures that two *Role Propagation* inferences use different sets of trigger definers. Suppose, for example, an intermediate ontology $\mathcal{O}^{int}$ has the following clauses.

$$\neg D_1 \sqcap \neg D_2 \sqcap \neg D_3 \sqsubseteq A_1 \tag{4.55}$$
$$\neg D_1 \sqcap \neg D_2 \sqcap \neg D_3 \sqsubseteq A_2 \tag{4.56}$$

where D_1, D_2, and D_3 are definers. The two clauses (4.55) and (4.56) are on the form of a trigger clause, with $P_0 = \neg D_1 \sqcap \neg D_2 \sqcap \neg D_3$, and D_1, D_2, and D_3 being the trigger definers. Neither of them can however be used as a trigger clause for a *Role Propagation* inference. Instead, the clause $\neg D_1 \sqcap \neg D_2 \sqcap \neg D_3 \sqsubseteq (A_1 \sqcap A_2)$, obtained by combining (4.55) and (4.56), can be used as a trigger clause for a *Role Propagation* inference. Note that the combine action is the reverse of the $(cnf\ 2)$ rule in Table 4.1. It therefore preserves logical equivalence (see Lemma 4.2.3).

The second premise of the *Role Propagation* rule is a set of m clauses $P_j \sqsubseteq C_j$, with $1 \leq j \leq m$ and $m \geq 0$. Like the first premise, negative definers cannot occur in C_j, and the concepts P_j are disjunctions of negative definers. However, the set of definers occurring in P_j must be a subset of the trigger definers occurring in P_0. For example, suppose an intermediate ontology $\mathcal{O}^{int}$ has the following clauses:

$$\neg D_1 \sqcap \neg D_2 \sqcap \neg D_3 \sqsubseteq A_1 \tag{4.57}$$
$$\neg D_1 \sqcap \neg D_3 \sqsubseteq A_2 \tag{4.58}$$
$$\neg D_2 \sqsubseteq A_3 \tag{4.59}$$

$$: D_2 \sqcup : D_4 \sqcup A_4 \tag{4.60}$$

where $D_1; D_2; D_3; D_4 \in N_d$ are definers, and $A_1; A_2; A_3; A_4 \in N_c$ are concept names. Let (4.57) be the trigger clause of the *Role Propagation* inference. The concept P_0 is:

$$P_0 = : D_1 \sqcup : D_2 \sqcup : D_3$$

which uses the trigger definers $D_1; D_2$, and D_4. The set, consisting of the clauses (4.58) and (4.59), is the second premise of the *Role Propagation* rule, because the negative definers occurring in (4.58) and (4.59) are trigger definers. The clause (4.60) cannot be included in this set, because (4.60) contains the definer D_4 negatively, and D_4 is not a trigger definer.

The idea of the second premise is that $P_j \sqsubseteq P_0$. Therefore, every domain element, that is not in the interpretation of P_0, is not in the interpretations of P_j. Furthermore, the domain element must then be in the interpretations of C_0 and C_j, because $O^{int} \models P_0 \sqcup C_0$ and $O^{int} \models \bigsqcup_{j=1}^{3} \{P_j \sqcup C_j\}$. For instance, in the above example, we had $P_0 = : D_1 \sqcup : D_2 \sqcup : D_3$. The second premise of the *Role Propagation* inference would then consist of two clauses $P_1 \sqcup C_1$ and $P_2 \sqcup C_2$, that correspond to the clauses (4.58) and (4.59) respectively. From this we have the following:

$$P_1 = : D_1 \sqcup : D_3 \qquad C_1 = A_2$$
$$P_2 = : D_2 \qquad C_2 = A_3$$

The following relations can be observed about $P_0; P_1$, and P_2.

$$P_1 \sqsubseteq P_0 \tag{4.61}$$
$$P_2 \sqsubseteq P_0 \tag{4.62}$$

Suppose I is a model of O^{int}, and there is a $d \in \Delta^I$ such that $d \in P_0^I$. From (4.61) and (4.62), we have that $d \in P_1^I$ and $d \in P_2^I$. Then (4.57), (4.58), and (4.59) would imply that $d \in A_1^I$, $d \in A_2^I$, and $d \in A_3^I$.

The clauses of the third and the fourth premises have similar forms. The two premises contain clauses where trigger definers occur positively under role restriction. The third premise allows for a trigger definer D_0 to occur under existential role restrictions. The other trigger definers can only occur under universal role restriction. $E_0; E_1; \ldots$ are concepts that may use definers.

There can be $n + 1$ clauses in the third and the fourth premises. Each clause uses a trigger definer under role restriction. As P_0 contains $n + 1$ trigger definers, each clause in the third and fourth premises corresponds to one of these trigger definers. If O^{int} has many clauses where a trigger definer occurs positively, then we combine them, and use the combined clause in the inference. Consider the following set of clauses that use a definer D in positive polarity.

$$E^1 \sqsubseteq \exists r.D \tag{4.63}$$

$$E^2 \sqsubseteq \exists r.D \tag{4.64}$$

$$\vdots \tag{4.65}$$

$$E^l \sqsubseteq \exists r.D \tag{4.66}$$

The combined clauses is the following clause.

$$(\bigsqcap_{i=1}^{l} E^i) \sqsubseteq \exists r.D \tag{4.67}$$

The third and the fourth premises imply that every domain element must be in the interpretation of $\bigsqcup_{i=0}^{n} E_i$ or $\exists r.(\bigsqcap_{i=0}^{n} D_i)$. The latter concept can be rewritten as $\exists r.\neg P_0$ because $P_0 = \neg D_1 \sqcup \ldots \sqcup \neg D_n$. But, $\exists r.(\bigsqcap_{j=0}^{m} C_j)$ subsumes $\exists r.\neg P_0$, because every domain element in the interpretation of $\neg P_0$ must be in the interpretation of $\bigsqcap_{j=0}^{m} C_j$ as discussed earlier. Therefore, arbitrary domain elements must be in the interpretation of $\bigsqcup_{i=0}^{n} E_i$ or $\exists r.(\bigsqcap_{j=0}^{m} C_j)$. We can explain this on the above example. Suppose we extend O^{int} from the above example with the following clauses.

$$E_1 \sqsubseteq \forall r.D_1 \tag{4.68}$$

$$E_2 \sqsubseteq \forall r.D_2 \tag{4.69}$$

$$E_3 \sqsubseteq \exists r.D_3 \tag{4.70}$$

The trigger clause is $\neg D_1 \sqcup \neg D_2 \sqcup \neg D_3 \sqcup A_1$, and the second premise is the set consisting of (4.58) and (4.59) as before. The third premise must be the clause (4.70) because it uses existential role restriction. The fourth premise is the set consisting of (4.68) and (4.69). Suppose I is a model of O^{int}, and $d \in \Delta^I$ a domain element. From (4.68), (4.69), and (4.70), it follows that if $d \in (E_1^I \cup E_2^I \cup E_3^I)$, then $d \in ((\forall r.D_1)^I \setminus (\forall r.D_2)^I \setminus (\exists r.D_3)^I)$. We now have the following:

1. Because $d \in \exists r.D_3$, there must exist a domain element $e \in \Delta^{\mathcal{I}}$ such that $e \in D_3^{\mathcal{I}}$ and $(d,e) \in r^{\mathcal{I}}$.

2. Because $d \in \forall r.D_2$ and $(d,e) \in r^{\mathcal{I}}$, it must be that $e \in D_2^{\mathcal{I}}$.

3. Because $d \in \forall r.D_1$ and $(d,e) \in r^{\mathcal{I}}$, it must be that $e \in D_1^{\mathcal{I}}$.

Summarizing the above, if $d \notin (E_1^{\mathcal{I}} \cup E_2^{\mathcal{I}} \cup E_3^{\mathcal{I}})$, then there is a domain element e such that $(d,e) \in r^{\mathcal{I}}$ and $e \in D_1^{\mathcal{I}} \setminus D_2^{\mathcal{I}} \setminus D_3^{\mathcal{I}}$. The first and second premises then imply that $e \in (A_1^{\mathcal{I}} \setminus A_2^{\mathcal{I}} \setminus A_3^{\mathcal{I}})$, as discussed before. So, every domain element $d \in \Delta^{\mathcal{I}}$ must be in the interpretation of $\bigcup_{i=1}^{3} E_i$ or $\exists r.(\bigcap_{i=1}^{3} A_i)$.

Lemma 4.4.1. *The conclusion of the* Role Propagation *rule in Figure 2 is entailed by the premises.*

Proof. Let $\mathcal{I}^{int}$ be an arbitrary model of $\mathcal{O}^{int}$ and d be a domain element in $\Delta^{\mathcal{I}^{int}}$. If $d \notin (E_0 \sqcup \ldots \sqcup E_n)^{\mathcal{I}^{int}}$, then it must be the case that $d \in (Q\, r.D_0 \sqcap \forall r.D_1 \sqcap \ldots \sqcap \forall r.D_n)^{\mathcal{I}^{int}}$. This is equivalent to saying:

$$d \in (Q\, r.D_0)^{\mathcal{I}^{int}} \tag{4.71}$$

$$d \in (\forall r.(D_1 \sqcap \ldots \sqcap D_n))^{\mathcal{I}^{int}} \tag{4.72}$$

where (4.72) is obtained by the prenexing rule (*pnx* 2), which preserves logical equivalence (see Lemma 4.2.3).

Suppose $Q = \exists$, then there is $e \in D_0^{\mathcal{I}^{int}}$ such that $(d,e) \in r^{\mathcal{I}^{int}}$. It must also be that $e \in (D_0 \sqcap \ldots \sqcap D_n)^{\mathcal{I}^{int}}$. Observe that $P_0 \sqsubseteq (D_0 \sqcap \ldots \sqcap D_n)$, so $e \notin P_0^{\mathcal{I}^{int}}$.

Since $\mathcal{I}^{int} \models P_0 \sqcup C_0$ we get that $e \in C_0^{\mathcal{I}^{int}}$. Similarly, since $P_j \sqsubseteq P_0$, we have $e \in C_j^{\mathcal{I}^{int}}$. Altogether, $d \in (E_0 \sqcup \ldots \sqcup E_n \sqcup \exists r.(C_0 \sqcap \ldots \sqcap C_m))^{\mathcal{I}^{int}}$ for any domain element d.

Suppose $Q = \forall$, then $d \in \forall r.(D_0 \sqcap \ldots \sqcap D_n))^{\mathcal{I}^{int}}$ which is equivalent to saying that $d \in (\forall r.\neg P_0)^{\mathcal{I}^{int}}$. It follows that $d \in (\forall r.C_0)^{\mathcal{I}^{int}}$. Additionally, it must be the case that $d \in (\forall r.\neg P_j)^{\mathcal{I}^{int}}$ because $\forall r.\neg P_j$ subsumes $\forall r.\neg P_0$. So $d \in (\forall r.C_j)^{\mathcal{I}^{int}}$ Altogether, $d \in (E_0 \sqcup \ldots \sqcup E_n \sqcup \forall r.(C_0 \sqcap \ldots \sqcap C_m))^{\mathcal{I}^{int}}$ for any domain element d. □

The following example shows the importance of combining clauses in the premises of the *Role Propagation* inferences. When the clauses are not combined, we may obtain weaker conclusions compared to those obtained when the clauses are combined.

Example 4.4.2. *Let $\mathcal{O}$ be the following ontology.*

$$A_1 \sqsubseteq \forall r.B \tag{4.73}$$
$$A_2 \sqsubseteq \exists r.(\neg B \sqcap C) \tag{4.74}$$
$$C \sqsubseteq E_1 \tag{4.75}$$
$$C \sqsubseteq E_2 \tag{4.76}$$

Consider the forgetting signature $\mathcal{F} = \{B, C\}$. In the normal form, $\mathcal{O}^{clausal}$ is the following ontology.

$$\neg A_1 \sqcup \forall r.D_1 \tag{4.77}$$
$$\neg A_2 \sqcup \exists r.D_2 \tag{4.78}$$
$$\neg D_1 \sqcup B \tag{4.79}$$
$$\neg D_2 \sqcup \neg B \sqcup C \tag{4.80}$$
$$\neg C \sqcup E_1 \tag{4.81}$$
$$\neg C \sqcup E_2 \tag{4.82}$$

The intermediate ontology $\mathcal{O}^{int}$ is constructed as:

$$\neg A_1 \sqcup \forall r.D_1 \tag{4.83}$$
$$\neg A_2 \sqcup \exists r.D_2 \tag{4.84}$$
$$\neg D_1 \sqcup \neg D_2 \sqcup E_1 \tag{4.85}$$
$$\neg D_1 \sqcup \neg D_2 \sqcup E_2 \tag{4.86}$$

The clauses (4.85) and (4.86) have the form of trigger clauses, and they use the same set $\{D_1, D_2\}$ of trigger definers. We compare the conclusions of the Role Propagation *inferences when (4.85) and (4.86) are combined, and when they are not combined and used in two separate inferences.*

When (4.85) and (4.86) are combined, we get the clause $\neg D_1 \sqcup \neg D_2 \sqcup (E_1 \sqcap E_2)$ as a trigger clause to a Role Propagation *inference. The empty set $\emptyset$ is the second premise of the inference. The third premise is the clause $\neg A_2 \sqcup \exists r.D_2$ because it uses existential role restriction. The forth premise is the set consisting of the clause $\neg A_1 \sqcup \forall r.D_1$. The following clause is the conclusion of the* Role Propagation *inference.*

$$\neg A_1 \sqcup \neg A_2 \sqcup \exists r.(E_1 \sqcap E_2) \tag{4.87}$$

Suppose (4.85) and (4.86) are not combined, and used as trigger clauses of two different Role Propagation *inferences. Using the second, third, and fourth premises of the two* Role Propagation *inferences as above, we obtain the following two conclusions.*

$$:A_1\ t\ :A_2\ t\ \exists r{:}E_1 \tag{4.88}$$

$$:A_1\ t\ :A_2\ t\ \exists r{:}E_2 \tag{4.89}$$

The conclusion (4.87) obtained when (4.85) and (4.86) are combined is stronger than the two clauses (4.88) and (4.89), because if (4.87) is satisfied in any model I, then (4.88) and (4.89) are satisfied, but not vice versa.

4.4.2 The Second Pass

In the previous section, we made some of the implicit information of O^d explicit using the *Role Propagation* rule. We assume hereafter that all possible *Role Propagation* inferences have been performed.

Definition 4.4.3. *We denote by O^{rp} the extension of the ontology O^{int} with the conclusions of the Role Propagation inferences.*

Due to combining clauses in the premises of the *Role Propagation* inferences, the preserved conclusions might not be in the normal form. So, a first step in this pass is to normalize O^{rp} using the process in Section 4.2.

Having O^{rp} prepared as above, we split it into two ontologies D^d and O^d. We call O^d a *deductively reduced* ontology because it is deductively inseparable from the input ontology O with respect to the non-forgotten symbols $sig(O) \cap F$. As such, O^d satisfies Condition 2 of Definition 3.5.14 of deductive forgetting. O^d consists of the clauses of O^{rp} that are not in D^d. This gives the following equation:

$$O^{rp} = O^d \cup \mathsf{D}^d \tag{4.90}$$

The clauses of D^d can be syntactically characterized.

Definition 4.4.4. *We denote by D^d the clauses of O^{rp} which contains two or more definers with negative polarity. We denote by O^d the subset of O^{rp} disjoint with D^d.*

We use the *Reduction* rule in Figure 4.5 to remove the D^d clauses from O^{rp} and obtain O^d as described by (4.90) and Definition 4.4.4.

Reduction

$$\frac{\mathcal{O} \cup \{ f : D_1 \sqcap \cdots \sqcap \top : D_n \sqsubseteq C \}}{\mathcal{O}}$$

where C is a concept, $D_1, \ldots, D_n$ are definer symbols, and $n \geq 2$.

Figure 4.5: $\mathcal{ALC}$ reduction rules.

Example 4.4.5. *Let $\mathcal{O}$ be the following ontology.*

$$A \sqsubseteq \forall r.B \sqcap \forall s.\top : B \tag{4.91}$$
$$G \sqsubseteq \exists r.(\neg B \sqcap C) \tag{4.92}$$
$$B \sqsubseteq H \tag{4.93}$$

Consider the forgetting symbol $\mathbf{F} = \{B\}$. We have the intermediate ontology $\mathcal{O}^{int}$ as follows.

$$\neg A \sqcap \forall r.D_1 \tag{4.94}$$
$$\neg A \sqcap \forall s.D_2 \tag{4.95}$$
$$\neg G \sqcap \exists r.D_3 \tag{4.96}$$
$$\neg D_1 \sqcap H \tag{4.97}$$
$$\neg A \sqcap \top : \neg G \sqcap \exists r.(C \sqcap H) \tag{4.98}$$
$$\neg D_1 \sqcap \top : D_2 \tag{4.99}$$
$$\neg D_1 \sqcap \top : D_3 \sqcap C \tag{4.100}$$

where D_1, D_2, and D_3 are definers.

The clause (4.98) is the conclusion of a role inference whose trigger clause is (4.100), second premise is the set consisting of (4.97), third premise is (4.96), and fourth premise is the set consisting of (4.94).

The ontology $\mathbf{D}^d$ is the subset consisting of the two clauses (4.99) and (4.100), because each contains two negative definers. The ontology $\mathcal{O}^d$ is the remaining subset consisting of the first five clauses.

Theorem 4.4.6. $\mathcal{O}^{int} \underset{sig(\mathcal{O}) \cap \mathbf{F}}{\equiv_C} \mathcal{O}^d$.

It suffices to prove that $\mathcal{O}^{rp} \underset{sig(\mathcal{O}) \cap \mathbf{F}}{\equiv_C} \mathcal{O}^d$ using the definitions and lemmas in the

rest of this section. The proof is by contradiction. This requires finding an $\mathcal{ALC}$ axiom α where $sig(\alpha) \not\subseteq sig(\mathcal{O}) \cup \mathbf{F}$ such that at least one of the following conditions holds:

1. $\mathcal{O}^{rp} \models \alpha$ and $\mathcal{O}^{d} \not\models \alpha$; or
2. $\mathcal{O}^{rp} \not\models \alpha$ and $\mathcal{O}^{d} \models \alpha$.

However, condition 2 is not satisfiable, because $\mathcal{O}^{d} \subseteq \mathcal{O}^{rp}$. The subsumption implies that every model of $\mathcal{O}^{rp}$ is a model of $\mathcal{O}^{d}$. If $\mathcal{O}^{rp} \not\models \alpha$, then there is a model $\mathcal{I}$ of $\mathcal{O}^{rp}$ where $\mathcal{I} \not\models \alpha$. Because $\mathcal{I}$ is also a model of $\mathcal{O}^{d}$, we must have that $\mathcal{O}^{d} \not\models \alpha$ which contradicts the second part of Condition 2. Definition 4.4.7, and Lemmas 4.4.8 and 4.4.9 below formalize this observation.

Definition 4.4.7. *Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be any two ontologies. By* $\mathrm{mDiff}(\mathcal{O}_1; \mathcal{O}_2)$ *we mean the set of models of $\mathcal{O}_1$ that are not models of $\mathcal{O}_2$.*

Lemma 4.4.8. *$mDiff(\mathcal{O}^{rp}; \mathcal{O}^{d}) = \emptyset$, but in general $mDiff(\mathcal{O}^{d}; \mathcal{O}^{rp}) \neq \emptyset$.*

Proof. (1) $\mathrm{mDiff}(\mathcal{O}^{rp}, \mathcal{O}^{d}) = \emptyset$: Let $\mathcal{I}$ be any model of $\mathcal{O}^{rp}$. Since $\mathcal{O}^{d} \subseteq \mathcal{O}^{rp}$, it must be that $\mathcal{I} \models \mathcal{O}^{d}$.

(2) $\mathrm{mDiff}(\mathcal{O}^{d}, \mathcal{O}^{rp}) \neq \emptyset$: We prove this by giving an example of a model of $\mathcal{O}^{d}$ that is not a model of $\mathcal{O}^{rp}$.

Let $\mathcal{O}$ be the following ontology.

$$\top \sqsubseteq \exists r.B \sqcup \exists r.\neg B$$

Consider the forgetting signature $\mathbf{F} = \{B\}$. Then $\mathcal{O}^{rp}$ is:

$$\exists r.D_1 \tag{4.101}$$
$$\exists r.D_2 \tag{4.102}$$
$$\neg D_1 \sqcup \neg D_2 \tag{4.103}$$

where D_1 and D_2 are definers. $\mathcal{O}^{d}$ is the subset consisting of the first two clauses (4.101) and (4.102).

Let $\mathcal{I}$ be the interpretation with $\Delta^{\mathcal{I}} = \{a, b\}$, and:

$$D_1^{\mathcal{I}} = \{b\}$$
$$D_2^{\mathcal{I}} = \{b\}$$
$$r^{\mathcal{I}} = \{(a, b)\}$$

$\mathcal{I}$ is a model of $\mathcal{O}^d$ but is not a model of $\mathcal{O}^{rp}$. □

Lemma 4.4.9. *$\mathcal{O}^{rp} \not\equiv^{\mathcal{C}}_{sig(\mathcal{O})nF} \mathcal{O}^d$ if and only if there exists a concept inclusion α over $sig(\mathcal{O})nF$ such that $\mathcal{O}^{rp} \models \alpha$ and $\mathcal{O}^d \not\models \alpha$.*

Proof. Right to left: suppose there is an $\mathcal{ALC}$ concept inclusion α over $sig(\mathcal{O})nF$, where $\mathcal{O}^{rp} \models \alpha$, and $\mathcal{O}^d \not\models \alpha$. From Definition 3.5.6 it follows that $\mathcal{O}^{rp} \not\equiv^{\mathcal{C}}_{sig(\mathcal{O})nF} \mathcal{O}^d$.

Left to right: Suppose $\mathcal{O}^{rp} \not\equiv^{\mathcal{C}}_{sig(\mathcal{O})nF} \mathcal{O}^d$. By definition there must be a concept inclusion $\alpha = C \sqsubseteq D$ over $sig(\mathcal{O})nF$ such that:

1. $\mathcal{O}^{rp} \models \alpha$ and $\mathcal{O}^d \not\models \alpha$, or

2. $\mathcal{O}^{rp} \not\models \alpha$ and $\mathcal{O}^d \models \alpha$.

Consider the second case, there must be a model $\mathcal{I}$ of $\mathcal{O}^{rp}$ that satisfies the $\mathcal{ALC}$ concept $C \sqcap \neg D$.

By Lemma 4.4.8 $\text{mDiff}(\mathcal{O}^{rp}; \mathcal{O}^d) = \emptyset$, hence $\mathcal{I}$ is a model of $\mathcal{O}^d$ and $C \sqcap \neg D$ is satisfiable in $\mathcal{O}^d$ which contradicts the assumption that $\mathcal{O}^d \models \alpha$. □

We examine in more details the models of $\mathcal{O}^d$ that are not models of $\mathcal{O}^{rp}$. That is, the models in the set $\text{mDiff}(\mathcal{O}^d; \mathcal{O}^{rp})$. Our aim is to show that every model in $\text{mDiff}(\mathcal{O}^d; \mathcal{O}^{rp})$ is *bisimilar* to a model of $\mathcal{O}^{rp}$. Bisimulation is an equivalence relation by which if two models are bisimilar and a concept C is true in one of the models, then it must be true in the other model [Sti98, Pir13]. We can use bisimulation as follows. If $\mathcal{O}^{rp} \not\equiv^{\mathcal{C}}_{sig(\mathcal{O})nF} \mathcal{O}^d$, then, from Lemma 4.4.9, there is a witness axiom α where $\mathcal{O}^{rp} \models \alpha$, and $\mathcal{O}^d \not\models \alpha$. Suppose $\alpha = C \sqsubseteq D$, then there must be a model $\mathcal{I}$ of $\mathcal{O}^d$ where $C \sqcap \neg D$ is true in $\mathcal{I}$. Note that $\mathcal{I}$ is in $\text{mDiff}(\mathcal{O}^d; \mathcal{O}^{rp})$ because $\mathcal{O}^{rp} \models C \sqsubseteq D$. If we can show that $\mathcal{I}$ is bisimilar to a model of $\mathcal{O}^{rp}$, then there is a model of $\mathcal{O}^{rp}$ where $C \sqcap \neg D$ is true, and we would obtain the contradiction. Bisimulation relies on the notion of *pointed interpretations* explained in the following definition.

Definition 4.4.10. *A* pointed interpretation *$(\mathcal{I}, d)$ is an interpretation $\mathcal{I}$ generated by $d \in \Delta^{\mathcal{I}}$. $(\mathcal{I}, d)$ is a labelled directed graph such that:*

1. *The nodes of the graph nodes are domain elements from $\Delta^{\mathcal{I}}$*

2. *d is the root node of the graph.*

3. *A node e in the graph is labelled with a concept C if and only if $e \in C^{\mathcal{I}}$.*

4. *For every two nodes $e_1; e_2 \in \Delta^{\mathcal{I}}$, there is a r-transition, or an edge labelled with r, from e_1 to e_2 if and only if $(e_1; e_2) \in r^{\mathcal{I}}$ where $r \in N_r$.*

Definition 4.4.11. *Let $(\mathcal{I}; d_1)$ and $(\mathcal{J}; d_2)$ be two pointed interpretations, and Σ a signature. $(\mathcal{I}; d_1); (\mathcal{J}; d_2)$ are* Σ-bisimilar, *in symbols $(\mathcal{I}; d_1) \leftrightarrow_\Sigma (\mathcal{J}; d_2)$, iff there is a relation $R \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ where $(d_1; d_2) \in R$ and for every $(d; d') \in R$ the following hold:*

1. *$d \in A^{\mathcal{I}}$ iff $d' \in A^{\mathcal{J}}$ for all concept names $A \in \Sigma$.*

2. *if $(d; e) \in r^{\mathcal{I}}$ then there is $e' \in \Delta^{\mathcal{J}}$ such that $(d'; e') \in r^{\mathcal{J}}$ for every role name $r \in \Sigma$ and $(e; e') \in R$.*

3. *if $(d'; e') \in r^{\mathcal{J}}$ then there is $e \in \Delta^{\mathcal{I}}$ such that $(d; e) \in r^{\mathcal{I}}$ for every role name $r \in \Sigma$ and $(e; e') \in R$.*

Example 4.4.12. *Let $\mathcal{I}$ and $\mathcal{J}$ be two models over $\Delta^{\mathcal{I}} = \{a; b_1; b_2; c\}$ and $\Delta^{\mathcal{J}} = \{x; y; z\}$ respectively. Figure 4.6 shows the two models, where $\mathcal{I}$ is the drawing on the left of the figure, and $\mathcal{J}$ is the drawing on the right.*

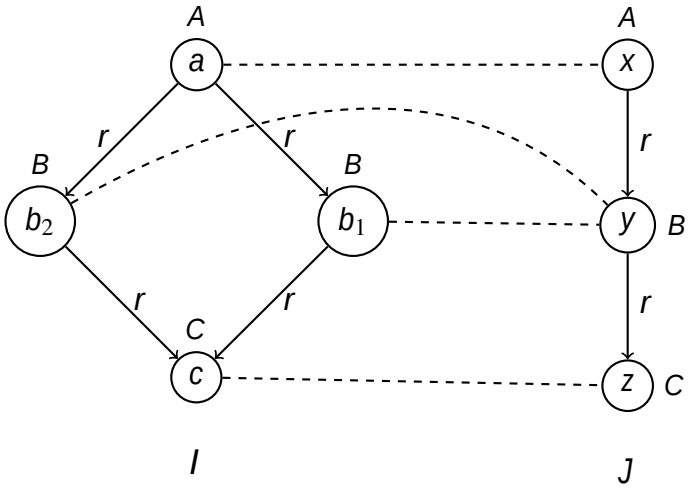

Figure 4.6: Bisimilar models $\mathcal{I}$ (left) and $\mathcal{J}$ (right).

Each circle represents a domain element whose name is written inside the circle. The circles are labelled with a concept name if it represents a domain element in the interpretation of this concept name. A solid arrow labelled with a role name connect two circles if the domain elements represented by the circles are related via r. The direction of the arrow defines the first and second position of the relation. For instance, $(a; b_1) \in r^{\mathcal{I}}$, but $(b_1; a) \notin r^{\mathcal{I}}$.

The models I and J in Figure 4.6 are $\{A;B;C\}$-bisimilar. The bisimulation relation R is described by the dashed lines. We have:

$$R = \{(a;x);(b_1;y);(b_2;y);(c;z)\}$$

The following lemma asserts that bisimulation is an equivalence relation.

Lemma 4.4.13. *If $(I_1;d_1);(I_2;d_2)$, and $(I_3;d_3)$ are three pointed interpretations, then the following properties of bisimulation are true.*

1. *Symmetry: $(I_1;d_1) \leftrightarrow_S (I_2;d_2)$ implies $(I_2;d_2) \leftrightarrow_S (I_1;d_1)$.*
2. *Transitivity: $(I_1;d_1) \leftrightarrow_S (I_2;d_2)$ and $(I_2;d_2) \leftrightarrow_S (I_3;d_3)$ implies $(I_1;d_1) \leftrightarrow_S (I_3;d_3)$.*

where S is a set of concept and role names.

Proof. Firstly, we prove the symmetry property. Suppose $(I_1;d_1) \leftrightarrow_S (I_2;d_2)$ is true, and $R \subseteq \Delta^{I_1} \times \Delta^{I_2}$ is the bisimulation relation. Define $R' \subseteq \Delta^{I_2} \times \Delta^{I_1}$ as the following relation.

$$R' = \{(e;d) \mid (d;e) \in R\} \tag{4.104}$$

We prove that the three conditions of Definition 4.4.11 are true for every $(e;d) \in R'$.

1. Condition 1: We see that if $(d;e) \in R$ then $e \in A^{I_2}$ if and only if $d \in A^{I_1}$. So, the condition is true for any $(e;d) \in R'$.
2. Condition 2: Suppose $(d;e) \in R$, and $(e;e') \in r^{I_2}$ for any role r. From Condition 3 of Definition 4.4.11, there must be $d' \in \Delta^{I_1}$, such that $(d;d') \in r^{I_1}$, and $(d';e') \in R$. So, the condition is true for any $(e;d) \in R'$.
3. Condition 3: Suppose $(d;e) \in R$, and $(d;d') \in r^{I_1}$ for any role r. From Condition 2 of Definition 4.4.11, there must be $e' \in \Delta^{I_2}$, such that $(e;e') \in r^{I_2}$, and $(d';e') \in R$. So, the condition is true for any $(e;d) \in R'$.

Secondly, we prove transitivity. Let $R_1 \subseteq \Delta^{I_1} \times \Delta^{I_2}$ be the bisimulation relation between $(I_1;d_1)$ and $(I_2;d_2)$, and $R_2 \subseteq \Delta^{I_2} \times \Delta^{I_3}$ be the bisimulation relation between $(I_2;d_2)$ and $(I_3;d_3)$. Define a relation R as follows.

$$R = \{(d;e) \mid (d;u) \in R_1 \text{ and } (u;e) \in R_2\}: \tag{4.105}$$

We show that R is a bisimulation relation between $(I_1;d_1)$ and $(I_3;d_3)$, by showing that the three conditions in Definition 4.4.11 are true for any $(d;e) \in R$.

1. Condition 1: Suppose $d \in A^{\mathcal{I}_1}$. From (4.105) we see that there is a domain element $u \in \Delta^{\mathcal{I}_2}$ such that $(d,u) \in R_1$ and $(u,e) \in R_2$. Then we must have that $u \in A^{\mathcal{I}_2}$ and $e \in A^{\mathcal{I}_3}$. Therefore, $d \in A^{\mathcal{I}_1}$ implies $e \in A^{\mathcal{I}_3}$ for any concept name $A \in \mathsf{S}$. The same argument shows that the reverse direction is true. That is, $e \in A^{\mathcal{I}_3}$ implies $d \in A^{\mathcal{I}_1}$ for any concept name $A \in \mathsf{S}$. So, Condition 1 is satisfied.

2. Condition 2: Since $(d,e) \in R$, from (4.105) there must be $u \in \Delta^{\mathcal{I}_2}$ where $(d,u) \in R_1$ and $(u,e) \in R_2$. Suppose $(d,d') \in r^{\mathcal{I}_1}$ for any role name r. We must have $u' \in \Delta^{\mathcal{I}_2}$, where $(u,u') \in r^{\mathcal{I}_2}$, and $(d',u') \in R_1$. Consequently, there is also $e' \in \Delta^{\mathcal{I}_3}$, where $(e,e') \in r^{\mathcal{I}_3}$, and $(u',e') \in R_2$. From (4.105), we see that $(d',e') \in R$. That is, if $(d,e) \in R$ and $(d,d') \in r^{\mathcal{I}_1}$, there is $e' \in \Delta^{\mathcal{I}_3}$ with $(e,e') \in r^{\mathcal{I}_3}$ and $(d',e') \in R$. So, Condition 2 is satisfied.

3. Condition 3 can be shown true with a similar argument to that used for Condition 2.

□

Lemma 4.4.14. *Let $(\mathcal{I},d_1),(\mathcal{J},d_2)$ be two pointed interpretations, and let S be some signature. If $(\mathcal{I},d_1),(\mathcal{J},d_2)$ are S-bisimilar then for every $\mathcal{ALC}$ concept C where $sig(C) \subseteq \mathsf{S}$ we have that $d_1 \in C^{\mathcal{I}}$ iff $d_2 \in C^{\mathcal{J}}$, in symbols $(\mathcal{I},d_1) \equiv_{\mathsf{S}} (\mathcal{J},d_2)$.*

An important subset of pointed interpretations is the subset consisting of *tree models*. These are tree-shaped pointed interpretations where each edge is labelled with one role name, and each node has at most one incoming edge. Because of these properties, tree models are more convenient algebraic structures than general pointed interpretations. Tree models can be obtained using the *tree unravelling* algorithm first shown in [DL59]. More recent descriptions of the algorithm can be found in [BvBW06, Pir13]. The algorithm unravels a pointed interpretation $(\mathcal{I},d)$ into a bisimilar tree-shaped interpretation $(\mathcal{I}',d)$ where $\Delta^{\mathcal{I}'}$ is defined as follows:

1. $d \in \Delta^{\mathcal{I}'}$; and

2. The word $w = d{:}r_1{:}d_1{:}r_2{:}d_2{:}\cdots{:}r_n{:}d_n$ is in $\Delta^{\mathcal{I}'}$ if and only if there is a path in $(\mathcal{I},d)$ from d to d_n along the edges $r_i \in N_r$ and the nodes $d_i \in \Delta^{\mathcal{I}}$ where $1 \leq i \leq n$.

For every concept name $A \in N_c \cup N_d$ and role name $r \in N_r$:

1. $A^{\mathcal{I}'} = \{d{:}r_1{:}\cdots{:}r_n{:}d_n \in \Delta^{\mathcal{I}'} \mid d_n \in A^{\mathcal{I}}\}$, and

2. $r^{\mathcal{I}'} = \{(w_1,w_1{:}r{:}d') \mid w_1,w_1{:}r{:}d' \in \Delta^{\mathcal{I}'} \wedge d' \in \Delta^{\mathcal{I}}\}$.

$(\mathcal{I}^0;d)$ can be seen as a tree whose nodes are paths in $(\mathcal{I};d)$ and there is a transition from one node to the other if the path represented by the later node is an increment of the path represented by the former node. By construction, $(\mathcal{I}^0;d)$ has the following properties:

1. Every node in $(\mathcal{I}^0;d)$ has exactly one predecessor, except the root node d which does not have a predecessor.

2. $r^{\mathcal{I}^0} \cap s^{\mathcal{I}^0} \neq \emptyset$ if and only if $r = s$ for all $r;s \in N_r$.

3. For every $e \in \Delta^{\mathcal{I}}$ there exists $e' \in \Delta^{\mathcal{I}^0}$ and vice versa such that $e \in C^{\mathcal{I}}$ if and only if $e' \in C^{\mathcal{I}^0}$ where C is an $\mathcal{ALC}$ concept.

4. If $(\mathcal{I};d)$ is a cyclic graph, then $(\mathcal{I}^0;d)$ is an acyclic tree with infinite depth.

Example 4.4.15. *Consider the model $\mathcal{I}$ from Example 4.4.12. We construct a tree interpretation $\mathcal{I}^0$ as follows:*

$$
\begin{aligned}
\Delta^{\mathcal{I}^0} &= \{a; a{:}r{:}b_1; a{:}r{:}b_2; a{:}r{:}b_1{:}r{:}c; a{:}r{:}b_2{:}r{:}c\} \\
A^{\mathcal{I}^0} &= \{a\} \\
B^{\mathcal{I}^0} &= \{a{:}r{:}b_1; a{:}r{:}b_2\} \\
C^{\mathcal{I}^0} &= \{a{:}r{:}b_1{:}r{:}c; a{:}r{:}b_2{:}r{:}c\} \\
r^{\mathcal{I}^0} &= \{(a; a{:}r{:}b_1); (a; a{:}r{:}b_2); (a{:}r{:}b_1; a{:}r{:}b_1{:}r{:}c); (a{:}r{:}b_2; a{:}r{:}b_2{:}c)\}
\end{aligned}
$$

A visualization of $\mathcal{I}$ and $\mathcal{I}^0$ is in Figure 4.7. Observe that $\mathcal{I}$ and $\mathcal{I}^0$ are $\{A;B;C\}$-bisimilar with $R = \{(a;a); (b_1; a{:}r{:}b_1); (b_2; a{:}r{:}b_2); (c; a{:}r{:}b_1{:}r{:}c); (c; a{:}r{:}b_2{:}r{:}c)\}$ being the bisimulation relation.

Back to Theorem 4.4.6 where the aim is to prove that $\mathcal{O}^{rp} \overset{C}{\underset{sig(\mathcal{O})\cap\mathcal{F}}{\equiv}} \mathcal{O}^d$, assume for the sake of contradiction that $\mathcal{O}^{rp} \not\equiv^{C}_{sig(\mathcal{O})\cap\mathcal{F}} \mathcal{O}^d$. We established through Lemma 4.4.9 that there is a witness concept inclusion $\alpha = C \sqsubseteq E$ over $sig(\mathcal{O})\cap\mathcal{F}$ where $\mathcal{O}^{rp} \models \alpha$ and $\mathcal{O}^d \not\models \alpha$. Our next step is to examine the models of $\mathcal{O}^d$ where α is false, and show that for every such model there is a bisimilar model of $\mathcal{O}^{rp}$. This gives a contradiction because $\mathcal{O}^{rp}$ entails α by assumption, and $\mathcal{O}^{rp}$ does not entail α because the bisimulation implies that there are models of $\mathcal{O}^{rp}$ where α is false. Let $\mathcal{I}$ be a model of $\mathcal{O}^d$ generated by an arbitrary $d \in C^{\mathcal{I}} \setminus :E^{\mathcal{I}}$, we have that $\mathcal{I} \in \text{mDiff}(\mathcal{O}^d;\mathcal{O}^{rp})$. The following Lemma sets a condition on $\mathcal{I}$.

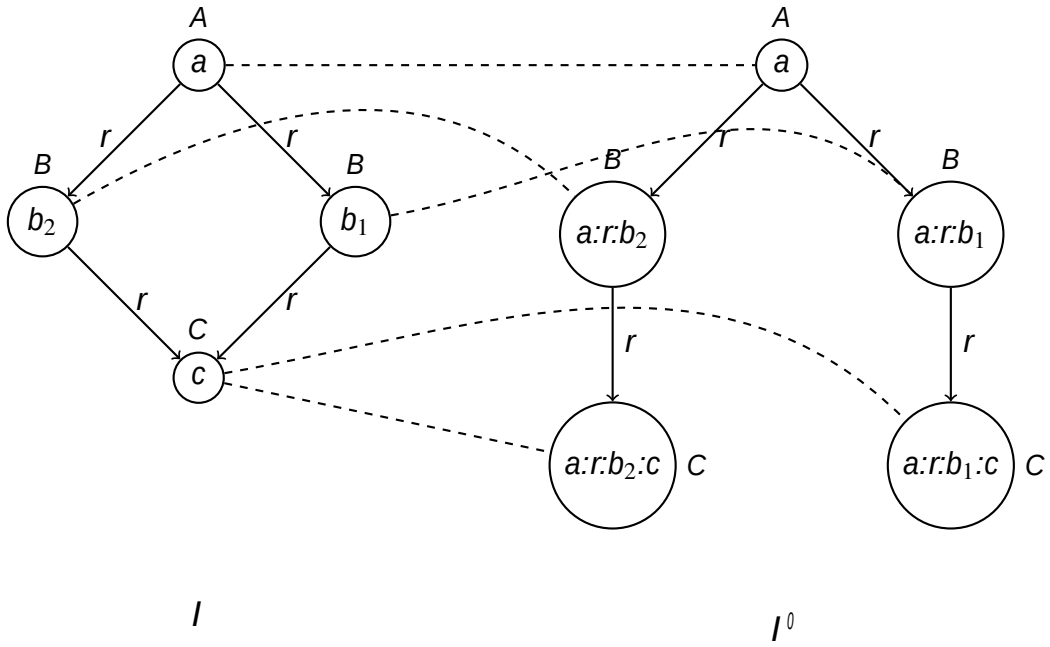

Figure 4.7: Tree unravelling of $\mathcal{I}$ (left) into $\mathcal{I}'$ (right).

Lemma 4.4.16. *Suppose $\mathcal{O}^{rp} = \mathcal{O}^{d} \cup \{ \top \sqsubseteq D_1 \sqcup \cdots \sqcup D_n \}$ where $n > 1$, and let $(\mathcal{I}, d)$ be as above. Then, $\mathcal{I} \in \text{mDiff}(\mathcal{O}^{d}, \mathcal{O}^{rp})$ if and only if there is $e \in \Delta^{\mathcal{I}}$ that is reachable from d where $e \in D_1^{\mathcal{I}} \setminus \cdots \setminus D_n^{\mathcal{I}}$ and $e \notin F^{\mathcal{I}}$.*

Proof. Right to left: Suppose there is $e \in \Delta^{\mathcal{I}}$ that is reachable from d where $e \in D_1^{\mathcal{I}} \setminus \cdots \setminus D_n^{\mathcal{I}}$ and $e \notin F^{\mathcal{I}}$. In other words, $\mathcal{I} \not\models \top \sqsubseteq D_1 \sqcup \cdots \sqcup D_n \sqcup F$. This implies that $\mathcal{I} \in \text{mDiff}(\mathcal{O}^{d}, \mathcal{O}^{rp})$, because $\top \sqsubseteq D_1 \sqcup \cdots \sqcup D_n \sqcup F \in \mathcal{O}^{rp}$ must be true in every model of $\mathcal{O}^{rp}$.

Left to right: Suppose $\mathcal{I} \in \text{mDiff}(\mathcal{O}^{d}, \mathcal{O}^{rp})$, and there is no such e as described in the lemma, then $\mathcal{I} \not\models \top \sqsubseteq D_1 \sqcup \cdots \sqcup D_n \sqcup F$. But $\mathcal{I} \models \mathcal{O}^{d}$, so we get that $\mathcal{I} \models \mathcal{O}^{rp}$ which contradicts that $\mathcal{I} \in \text{mDiff}(\mathcal{O}^{d}, \mathcal{O}^{rp})$. □

Definition 4.4.17. *Let $\mathcal{I}$ be a model and $e \in \Delta^{\mathcal{I}}$. Recall that N_d is the set of definer symbols, and define $C_{\mathcal{I}}(e)$ to be the symbols in the closure of $N_c \cup N_d$ under single negation that contain e in their interpretation in $\mathcal{I}$.*

Example 4.4.18. *Consider $\mathcal{I}'$ from Example 4.4.15. Suppose $N_c = \{A, B, C\}$, and $N_d = \{D\}$. We have $C_{\mathcal{I}'}(a:r:b_1) = \{B, \neg A, \neg C, \neg D\}$.*

Lemma 4.4.19. *Let $\mathcal{O}^{rp}, \mathcal{O}^{d}$, and $\mathcal{I}$ be defined as in Lemma 4.4.16. There is a model $\mathcal{J}$ generated by d such that:*

1. $\mathcal{J} \models \mathcal{O}^d$;

2. $(\mathcal{I}, d) \equiv_{sig(\mathcal{O}^d) \cap N_d} (\mathcal{J}, d)$; and

3. There is no $e \in \Delta^{\mathcal{J}}$ such that $e \in D_1^{\mathcal{J}} \sqcap \ldots \sqcap D_n^{\mathcal{J}} \sqcap (\neg F)^{\mathcal{J}}$.

Proof. Let $\mathcal{I}_0$ be a model that coincides with $\mathcal{I}$ on everything but reinterprets the definer symbols as follows:

$$D^{\mathcal{I}_0} := D^{\mathcal{I}} \setminus \{y \in \Delta^{\mathcal{I}_0} \mid (x,y) \in r^{\mathcal{I}_0} \text{ and } x \notin G^{\mathcal{I}_0} \text{ and } (\mathcal{O}^d \models G \sqsubseteq \exists r.D; \text{ or } \mathcal{O}^d \models G \sqsubseteq \forall r.D)\} \quad (4.106)$$

where $D \in N_d, r \in N_r$, and G is an $\mathcal{ALC}$ concept.

The idea of (4.106) is to restrict the elements in the extension of D to the minimum set that is required for $\mathcal{O}^d$. We explain this idea more through the following example

Example 4.4.20. *Suppose $\mathcal{O}^d$ is the ontology:*

$$\neg A \sqsubseteq \forall r.D_1$$
$$\neg B \sqsubseteq \forall r.D_2$$
$$\neg D_1 \sqsubseteq \neg D_2$$
$$\neg A \sqsubseteq \neg B \sqsubseteq \forall r.\bot$$

The model $\mathcal{I}$ with $\Delta^{\mathcal{I}} = \{a,b\}; A^{\mathcal{I}} = \{a\}; B^{\mathcal{I}} = \{g\}; D_1^{\mathcal{I}} = D_2^{\mathcal{I}} = \{b\}; r^{\mathcal{I}} = \{(a,b)\}$ is a tree model of $\mathcal{O}^d$. But $D_2^{\mathcal{I}} = \{b\}$, which violates the intuition that D_2 represents the r-successors of elements in $B^{\mathcal{I}}$. Re-interpreting D_2 using (4.106) removes b from $D_2^{\mathcal{I}}$.

Continuing with the proof of Lemma 4.4.19, recall that in $\mathcal{O}^{rp}$ a definer symbol D exists positively only in clauses of the form $C \sqsubseteq \exists r.D$ or $C \sqsubseteq \forall r.D$, and negatively only in clauses of the form $\neg D \sqsubseteq C$. Then, it can be seen that $\mathcal{I}_0$ is a model of $\mathcal{O}^d$ since all clauses of $\mathcal{O}^d$ on the forms of $G \sqsubseteq \exists r.D$ and $G \sqsubseteq \forall r.D$ are true in $\mathcal{I}_0$ by the side conditions of (4.106). Also, since (4.106) only removes elements from the extension of D, clauses on the form of $\neg D \sqsubseteq H$ remain satisfied in $\mathcal{I}_0$. As (4.106) only modifies the interpretations of definer symbols, it also follows that $(\mathcal{I}, d) \equiv_{sig(\mathcal{O}^d) \cap N_d} (\mathcal{I}_0, d)$.

We shall now construct a sequence of interpretations $\mathcal{I}_1, \mathcal{I}_2, \ldots$. The interpretations are constructed such that $\mathcal{I}_{k+1}$ is a transformation of $\mathcal{I}_k$ that eliminates one domain element $e \in D_1^{\mathcal{I}_k} \sqcap \ldots \sqcap D_n^{\mathcal{I}_k} \sqcap (\neg F)^{\mathcal{I}_k}$ where $k \geq 0$, and $(\mathcal{I}_k, d) \equiv_{sig(\mathcal{O}^d) \cap N_d} (\mathcal{I}_{k+1}, d)$. The

limit of this sequence is $\mathcal{J}$. It then follows by transitivity of that $(\mathcal{I};d)$ $\sim_{sig(\mathcal{O}^d)\cap N_d}$ $(\mathcal{J};d)$, and there is no $e \in \Delta^{\mathcal{J}}$ such that $e \in D_1^{\mathcal{J}} \sqcap \ldots \sqcap D_n^{\mathcal{J}} \sqcap (\neg F)^{\mathcal{J}}$.

Suppose there is a domain element $e \in \Delta^{\mathcal{I}_k}$ such that $e \in D_1^{\mathcal{I}_k} \sqcap \ldots \sqcap D_n^{\mathcal{I}_k} \sqcap (\neg F)^{\mathcal{I}_k}$. It is guaranteed by (4.106) that e has a predecessor, call it e_{pre}, such that for every i with $1 \leq i \leq n$ we have $\mathcal{O}^d \models G_i \sqcap \exists r.D_i$ or $\mathcal{O}^d \models G_i \sqcap \forall r.D_i$, and $e_{pre} \notin G_i^{\mathcal{I}_k}$. Note that the case when $\mathcal{O}^d \models G_i \sqcap \forall r.D_i$ for all i with $1 \leq i \leq n$ is not possible since by the construction of $\mathcal{O}^d$ the clause $G_1 \sqcap \ldots \sqcap G_n \sqcap \forall r.F$ must have been introduced in $\mathcal{O}^d$ by the *Role Propagation* rule in Figure 2. Since $e_{pre} \notin G_i^{\mathcal{I}_k}$ with $1 \leq i \leq n$, it must be that $e_{pre} \in (\forall r.F)^{\mathcal{I}_k}$, hence $e \in F^{\mathcal{I}_k}$ which contradicts the hypothesis that $e \notin F^{\mathcal{I}_k}$. For the other cases we transform $\mathcal{I}_k$ to $\mathcal{I}_{k+1}$ as follows:

(I) Suppose that there is an l such that $1 \leq l \leq n$ with $\mathcal{O}^d \models G_l \sqcap \exists r.D_l$ and $\mathcal{O}^d \models G_i \sqcap \forall r.D_i$ where $1 \leq i \leq n$ and $i \neq l$. Again from the *Role Propagation* rule we must have $\mathcal{O}^d \models G_1 \sqcap \ldots \sqcap G_n \sqcap \exists r.(F \sqcap \bigsqcap_{j=1}^{m} F_j)$, where $\mathcal{O}^{rp} \models P_j \sqcap F_j$ and P_j is any sub-concept of $\neg D_1 \sqcap \ldots \sqcap \neg D_n$ (these are the second premise of the *Role Propagation* rule). Since $e_{pre} \notin G_i^{\mathcal{I}_k}$ with $1 \leq i \leq n$, it must be that $e_{pre} \in (\exists r.(F \sqcap \bigsqcap_{j=1}^{m} F_j))^{\mathcal{I}_k}$. That is, there is a child e' of e_{pre} via r, such that $e' \in D_i^{\mathcal{I}_k}$ where $1 \leq i \leq n$ and $i \neq l$, and $e' \in F^{\mathcal{I}_k}$.

We construct a model $\mathcal{I}_{k+1}$ which is equivalent to $\mathcal{I}_k$ in everything except D_l which it interprets differently: If $e' \in D_l^{\mathcal{I}_k}$, then $D_l^{\mathcal{I}_{k+1}} := D_l^{\mathcal{I}_k} \setminus \{e\}$. If $e' \notin D_l^{\mathcal{I}_k}$, then $D_l^{\mathcal{I}_{k+1}} := (D_l^{\mathcal{I}_k} \cup \{e'\}) \setminus \{e\}$.

We show that $\mathcal{I}_{k+1}$ is a model of $\mathcal{O}^d$.

1. By removing e from the interpretation of D_l, we need only to worry about clauses of $\mathcal{O}^d$ where D_l appears positively, that is, we show that clauses of the form $G \sqcap \exists r.D_l$ are satisfied at e_{pre}. Since by the transformation above $e' \in D_l^{\mathcal{I}_{k+1}}$ and $(e_{pre}, e') \in r^{\mathcal{I}_{k+1}}$, we have $e_{pre} \in (\exists r.D_l)^{\mathcal{I}_{k+1}}$. Therefore $e_{pre} \in (G \sqcap \exists r.D_l)^{\mathcal{J}_d}$ for any $\mathcal{ALC}$ concept G.

2. By adding e' to the interpretation of D_l, we need only to worry about clauses of $\mathcal{O}^d$ where D_l appears negatively, that is, we show that $e' \in (\neg D_l \sqcap G)^{\mathcal{I}_{k+1}}$ for any $\mathcal{ALC}$ concept G. Since $\neg D_l \sqcap G \in \mathcal{O}^d$, it must have been a premise for the *Role Propagation* rule, i.e., there must be a j such that $1 \leq j \leq m$ such that $G = F_j$. Since $e' \in (F \sqcap \bigsqcap_{j=1}^{m} F_j)^{\mathcal{I}_{k+1}}$, then $e' \in F_j$, consequently $e' \in (\neg D \sqcap G)^{\mathcal{I}_{k+1}}$.

As the transformation only changes the interpretation of D_l, we conclude that the models $\mathcal{I}_k$ and $\mathcal{I}_{k+1}$ are $sig(\mathcal{O}^d) \setminus \{D_l\}$-bisimilar.

(II) Suppose that $\mathcal{O}^d \models G_i \sqsubseteq \exists r.D_i$ with $1 \leq i \leq p$ and $2 \leq p \leq n$, and $\mathcal{O}^d \models G_j \sqsubseteq \forall r.D_j$ with $p < j \leq n$. That is, two or more definers in $D_1, \ldots, D_n$ occur under existential role restriction. In this case the *Role Propagation* rule in Figure 2 does not apply.

We construct a model $\mathcal{I}_{k+1}$ which is equivalent to $\mathcal{I}_k$ in everything except that it replaces e with fresh domain elements e_i where $1 \leq i \leq p$ such that $e_i \in D_i^{\mathcal{I}_{k+1}}$, $e_i \in D_j^{\mathcal{I}_{k+1}}$ with $1 \leq j \leq p$ and $j \neq i$, and:

1. $e_i \in A^{\mathcal{I}_{k+1}}$ iff $A \in C_{\mathcal{I}_k}(e) \cap \{D_i\}$ for every $A \in N_c \cup N_d$;
2. $(e_{pre}, e_i) \in r^{\mathcal{I}_{k+1}}$;
3. $(e_i, e') \in r^{\mathcal{I}_{k+1}}$ iff $(e, e') \in r^{\mathcal{I}_k}$.

where $r \in N_r$.

We prove first that $\mathcal{I}_{k+1} \models \mathcal{O}^d$:

1. As before, by removing e from $\Delta^{\mathcal{I}_{k+1}}$ we need only to show that clauses of $\mathcal{O}^d$ of the form $G_i \sqsubseteq \exists r.D_i$ where $1 \leq i \leq p$ are satisfied at e_{pre}. By construction, e is replaced with e_i where $1 \leq i \leq p$ such that $e_i \in D_j^{\mathcal{I}_{k+1}}$ with $1 \leq j \leq p$ and $j \neq i$. So for every D_i with $1 \leq i \leq p$ there are $p - 1$ new domain elements e_j such that $e_j \in D_i^{\mathcal{I}_{k+1}}$ and $(e_{pre}, e_j) \in r^{\mathcal{I}_{k+1}}$. Therefore, $e_{pre} \in (\exists r.D_i)^{\mathcal{I}_{k+1}}$ consequently $e_{pre} \in (G_i \sqsubseteq \exists r.D_i)^{\mathcal{I}_{k+1}}$ for $1 \leq i \leq p$.
2. By introducing new elements e_i such that $e_i \in D_i^{\mathcal{I}_{k+1}}$ with $1 \leq i \leq p$, we need to show that if $\neg D_i \sqsubseteq G_i \in \mathcal{O}^d$ then $e_i \in G_i^{\mathcal{I}_{k+1}}$. Since $e \in G_i^{\mathcal{I}_k}$, it follows from the first condition of the transformation that $e_i \in G_i^{\mathcal{I}_{k+1}}$.

Altogether, $\mathcal{I}_{k+1} \models \mathcal{O}^d$.

We prove second that $(\mathcal{I}_k, d) \leftrightarrow_{sig(\mathcal{O}^d) \cap N_d} (\mathcal{I}_{k+1}, d)$ by induction from bottom to top.

1. Assume that e, (hence e_i) is reachable from d in m transitions.

   Let $S = sig(\mathcal{O}^d) \cap \{D_1, \ldots, D_n\}$. By construction, for any node $u \in (\mathcal{I}_k, d)$ or $(\mathcal{I}_{k+1}, d)$ such that u is reachable from d in at least $m + 1$ transitions, we have $(\mathcal{I}_k, u) \leftrightarrow_S (\mathcal{I}_{k+1}, u)$. That is, nothing below e and e_i has changed.

2. For every $u \in \mathcal{I}_k$ at depth m we have:

   (a) If $u = e$, then by construction $C_{\mathcal{I}_k}(u) \cap \{D_i\} = C_{\mathcal{I}_{k+1}}(e_i)$. Also, since the third condition of the transformation guarantees that everything below u and e_i is the same, it follows that $(\mathcal{I}_k, u) \leftrightarrow_S (\mathcal{I}_{k+1}, e_i)$.

(b) If $u \neq e$, then u is also a node in $(\mathcal{I}_{k+1}; d)$, and $(\mathcal{I}_k; u) \leftrightarrow_S (\mathcal{I}_{k+1}; u)$.

3. For every $u \in (\mathcal{I}_k; d)$ such that u is reachable in at most $m - 1$ transitions from d we have that $u \in (\mathcal{I}_{k+1}; d)$, and we have the following cases:

   (a) $e \notin (\mathcal{I}_k; u)$. Then, as before $(\mathcal{I}_k; u) \leftrightarrow_S (\mathcal{I}_{k+1}; u)$.

   (b) $e \in (\mathcal{I}_k; u)$. Condition 1 in the transformation guarantees that $C^{\mathcal{I}_k}(u) = C^{\mathcal{I}_{k+1}}(u)$ on all $C \in (N_c \cup N_d)$.

   Suppose that $u \in (\exists r.C)^{\mathcal{I}_k}$ where $\exists r = \exists r_1.\exists r_2.\ldots.\exists r_l$, $r_i \in N_r$, and C is any $\mathcal{ALC}$ concept over S. Then there must be a path $ur_1u_1r_2u_2r_3\ldots r_lu_l$ in $\mathcal{I}_k$ such that $u_l \in C^{\mathcal{I}_k}$. If $u_i \neq e$ for every $i \in [1..l]$ then by construction this path also exists in $(\mathcal{I}_{k+1}; d)$ and $u \in (\exists r.C)^{\mathcal{I}_{k+1}}$. If $u_i = e$ for any $i \in [1..l]$, then again by construction there is a path $ur_1u_1r_2\ldots r_ivr_{i+1}\ldots r_lu_l$ and we have the choice to set v to any of $\{e_1, \ldots, e_n\}$. In particular, we have $u_l \in C^{\mathcal{I}_{k+1}}$ and $u \in (\exists r.C)^{\mathcal{I}_{k+1}}$. We get that $(\mathcal{I}_k; u) \leftrightarrow_S (\mathcal{I}_{k+1}; u)$ for every u reachable in at most $m - 1$ transitions from d.

4. The same arguments as above can be used to show that every node in $(\mathcal{I}_{k+1}; d)$ has a bisimilar node in $(\mathcal{I}_k; d)$.

Altogether, we get that $\mathcal{I}_k \leftrightarrow_S (\mathcal{I}_{k+1}; d)$.

In the sequence of constructed models, it follows that:

1. $\mathcal{J} \models \mathcal{O}^d$;
2. $\mathcal{I} \leftrightarrow_{sig(\mathcal{O}^d) \cap N_d} \mathcal{J}$; and
3. $\mathcal{J} \not\models D_1 \sqcap D_2 \sqcap \ldots \sqcap D_n \sqcap \neg F$.

□

Continuing with the proof of Theorem 4.4.6, since the interpretation $\mathcal{J}$ constructed by Lemma 4.4.19 is a model of $\mathcal{O}^d$, and there is no $e \in \Delta^{\mathcal{J}}$ such that $e \in D_1^{\mathcal{J}} \setminus \ldots \setminus D_n^{\mathcal{J}} \setminus (\neg F)^{\mathcal{J}}$, it follows by Lemma 4.4.16 that $\mathcal{J} \models \mathcal{O}^{rp}$. Also, since $\mathcal{I} \leftrightarrow_{sig(\mathcal{O}^d) \cap N_d} \mathcal{J}$, and $sig(\mathcal{O}^d) \cap N_d = sig(\mathcal{O}) \cap F$, it follows by Lemma 4.4.14 that $\mathcal{J} \models C \sqsubseteq E$. But this contradicts the assumption that $\mathcal{O}^{rp} \models C \sqsubseteq E$ implying that the assumption that there is $\alpha = C \sqsubseteq E$ with $\mathcal{O}^{rp} \models \alpha$ and $\mathcal{O}^d \not\models \alpha$ is incorrect, and $\mathcal{O}^{rp} \overset{C}{\leftrightarrow}_{sig(\mathcal{O}^d) \cap N_d} \mathcal{O}^d$.

Finally assume $\mathcal{O}^{rp} = \mathcal{O}^d \cup \Delta^d$ with Δ^d being a set of n clauses of the form $\neg D_1 \sqcup \ldots \sqcup \neg D_n \sqcup F$. We construct n ontologies $\mathcal{O}_i^{rp}$ where $\mathcal{O}_n^{rp} = \mathcal{O}^{rp}$, $\mathcal{O}_k^{rp} = \mathcal{O}_{k-1}^{rp} \cup \{S\}$ where

S is a clause in D^d and $\mathcal{O}_0^{rp} = \mathcal{O}^d$. By the above proof we have $\mathcal{O}_k^{rp} \overset{C}{\underset{sig(\mathcal{O}^d)nN_d}{\equiv}} \mathcal{O}_{k-1}^{rp}$ for $1 \leq k \leq n$. By transitivity of $\overset{C}{\equiv}$ and because $sig(\mathcal{O}^d)nN_d = sig(\mathcal{O})nF$ we conclude $\mathcal{O}^{rp} \overset{C}{\underset{sig(\mathcal{O})nF}{\equiv}} \mathcal{O}^d$.

We end this section noting that the deductive reduction process is terminating as can be seen from the following.

1. The first pass of the reduction process where *Role Propagation* inferences are performed is terminating because we perform one *Role Propagation* inference for every trigger clause $P_0 \sqsubseteq C_0$ and there is only a finite number of trigger clauses in D^d.

2. The second pass of the reduction process where the D^d clauses are removed from $\mathcal{O}^{rp}$ is terminating because the clauses of D^d are finite.

4.4.3 Notes on The Deductive Reduction Process

Observe that $\mathcal{O}^{rp} = \mathcal{O}^d \cup \mathsf{D}^d$. Additionally, the clauses in D^d have been generated in $\mathcal{O}^{int}$ by resolution inferences over the forgetting symbols, and the premises of these inferences were removed by purity deletion. Therefore, we find in general that $\mathcal{O}^d \not\models \mathsf{D}^d$. This implies that D^d can be viewed as representing the information difference between $\mathcal{O}^{int}$ and $\mathcal{O}^d$. We can view D^d therefore as the information difference between $\mathcal{O}$ and the deductive view with respect to $sig(\mathcal{O})nF$.

The set D^d consists of clauses of the format $\neg D_1 \sqcap \cdots \sqcap \neg D_n \sqsubseteq F$ where $n \geq 2$, or in axiom form, $D_1 \sqcap \cdots \sqcap D_n \sqsubseteq F$. Since we introduced the definer symbols to represent subsets of role successors, these clauses can be understood as information on the conjunctions of different subsets of role successors. For instance, in Example 4.3.1, the clause $\neg D_1 \sqcap \neg D_2 \sqsubseteq \mathsf{D}^d$ specifies the constraint that the subset of r-successors and the subset of s-successors of domain elements in the interpretation of A are disjoint. It was not a coincidence that we introduced definer symbols to represent subsets of role successors. Using them in this way and introducing them via structural transformation forces the clauses in D^d to be explicit members of $\mathcal{O}^{int}$ which simplifies their extraction, giving us a representation of the difference between $\mathcal{O}$ and the deductive view. In other words, our adaptation of the method of [LR94a] is implemented in a non-trivial way that allows for comparing the information preserved by the semantic forgetting view and the deductive forgetting view of the input ontology with respect to the given forgetting signature.

4.5 Stage Three: Definer Elimination

In the previous section, D^d was identified as the set consisting of axioms that use two or more negative definers. The reduced ontology $\mathcal{O}^d$ was identified as the subset of $\mathcal{O}^{int}$ enhanced with the conclusions of the *Role Propagation* inferences, and disjoint with D^d. While $\mathcal{O}^d$ may use definers, most of these definers can be eliminated without violating concept inseparability from $\mathcal{O}^{int}$. The process to eliminate these definers is presented in this section. The output of the process is an ontology $\mathcal{V}^d$, which is model inseparable from $\mathcal{O}^d$ with respect to the non-forgotten vocabulary $sig(\mathcal{O}) \setminus \mathsf{F}$, and deductively inseparable from $\mathcal{O}$ with respect to $sig(\mathcal{O}) \setminus \mathsf{F}$. If all definers have been eliminated from $\mathcal{V}^d$, then $\mathcal{V}^d$ is a deductive forgetting view of the input ontology $\mathcal{O}$ with respect to the forgetting signature F. Otherwise, $\mathcal{V}^d$ is an approximation to the deductive forgetting views, because it is deductively inseparable from the input ontology with respect to the non-forgotten signature.

There are reasons why definers cannot be eliminated completely. Recall from the literature that a deductive forgetting view of an $\mathcal{ALC}$ ontology with respect to a forgetting signature F does not always exist [LR94a, ZZ09, LW11, KS13c] (see also Chapter 3). Eliminating all definers would be contradictory to the literature, as it implies that a deductive forgetting view is obtainable for every input ontology and forgetting signature. Moreover, due to the triple exponential complexity of forgetting [LW11], a deductive forgetting view can exist theoretically but not practically.

There is a particular class of definers that we cannot eliminate. We call it the class of *cyclic definers*.

Definition 4.5.1. *A definer is called* cyclic definer *if it occurs both positively and negatively in a clause.*

Example 4.5.2. *Consider the following ontology* $\mathcal{O}$

$$A \sqsubseteq B$$
$$B \sqsubseteq C$$
$$C \sqsubseteq E$$
$$E \sqsubseteq \exists r.B$$

and the forgetting signature $\mathsf{F} = \{B, C, E\}$. *We have* $\mathcal{O}^{int}$ *and* $\mathcal{O}^d$ *as the following*

ontology

$$\begin{array}{l} \bot \sqsubseteq A \sqcup \exists r.D \\ \bot \sqsubseteq D \sqcup \exists r.D \end{array}$$

where $D \in N_d$ is a definer. D is a cyclic definer because it occurs both positively and negatively in the clause $\bot \sqsubseteq D \sqcup \exists r.D$.

The presence of cyclic definers indicates syntactic cycles over some forgetting symbols. These are axioms of the ontology whose strongest consequences [DŁS01] with respect to a set of forgetting symbols are defined in terms of some of these forgetting symbols.

Example 4.5.3. *Continuing with Example 4.5.2. The strongest consequence of the axioms $\{B \sqsubseteq C; C \sqsubseteq E; E \sqsubseteq \exists r.B\}$ with respect to the forgetting signature $\mathbf{F} = \{B; C; E\}$ is $\exists r.B$, which has the forgetting symbol B in it.*

Elimination of cyclic definers can result in infinite forgetting views. For instance, eliminating D from $\mathcal{O}^{int}$ from Example 4.5.2 would have resulted in the following infinite forgetting view:

$$\begin{array}{l} A \sqsubseteq \exists r.\top \\ A \sqsubseteq \exists r.\exists r.\top \\ A \sqsubseteq \exists r.\exists r.\exists r.\top \\ \vdots \end{array}$$

Different approaches have been followed in the literature to obtain a finite representation of the forgetting view. One such approach is using fixpoint operators [NS98]. For instance the forgetting view above can be finitely represented as $\nu X.\exists r.X$ where ν is the greatest fix-point operator.

Another approach has been proposed in [KS13c] to leave the cyclic definers to remain as witnesses on the syntactic cycles in the generated forgetting view. Similar to what was said earlier about $\mathcal{O}^{int}$, the use of definers in the final ontology $\mathcal{V}^d$ violates Definition 3.5.14, and so $\mathcal{V}^d$ would not be a deductive forgetting view of $\mathcal{O}$ with respect to $\mathbf{F}$. It remains however a faithful representation of the deductive forgetting views of $\mathcal{O}$ with respect to $\mathbf{F}$.

Non-cyclic definers can be eliminated safely while preserving the interpretations of the non-forgotten vocabulary. For this, we use the rules in Figure 4.8.

Definer Elimination

$$\frac{\mathcal{O} \cup \{ D \sqsubseteq C_1, \ldots, D \sqsubseteq C_n \}}{\mathcal{O}[D/C]}$$

where $C = \bigsqcup_{i=1}^{n} C_i$ and $D \in sig(C)$, C does not contain any negative definers, and $\mathcal{O}$ does not contain D negatively.

Positive Definer Purification

$$\frac{\mathcal{O}}{\mathcal{O}[D/\top]}$$

where $D \in N_d$ is a definer symbol, and D does not occur negatively in $\mathcal{O}$.

Negative Definer Purification

$$\frac{\mathcal{O}}{\mathcal{O}[D/\bot]}$$

where $D \in N_d$ is a definer symbol, and D does not occur positively in $\mathcal{O}$.

Figure 4.8: Definer elimination rules

The *Positive Definer Elimination* and the *Negative Definer Elimination* rules are the basic purification rules encountered earlier in Figure 4.3. They are used to eliminate definers that exist only positively or only negatively in an ontology.

Non-cyclic definers that occur both positively and negatively in an ontology are eliminated using the *Definer Elimination* rule. The side conditions of the *Definer Elimination* rule ensures that the definer being eliminated is not cyclic. The rule replaces the definer symbol D with its super-concept $C_1 \sqcap \ldots \sqcap C_n$. Note that, in the *Definer Elimination* rule, C may be $\bot$.

Example 4.5.4. *Continuing with Example 4.4.5, $\mathcal{V}^d$ is extracted from $\mathcal{O}^d$ as follows:*

1. *The definers D_2 and D_3 are eliminated using* Purification. *Since D_2 and D_3 appear only positively in $\mathcal{O}^d$, they are purified by replacing them with $\top$ which gives:*

$$\neg A \sqsubseteq \forall s.\top$$
$$\neg A \sqsubseteq \forall r.D_1$$
$$\neg G \sqsubseteq \exists r.\top$$
$$\neg D_1 \sqsubseteq H$$

$$:A \sqsubseteq :G \sqsubseteq \exists r.(C \sqcup H)$$

As $\forall s.\top$ evaluates to $\top$, the result can be simplified further to:

$$:A \sqsubseteq \forall r.D_1$$
$$:G \sqsubseteq \exists r.\top$$
$$:D_1 \sqsubseteq H$$
$$:A \sqsubseteq :G \sqsubseteq \exists r.(C \sqcup H)$$

2. *The definer D_1 is eliminated by the* Definer Elimination *rule in Figure 4.8 giving:*

$$:A \sqsubseteq \forall r.H$$
$$:G \sqsubseteq \exists r.\top$$
$$:H \sqsubseteq H$$
$$:A \sqsubseteq :G \sqsubseteq \exists r.(C \sqcup H)$$

The clause $:H \sqsubseteq H$ is a tautology so the final forgetting view V^d is:

$$:A \sqsubseteq \forall r.H$$
$$:G \sqsubseteq \exists r.\top$$
$$:A \sqsubseteq :G \sqsubseteq \exists r.(C \sqcup H)$$

Theorem 4.5.5. *Let V^d be generated from O^d by applying the Definer Elimination rule from Figure 4.8 exhaustively. We have:*

1. $O^d \underset{sig(O) \cap F}{\equiv_M} V^d$;
2. *if $sig(V^d) \setminus N_d = \emptyset$ then V^d is a deductive forgetting view of O with respect to F.*

Proof. First we prove Clause 1. The Definer Elimination rule in Figure 3 can be seen as a two step operation. The first replaces all clauses of the form $:D \sqsubseteq C_i$ with a single clause $:D \sqsubseteq C$ where $C = \sqcup C_i$. This step clearly preserves equivalence. The second replaces every D in O with the concept C. This step is the inverse of structural transformation. Therefore, by Lemma 4.2.4 we get that $O^d \underset{sig(V^d)}{\equiv_M} V^d$.

Second we prove Clause 2. From Theorems 4.3.3, 4.4.6, and Clause 1 proved above, we have that:

1. $\mathcal{O} \underset{sig(\mathcal{O})nF}{\overset{M}{}} \mathcal{O}^{int}$.
2. $\mathcal{O}^{int} \underset{sig(\mathcal{O})nF}{\overset{C}{}} \mathcal{O}^{d}$.
3. $\mathcal{O}^{d} \underset{sig(\mathcal{O})nF}{\overset{M}{}} \mathcal{V}^{d}$.

It follows from these that $\mathcal{O} \underset{sig(\mathcal{O})nF}{\overset{C}{}} \mathcal{V}^{d}$. If additionally $sig(\mathcal{V}^{d}) \setminus N_d = \emptyset$ then by Definition 3.5.14 we have $\mathcal{V}^{d}$ is a deductive view of $\mathcal{O}$ with respect to F. □

It can be seen from the following that the definer elimination process is terminating. First, the purification rules substitute $>$ for a definer D if it occurs only positively and $?$ if it occurs only negatively in $\mathcal{O}^{d}$. Since there is a finite number of definers introduced in Stage 1 of the framework, we see that the two purification rules can be exhaustively run in finite time. Second, the *Definer Elimination* rule combines the clauses where D occurs negatively in one clause $:D \sqcup C$ and replaces every occurrence of D with C. The definer D does not occur in C because we disallow eliminating cyclic definers, but C can contain other definers. Let $<$ denote a lexical ordering. Suppose for every clause $:D \sqcup C \in \mathcal{O}^{d}$ where $D \in N_d$ is a definer we have that if $D' \in sig(C)$ then $D < D'$ Then definer elimination is terminating because the elimination of a definer can only introduce another definer that is higher in the lexical ordering. In other cases we may have $D < D'$ in some clauses but $D' > D$ in some other clauses. For instance, suppose we have the following three clauses.

$$:D_1 \sqcup \exists r.D_2 \tag{4.107}$$

$$:D_2 \sqcup \exists r.D_3 \tag{4.108}$$

$$:D_3 \sqcup \exists r.D_1 \tag{4.109}$$

Let us eliminate the definers according to their lexical ordering $D_1 < D_2 < D_3$. Eliminating D_1 substitutes the clause $:D_3 \sqcup \exists r.D_2$ for the clause (4.109) and removes the clause (4.107). Eliminating D_2 substitutes the clause $:D_3 \sqcup \exists r.D_3$ for $:D_3 \sqcup \exists r.D_2$. At this point we find that the definer D_3 is a cyclic definer and we stop the elimination process. Altogether we find that the definer elimination process is terminating.

4.6 Cyclic Definers

The deductive forgetting method described in the previous sections allows cyclic definers to occur in $\mathcal{V}^{d}$ as witnesses on syntactic cycles in the original ontology over

some forgetting symbols. Example 4.5.3 shows that eliminating cyclic definers may result in an infinite forgetting view. However, there are cases where cyclic definers can be eliminated without resulting in an infinite forgetting view as shown in the following examples.

Example 4.6.1. *Consider the following ontology* $\mathcal{O}$

$$B \sqsubseteq C \sqcap \exists r.B$$
$$C \sqsubseteq \top;$$

and the forgetting signature $\mathbf{F} = \{B\}$*. With our method we obtain* $\mathcal{O}^{int}$ *as:*

$$\neg D \sqcup C$$
$$\neg D \sqcup \exists r.D$$
$$\neg C$$

where D is a definer symbol. We also have, $\mathrm{D}^d = \emptyset$*, and* $\mathcal{V}^d = \mathcal{O}^d = \mathcal{O}^{int}$ *because D is a cyclic definer.*

However, $\{C \sqsubseteq \top\}$ *is a deductive forgetting view of* $\mathcal{O}$ *with respect to* $\mathbf{F}$*. That is, although* $\mathcal{V}^d$ *contains a cyclic definer which our method would have failed to eliminate, a finite deductive view* $\mathcal{UI} = \{C \sqsubseteq \top\}$ *exists.*

Example 4.6.2. *Consider the following ontology* $\mathcal{O}$

$$A \sqsubseteq B$$
$$B \sqsubseteq C \sqcap \forall r.B$$
$$C \sqsubseteq \forall r.\forall r.\forall r.\bot$$

and the forgetting signature $\mathbf{F} = \{B\}$*. We have* $\mathcal{O}^{int}$ *as the ontology:*

$$\neg A \sqcup C$$
$$\neg A \sqcup \forall r.D$$
$$\neg D \sqcup C$$
$$\neg D \sqcup \forall r.D$$
$$\neg C \sqcup \forall r.\forall r.\forall r.\bot$$

where $D \in \mathbf{N}_d$ *is a cyclic definer symbol. We also have* $\mathrm{D}^d = \emptyset$*, and* $\mathcal{V}^d = \mathcal{O}^d = \mathcal{O}^{int}$

because D is cyclic.

Although our method would fail to eliminate the cyclic definer D, one can observe that the following ontology $\mathcal{U}$ is a finite deductive forgetting view of $\mathcal{O}$ with respect to $\mathbf{F}$.

$$
\begin{aligned}
&: A \sqsubseteq C \\
&: A \sqsubseteq \forall r.C \\
&: A \sqsubseteq \forall r.\forall r.C \\
&: C \sqsubseteq \forall r.\forall r.\forall r.? \\
\end{aligned}
$$

Example 4.6.1 shows that the elimination of cyclic definers is a non-trivial problem because entailment checking is EXPTIME-complete [SSS91, Sat07]. Example 4.6.2 suggests that the problem is even harder as it requires finding the maximum depth of the tree models rooted at domain elements in the interpretation of A whose edges are labelled with the role name r. Here, by *depth* we mean the maximum length of a path in a tree model. For instance, the maximum depth of the models rooted at domain elements in the interpretations of A in Example 4.6.2 is three. This depth was imposed by the clauses $: A \sqsubseteq C; : C \sqsubseteq \forall r.\forall r.\forall r.?$. As such, the tree model $\mathcal{I}$ in Figure 4.9, with depth three, is a model of the deductive view (and the original ontology) whereas $\mathcal{J}$, whose depth is four, is not.

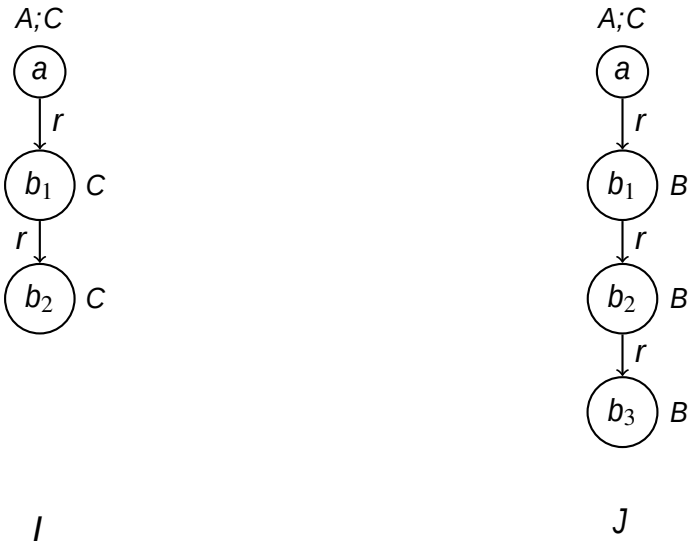

Figure 4.9: Model $\mathcal{I}$ on the left is a tree model with depth three. Model $\mathcal{J}$ on the right is a tree model with depth four

The worst case time of deciding the existence of a finite deductive forgetting view is double exponential in the size of the input ontology in [LW11]. Therefore, it was not suitable to integrate such a process in our forgetting framework. But, if we allow V^d in some cases to use cyclic definers, *how does it relate to the deductive forgetting views that do not use definers, if any exists?* For the above two examples, we can show that V^d and the suggested deductive forgetting views are model inseparable with respect to the non-forgetting signature.

Example 4.6.3. *Recall the forgetting view $V^d = \{\neg D \sqsubseteq C; \neg D \sqsubseteq \exists r.D; \neg C\}$ computed by our method in Example 4.6.1. Observe that the deductive forgetting view $\{\top \sqsubseteq \neg C\}$ can be obtain from V^d by:*

1. *Writing the first and second clauses as $\neg D \sqsubseteq (C \sqcap \exists r.D)$.*
2. *Resolving with the third clause to give $\neg D \sqsubseteq (\bot \sqcap \exists r.D)$, which simplifies to $\neg D$.*
3. *Applying the negative definer purification rule to eliminate D, which shall remove the clause $\neg D$, and return the ontology containing the third clause $\neg C$.*

As all the operations performed to obtain the deductive forgetting view $\{\neg C\}$ are model preserving, we have $V^d \underset{\{C\}}{\equiv_M} \{\top \sqsubseteq \neg C\}$.

Example 4.6.4. *Recall the forgetting view $V^d = \{\neg A \sqsubseteq C; \neg A \sqsubseteq \forall r.D; \neg D \sqsubseteq C; \neg D \sqsubseteq \forall r.D; \neg C \sqsubseteq \forall r.\forall r.\forall r.\bot\}$ computed by our method in Example 4.6.2.*

We obtain the deductive forgetting view $UI = \{\neg A \sqsubseteq C; \neg A \sqsubseteq \forall r.C; \neg A \sqsubseteq \forall r.\forall r.C; \neg C \sqsubseteq \forall r.\forall r.\forall r.\bot\}$ from V^d by:

1. *Resolving the first and the fifth clauses of V^d to obtain $\neg A \sqsubseteq \forall r.\forall r.\forall r.\bot$.*
2. *Writing the third and fourth clauses as $\neg D \sqsubseteq (C \sqcap \forall r.D)$.*
3. *Resolving the second clause with $\neg D \sqsubseteq (C \sqcap \forall r.D)$ to obtain the following finite sequence of clauses:*

$$\neg A \sqsubseteq \forall r.(C \sqcap \forall r.D)$$
$$\neg A \sqsubseteq \forall r.(C \sqcap \forall r.(C \sqcap \forall r.D))$$

We may ignore further resolutions since we have from 1 that $\neg A \sqsubseteq \forall r.\forall r.\forall r.\bot$.

4. *Remove the second clause* $: A \sqsubseteq \forall r.D$ *and* $: D \sqsubseteq (C \sqcup \forall r.D)$ *by purity deletion.*

5. *Eliminate D using the positive definer purification rule.*

All the operations performed to obtain the deductive forgetting view preserve the interpretations of A;C, and r. Hence, we have $\mathcal{V}^d \underset{\{A,C,r\}}{\equiv_M} \mathcal{UI}$.

Let us consider the general case. The aim is to compare the models of $\mathcal{V}^d$ to the models of all deductive forgetting views in the general case. The following lemma implies that it is sufficient to compare $\mathcal{V}^d$ with just one deductive forgetting view.

Lemma 4.6.5. *Let* $\mathcal{O}$ *be an ontology and* $\mathbf{F}$ *be a forgetting signature. Suppose* $\mathcal{UI}_1$ *and* $\mathcal{UI}_2$ *are two deductive forgetting views of* $\mathcal{O}$ *with respect to* $\mathbf{F}$ *such that* $sig(\mathcal{UI}_1) \subseteq sig(\mathcal{UI}_2) \cup sig(\mathcal{O}) \cap \mathbf{F}$. *Then,* $\mathcal{UI}_1$ *and* $\mathcal{UI}_2$ *are logically equivalent.*

Proof. Let α be an arbitrary axiom in $\mathcal{UI}_1$. We have the following.

1. $\mathcal{UI}_1 \models \alpha$ because α is an axiom of $\mathcal{UI}_1$.

2. $\mathcal{O} \models \alpha$ because $sig(\alpha) \subseteq sig(\mathcal{O}) \cap \mathbf{F}$ and $\mathcal{O} \underset{sig(\mathcal{O}) \cap \mathbf{F}}{\equiv_C} \mathcal{UI}_1$.

3. $\mathcal{UI}_2 \models \alpha$ because $\mathcal{O} \underset{sig(\mathcal{O}) \cap \mathbf{F}}{\equiv_C} \mathcal{UI}_2$.

Since α is arbitrary, we get that $\mathcal{UI}_2 \models \mathcal{UI}_1$. In the same way we see that $\mathcal{UI}_1 \models \mathcal{UI}_2$. So, we have that $\mathcal{UI}_1$ and $\mathcal{UI}_2$ are logically equivalent. □

Let $\mathcal{UI}$ be a deductive forgetting view of our input ontology $\mathcal{O}$ with respect to the forgetting signature $\mathbf{F}$. We split our previous question in the following two questions.

1. Are the models of $\mathcal{V}^d$ also models of $\mathcal{UI}$?

2. Are the models of $\mathcal{UI}$ also models of $\mathcal{V}^d$?

With a similar argument to that used in the proof of Lemma 4.6.5 we can show that the answer to the first question is *yes*.

Lemma 4.6.6. *Let* $\mathcal{O}$ *be an ontology and* $\mathbf{F}$ *a forgetting signature. Suppose* $\mathcal{V}^d$ *is a view generated by our method such that* $\mathcal{V}^d$ *uses cyclic definers, and suppose* $\mathcal{UI}$ *a deductive forgetting view of* $\mathcal{O}$ *with respect to* $\mathbf{F}$ *where* $sig(\mathcal{UI}) \subseteq sig(\mathcal{O}) \cap \mathbf{F}$. *We have* $\mathcal{V}^d \models \mathcal{UI}$.

Proof. From Theorem 4.5.5, we have $\mathcal{O} \overset{C}{\underset{sig(\mathcal{O})nF}{\equiv}} \mathcal{V}^d$. Also, since $\mathcal{UI}$ is a deductive forgetting view of $\mathcal{O}$ with respect to F, we have $\mathcal{O} \overset{C}{\underset{sig(\mathcal{O})nF}{\equiv}} \mathcal{UI}$. By transitivity of $\overset{C}{\equiv}$, we have $\mathcal{V}^d \overset{C}{\underset{sig(\mathcal{O})nF}{\equiv}} \mathcal{UI}$. Let α be an arbitrary axiom in $\mathcal{UI}$. We have $\mathcal{UI} \models \alpha$ which implies $\mathcal{V}^d \models \alpha$. Since α is arbitrary, we have $\mathcal{V}^d \models \mathcal{UI}$. That is, every model of $\mathcal{V}^d$ is also a model of $\mathcal{UI}$. □

The answer to second question about whether the models of $\mathcal{UI}$ are also models of $\mathcal{V}^d$ is *no* because the models of $\mathcal{UI}$ need not have interpretations for the definers in $\mathcal{V}^d$. We conjecture though that $\mathcal{V}^d$ and $\mathcal{UI}$ are model inseparable with respect to the non-forgetting signature $sig(\mathcal{O})nF$. We construct series of ontologies from $\mathcal{V}^d$ over $sig(\mathcal{O})nF$ such that every ontology in the series is stronger than the ontology that comes before it in the sense that the models of the stronger ontology are also models of the weaker ontology but not vice versa. The limit of this series is $\mathcal{V}^d$. Next, we show that $\mathcal{UI}$ entails every ontology in the series. Because $\mathcal{V}^d$ is the limit to this series, we argue that the models of $\mathcal{UI}$ and the models of $\mathcal{V}^d$ coincide on the interpretations the symbols from $sig(\mathcal{O})nF$. If this series is finite, then we show that $\mathcal{V}^d$ and the last ontology in the series are model inseparable with respect to $sig(\mathcal{O})nF$.

Assume $\mathcal{V}^d$ uses a cyclic definer D. We group the clauses and axioms of $\mathcal{V}^d$ in the following four sets.

- $\mathcal{V}^B$: The clauses and axioms where D does not occur.
- $\mathcal{V}^+$: Clauses where D occurs only positively below role restrictions.
- $\mathcal{V}^-$: Clauses of the form $\neg D \sqcup C$ where C is an $\mathcal{ALC}$ concept that does not contain D.
- $\mathcal{V}^*$: Clauses of the form $\neg D \sqcup E$ where E use the definer D positively below role restrictions.

We excluded from the above groups the clauses that use D and another definer as a top level negative disjuncts. Clauses of this form are part of the set D^d which has been removed in the deductive reduction stage (see Section 4.4). The clauses in $\mathcal{V}^-$ allow for inferring infinite number of clauses by resolving them with the clauses of $\mathcal{V}^+$. If $\mathcal{V}^-$ is empty, then the definer D is not cyclic and can be eliminated using the definer elimination process presented in Section 4.5.

Let $\mathcal{V}$ denote the union of $\mathcal{V}^B; \mathcal{V}^+$, and $\mathcal{V}^*$. So we have $\mathcal{V} \cup \mathcal{V}^* \equiv \mathcal{V}^d$. Merge $\mathcal{V}$ and $\mathcal{V}^-$ in one set of clauses. Assume this set contains n clauses $\neg D \sqcup F_i$ where n is

a natural number greater than or equal to one, $1 \quad i \quad n$, and F_i's are concepts that take the form of C in the definition of $V \quad$ or E in the definition of $V \quad$. Combine the clauses of the merged set in one clause $C = :D \; t \; F$ where F is the conjunction of all F_i's. We compute an ontology V_i^d at position i in the series using the following steps.

1. if $i > 1$, perform $i \quad 1$ resolution rounds, where in the first round we resolve the clauses of V^+ with C, and in any round j with $2 \quad j \quad i \quad 1$ we resolve C with the resolvents obtained from the $j \quad 1$ round. Denote by V_i the union of V and the resolvents obtained from all $i \quad 1$ resolution rounds. If $i = 1$, set V_1 as V.

2. Eliminate D from V_i using the definer elimination process presented in Section 4.5.

The ontologies of the series are computed in order, V_1^d then V_2^d and so on. We stop when V_i^d and V_{i+1}^d are logically equivalent and in this case the series will contain just the ontologies $V_1^d; V_2^d; :::;$ and V_i^d. The ontologies in the computed series do not contain the cyclic definer D but if V^d has other cyclic definers then they will also occur in V_i^d. If a cyclic definer D^0 occurs in the computed ontologies, we take every one of them and compute from it another series of ontologies that do not use D^0. We repeat this process until we obtain several series of ontologies where no cyclic definers occur. The following lemmas and discussions are invariant to any obtained series of ontologies. For simplicity we assume there is one series of ontologies where a definer D is eliminated. The first lemma says that the ontologies in a computed series is monotonic, that is, every ontology in the series entails the ontologies that come before it.

Lemma 4.6.7. *Let V_i^d and V_j^d be two ontologies in a computed series such that i and j are natural numbers greater than or equal to one, and $j > i$. We have $V_j^d \models V_i^d$.*

Proof. The only difference between V_j^d and V_i^d is that more resolution rounds were performed in Step 1 when V_j^d was computed than those performed when V_i^d computed because $j > i$. Let V_i be the output of Step 1 when V_i^d was computed and V_j be the output of the same step when V_j^d was computed. We have $V_i \quad V_j$.

Now, V_i^d is obtained from V_i by eliminating a definer symbol using the process presented in Section 4.5, and V_j^d is obtained from V_j by the same process. Since $V_i \quad V_j$, all definer elimination inferences performed in V_i are also performed in V_j. Furthermore, the same output of the definer elimination inferences performed in V_i will be obtained as part of the output of the definer elimination inferences performed in V_j. In other words, we have that $V_i^d \quad V_j^d$. Therefore, $V_j^d \models V_i^d$. □

The next lemma says that $\mathcal{V}^d$ entails all the computed ontologies in all series.

Lemma 4.6.8. *For every computed ontology $\mathcal{V}_i^d$ we have $\mathcal{V}^d \models \mathcal{V}_i^d$.*

Proof. We have the following:

1. Step 1 in generating $\mathcal{V}_i^d$ resolves the clause C with $\mathcal{V}^+$ and then returns the union of $\mathcal{V}$ and the computed resolvents. By construction we have $\mathcal{V}^d \models \mathcal{V}^+$ because $\mathcal{V}^+$ is a subset of $\mathcal{V}^d$, and $\mathcal{V}^d \models C$ because $\{C, \quad \mathcal{V} \quad [\mathcal{V} \quad$ and $\mathcal{V} \quad [\mathcal{V} \quad \quad \mathcal{V}^d$. So the resolvent of C and $\mathcal{V}^+$ are entailed by $\mathcal{V}^d$. Also, by construction we have $\mathcal{V}^d \models \mathcal{V}$, so $\mathcal{V}^d$ entails the union of $\mathcal{V}$ and the computed resolvents, that is, $\mathcal{V}^d \models \mathcal{V}_i$.

2. Step 2 eliminates D from $\mathcal{V}_i$ using the definer elimination process of Section 4.5. From the proof of Theorem 4.5.5, we find that $\mathcal{V}_i$ and $\mathcal{V}_i^d$ are model inseparable modulo the eliminated definer, so $\mathcal{V}_i \models \mathcal{V}_i^d$.

3. From the two points above and the transitivity of $\models$ we find that $\mathcal{V}^d \models \mathcal{V}_i^d$.

□

Recall that the computed ontologies are obtained by (1) resolving the clause C, which preserves all information of $\mathcal{V} \quad$ and $\mathcal{V} \quad$, with the clauses of $\mathcal{V}^+$, (2) adding the obtained resolvents to $\mathcal{V}^B \; [\; \mathcal{V}^+ \; [\; \mathcal{V} \quad$, and (3) eliminating the cyclic definer using the definer elimination process which preserves model inseparability modulo the eliminated definer. Therefore, the set $\mathcal{V} \quad$ resembles the difference between a computed ontology $\mathcal{V}_i^d$ and $\mathcal{V}^d$. When $\mathcal{V}_{i+1}^d$ is computed, the set $\mathcal{V} \quad$ is restored back through C to obtain new clauses in $\mathcal{V}_{i+1}^d$ that were not in $\mathcal{V}_i^d$. Therefore, we see $\mathcal{V}^d$ as the limit of the computed series in the sense that as i goes to infinity, the models of $\mathcal{V}_i^d$ and the models of $\mathcal{V}^d$ coincide on the interpretations of the non-forgetting symbols. The main point here is that all axioms of $\mathcal{V}^d$ are used in computing the ontologies of the series. For instance, we can compute the same series from the intermediate ontology $\mathcal{V}^d$, but we cannot view $\mathcal{V}^d$ as the limit of this series because it has the D^d clauses which may contain cyclic definers but are not used in the computation process.

The following lemma shows that models of $\mathcal{UI}$ are also models of every ontology in the series.

Lemma 4.6.9. *For every computed ontology we have $\mathcal{UI} \models \mathcal{V}_i^d$.*

Proof. Since V_i^d does not use definers, we have that $sig(V_i^d) \quad sig(O)nF$. From Lemma 4.6.8 we have $V^d \models V_i^d$. Let α be an arbitrary axiom in V_i^d. Since V^d and UI are deductively inseparable with respect to $sig(O)nF$, we have that $UI \models \alpha$. Therefore, $UI \models V_i^d$. □

In the following lemma we consider the last ontology of all series. When there are several series computed, we start at the last ontology of the first series, and move from it to the last ontology of the second series, and so on. The very last ontology that we reach is the last ontology of all series.

Lemma 4.6.10. *Suppose the computed series are finite, and let V_n^d be the last ontology of all series. Then the following is true.*

1. *V^d and V_n^d are model inseparable with respect to $sig(O)nF$.*
2. *V_n^d is a deductive forgetting view of O with respect to F.*

Proof. Let us prove Item 1, then Item 2 will follow from Item 1 and the fact that V_n^d does not contain definers, that is, $sig(V_n^d) \quad sig(O)nF$.

By construction we stop computing new ontologies in a series when the ontology being computed is logically equivalent to the last computed ontology. That is, the computed series is finite if V_n^d and V_{n+1}^d are logically equivalent. Let V_n be the output of Step 1 when V_n^d is being computed, and V_{n+1} be the output of the same step when V_{n+1}^d is being computed. Since V_n^d and V_{n+1}^d are obtained from V_n and V_{n+1} respectively by the definer elimination process which preserves model inseparability modulo definers, we have that V_n and V_{n+1} are model inseparable with respect to $sig(O)nF$. The differences between V_{n+1} and V_n are the resolvents obtained by the last resolution round performed while computing V_{n+1}. Suppose the last resolution rounds when computing V_n has m resolvents $C_j^n \vee E_j^n$ with with $1 \quad j \quad m$ where the cyclic definer D occurs in E_j^n below role restrictions. The last resolution round when computing V_{n+1} resolves the clause $C = \neg D \vee F$ obtained from $V \quad$ and $V \quad$ with the m resolvents $C_j^n \vee E_j^n$. The outcome of this round is a set of m clauses $C_j^{n+1} \vee E_j^{n+1}$ where the cyclic definer D occurs in E_j^{n+1} below role restrictions. But since V_n and V_{n+1} are model inseparable with respect to $sig(O)nF$, we see that these resolvents are not providing any information on the symbols in $sig(O)nF$. This in turn implies that the concept C is not providing any new information on the symbols in $sig(O)nF$. So, we have $V_n \cup \{C\} \underset{sig(O)nF}{\equiv} V_n$. But by construction we have $V_n \cup \{C\} \quad V^d$. So, $V^d \underset{sig(O)nF}{\equiv} V_n$, and by Lemma 3.5.9 we have $V^d \underset{sig(O)nF}{\equiv} V_n^d$. □

Lemma 4.6.11. *Assume the computed series are finite, then V^d and UI are model inseparable with respect to $sig(O) \setminus F$.*

Proof. Let V_n^d be the last ontology of all series. We have the following.

1. From Lemma 4.6.10, V^d and V_n^d are model inseparable with respect to $sig(O) \setminus F$
2. Also from the same Lemma, V_n^d is a deductive forgetting view of O with respect to F.
3. From Lemma 4.6.5 and the above point, $UI \quad V_n^d$.
4. From the above point and Point 1, we get that UI and V^d are model inseparable with respect to $sig(O) \setminus F$.

□

4.7 Comparison with Existing Semantic Forgetting Methods

Lemma 4.3.3 showed that O^{int} is model inseparable from the input ontology O with respect to the non-forgotten vocabulary $sig(O) \setminus F$, but it may contain definers introduced in the outset of our method when O is transformed to $O^{clausal}$. We observe that the use of definers in O^{int} is not limiting for standard reasoning tasks. The following example highlights the observation.

Example 4.7.1. *Let O be the following ontology.*

$$A_1 \sqsubseteq \forall r.B$$
$$A_2 \sqsubseteq \forall r.\neg B$$

and $F = \{B\}$ a forgetting signature. Using the method described in the previous section, we have O^{int} as the following ontology

$$\neg A_1 \sqcup \forall r.D_1$$
$$\neg A_2 \sqcup \forall r.D_2$$
$$\neg D_1 \sqcup \neg D_2$$

where D_1 and D_2 are helper symbols. Both $\mathcal{O}$ and $\mathcal{O}^{int}$ model the information that the r-successors of the elements in the interpretation of A_1 are disjoint from the r-successors of the elements in the interpretation of A_2.

Suppose in a query answering application we have the following ABox $\mathcal{A}$

$$A_1(a_1)$$
$$A_2(a_2)$$
$$r(a_1, b)$$

where a_1, a_2, b are individuals. Suppose also that we want to prove the unsatisfiability of $r(a_2, b)$ with respect to the knowledge base $(\mathcal{O}, \mathcal{A})$. That is, we want to answer the boolean query $(\mathcal{O}, \mathcal{A}) \models \neg r(a_2, b)$. This can be done using simple forward reasoning to show that $\mathcal{O}, \mathcal{A}, r(a_2, b) \models ?$. Replacing $\mathcal{O}$ with $\mathcal{O}^{int}$, would still prove the unsatisfiability of $r(a_2, b)$ with the same forward reasoning:

1. *$\neg A_1 \sqcup \forall r.D_1$ and $A_1(a_1)$ imply $\forall r.D_1(a_1)$.*
2. *$\forall r.D_1(a_1)$ and $r(a_1, b)$ imply $D_1(b)$.*
3. *$\neg A_2 \sqcup \forall r.D_2$ and $A_2(a_2)$ imply $\forall r.D_2(a_2)$.*
4. *$\forall r.D_2(a_2)$ and $r(a_2, b)$ imply $D_2(b)$.*
5. *$\neg D_1 \sqcup \neg D_2$ and $D_1(b)$ imply $\neg D_2(b)$.*
6. *$\neg D_2(b)$ and $D_2(b)$ imply a contradiction.*

In the above reasoning, the use of the definer symbols D_1 and D_2 in $\mathcal{O}^{int}$ was not limiting to the query answering method.

The observation made in Example 4.7.1 is not surprising because definers are used to denote subsets of role successors. That is, they are existentially quantified symbols. Existing methods performing standard reasoning tasks are designed to operate on ontologies whose concept and role names can also be viewed as existentially quantified. From this perspective, definers are not different from non-definers, that is, the symbols $sig(\mathcal{O}) \cap \mathbf{F}$ that were used in the input ontology $\mathcal{O}$.

An important difference between the non-definers and the definers used in $\mathcal{O}^{int}$ is that the interpretations of the non-definer symbols are usually known to the users. However, definers represent subsets of role successors whose precise definition cannot

always be computed. In this sense, definers can also be seen as second-order existentially quantified symbols.

Example 4.7.2. *Consider the following ontology $\mathcal{O}$.*

$$Supervisor \sqsubseteq \exists supervise.PhDStudent \sqcap \exists supervise.MasterStudent$$
$$PhDStudent \sqcap MasterStudent \sqsubseteq \bot$$

Let $\mathcal{I}$ be a model of $\mathcal{O}$ with $\Delta^{\mathcal{I}} = \{a,b,c,d\}$, and:

$$Supervisor^{\mathcal{I}} = \{a\}$$
$$PhDStudent^{\mathcal{I}} = \{b\}$$
$$MasterStudent^{\mathcal{I}} = \{c,d\}$$
$$Supervise^{\mathcal{I}} = \{(a,b),(a,c),(a,d)\}$$

Let $\mathbf{F} = \{PhDStudent, MasterStudent\}$ is a forgetting signature. We have $\mathcal{O}^{int}$ as the following ontology.

$$\neg Supervisor \sqcup \exists supervise.D_1$$
$$\neg Supervisor \sqcup \exists supervise.D_2$$
$$\neg D_1 \sqcup \neg D_2$$

where D_1 and D_2 are definers. Suppose we want to extend $\mathcal{I}$ with the interpretations of D_1 and D_2. Some of the possible interpretations are:

1. $D_1^{\mathcal{I}} = \{b\}$, and $D_2^{\mathcal{I}} = \{c\}$,
2. $D_1^{\mathcal{I}} = \{b\}$, and $D_2^{\mathcal{I}} = \{d\}$,
3. $D_1^{\mathcal{I}} = \{b,c\}$, and $D_2^{\mathcal{I}} = \{d\}$,
4. $D_1^{\mathcal{I}} = \{b,d\}$, and $D_2^{\mathcal{I}} = \{c\}$,

The definers D_1 and D_2 denote arbitrary disjoint subsets of the successors of the role 'supervise'.

Seeing definers as second-order existentially quantified, we may write the intermediate ontology as a second-order formula $\exists D_1.\exists D_2.\ldots.\exists D_n.\mathcal{O}_{fo}$ where $\mathcal{O}_{fo}$ is the first-order translation of $\mathcal{O}^{int}$, and $D_1,\ldots,D_n$ are the definers occurring in $\mathcal{O}^{int}$. The negation

of $\mathcal{O}^{int}$ can be written as the second-order formula $\forall D_1 . \forall D_2 . \ldots \forall D_n . \mathcal{O}_{fo}$. Reasoning methods which require negating $\mathcal{O}^{int}$ would need new inference systems in order to reason over the now universally quantified definers. This however is not observed with non-definers which are first-order.

We can conclude from the above that although $\mathcal{O}^{int}$ does not follow Definition 3.5.16, it can in theory be used in reasoning tasks to allow for inferring information over the non-forgetting signature, provided that it is not negated. Our empirical evaluation on real data showed that the number of definers introduced in $\mathcal{O}^{int}$ was significantly smaller than the number of forgotten symbols, see Chapter 7 for more details. Therefore, $\mathcal{O}^{int}$ can provide optimizations to automated reasoning applications, being a substitute for large ontologies that allows efficient inference of the information.

The use of second-order symbols was investigated the literature. Frangzhen Lin and Ray Reiter proposed in [LR94a] a semantic forgetting method for first-order theories which uses second-order existentially quantified symbols in the semantic forgetting view.

We can infer a common outline for our method of generating $\mathcal{O}^{int}$ and the forgetting method of [LR94a]. This outline is:

1. Normalization

2. Forgetting

3. Definer elimination

Normalization is performed in [LR94a] by transformation to NNF and Skolemization, whereas it is performed in our method by writing the input ontology in the clausal form of $\mathcal{O}^{clausal}$. The Normalization step of our method may introduce definers.

The Forgetting step is performed in [LR94a] by replacing the symbols from $\mathcal{F}$ by fresh existentially quantified predicate names which are indifferent to definers. Forgetting is performed in our method using unrestricted resolution.

The Definer Elimination step is left open in the method of [LR94a], and the authors suggest the method designed in [GO92] to be used. In Section 4.5, we describe a method for eliminating some of the definers from $\mathcal{O}^{int}$.

A difference between our method of generating $\mathcal{O}^{int}$ and [LR94a] is the interpretation of the definers. [LR94a] uses definers as synonyms of the forgetting symbols, whereas our method introduces them to denote subsets of role successors.

Example 4.7.3. *Let $\mathcal{O}$ be an ontology consisting of the following axiom*

$$A \sqsubseteq \exists r{:}B \sqcup \exists r{:}\neg B$$

and $\mathbf{F} = \{B\}$ a forgetting signature.

The ontology $\mathcal{V} = \{\neg A \sqcup (\exists r{:}D \sqcup \exists r{:}\neg D)\}$ is a DL-translation of the output of [LR94a], where D is a definer. Our method would compute $\mathcal{O}^{int}$ as:

$$\neg A \sqcup \exists r{:}D_1$$
$$\neg A \sqcup \exists r{:}D_2$$
$$\neg D_1 \sqcup \neg D_2$$

where D_1 and D_2 are definers.

Here, our method identifies two disjoint subsets of role successors D_1 and D_2, whereas the output of [LR94a] renames the symbol B as the new definer D.

Using definers to represent subsets of role successors adds to the granularity of $\mathcal{O}^{int}$ and allows for extracting more precise forgetting views. For instance, it was possible in the above example to extract the subset $\mathcal{O}^d$ which is model inseparable from the deductive forgetting view $\mathcal{V}^d$. It is not possible to extract $\mathcal{O}^d$ in such a direct way if definers are used to obfuscate the forgetting symbols as in [LR94a].

4.8 Computing More Informative Forgetting Views

We can design a process to enhance the informativeness of the final generated view according to the user's requirements. Following this process reveals a spectrum of views that are more informative than the deductive forgetting view and at most as informative as the semantic forgetting view. The process works simply by overriding the *Reduction* rule and allow clauses from D^d to be present in $\mathcal{O}^d$. We illustrate the idea through the following example.

Example 4.8.1. *Consider $\mathcal{O}^{int}$ from Example 4.3.1. Applying the rules in Figure 4.5 gives $\mathcal{O}^d$ and D^d from Example 4.4.5.*

We may increase the informativeness of the final forgetting view by overriding the Reduction *rule. Suppose we want to preserve all the information about the r-successors of A, then we override the* Reduction *rule to retain the clauses where D_1*

occurs. In this case, $\mathsf{D}^d = \emptyset$*, and* $\mathcal{O}^d$ *is the following ontology.*

$$\begin{aligned}
&\neg A \sqsubseteq \forall r.D_1\\
&\neg A \sqsubseteq \forall s.D_2\\
&\neg G \sqsubseteq \exists r.D_3\\
&\neg D_1 \sqsubseteq \neg D_2\\
&\neg D_1 \sqsubseteq \neg D_3 \sqcup C\\
&\neg A \sqsubseteq \neg G \sqcup \exists r.C
\end{aligned}$$

As the rules in Figure 4.8 are not applicable to $\mathcal{O}^d$*, we have the final forgetting view* $\mathcal{V}^d = \mathcal{O}^d$*.*

In the above example we preserved all information related to the r-successors of domain elements in the interpretation of A. The following example shows that more fine-grained customization of the preserved information can be performed.

Example 4.8.2. *Continuing with Example 4.8.1, suppose we are interested in the relation between the r and s successors of A. We override the* Reduction *rule to remove* $\neg D_1 \sqsubseteq \neg D_3 \sqcup C$ *but not* $\neg D_1 \sqsubseteq \neg D_2$*. That is* $\mathsf{D}^d = \{\neg D_1 \sqsubseteq \neg D_3 \sqcup C\}$*. Consequently, we get* $\mathcal{O}^d$ *as the following ontology.*

$$\begin{aligned}
&\neg A \sqsubseteq \forall r.D_1\\
&\neg A \sqsubseteq \forall s.D_2\\
&\neg G \sqsubseteq \exists r.D_3\\
&\neg A \sqsubseteq \neg G \sqcup \exists r.C\\
&\neg D_1 \sqsubseteq \neg D_2
\end{aligned}$$

The definer D_3 *can be eliminated by applying the Positive Definer Purification rule in Figure 4.8 to obtain the final forgetting view* $\mathcal{V}^d$ *as:*

$$\begin{aligned}
&\neg A \sqsubseteq \forall r.D_1\\
&\neg A \sqsubseteq \forall s.D_2\\
&\neg G \sqsubseteq \exists r.\top\\
&\neg A \sqsubseteq \neg G \sqcup \exists r.C\\
&\neg D_1 \sqsubseteq \neg D_2
\end{aligned}$$

The following example shows a different customization of the preserved information where it is desired to preserve the relation between role successors of different subsets of domain element.

Example 4.8.3. *Continuing with Examples 4.8.1 and 4.8.2. Suppose we want to preserve the information about the r-successors of A in relation to the r-successors of G. We override the* Reduction *rule to retain the clauses where both D_1 and D_3 occur. That is,* $\mathsf{D}^d = \{ : D_1 \sqcup : D_2, : D_1 \sqcup : D_3 \sqcup C \mid \mathcal{O} \in \mathsf{D}^d$*. The following ontology is $\mathcal{O}^d$.*

$$: A \sqcup \forall r.D_1$$
$$: A \sqcup \forall s.D_2$$
$$: G \sqcup \exists r.D_3$$
$$: D_1 \sqcup : D_3 \sqcup C$$
$$: A \sqcup : G \sqcup \exists r.C$$

The definer D_2 can be eliminated by applying the Positive Definer Purification rule in Figure 4.8 to obtain the final forgetting view $\mathcal{V}^d$ as:

$$: A \sqcup \forall r.D_1$$
$$: G \sqcup \exists r.D_3$$
$$: D_1 \sqcup : D_3 \sqcup C$$
$$: A \sqcup : G \sqcup \exists r.C$$

In Examples 4.8.1 and 4.8.3 the clause $: A \sqcup : G \sqcup \exists r.C$ is redundant with respect to the subset $\{ : A \sqcup \forall r.D_1, : G \sqcup \exists r.D_3, : D_1 \sqcup : D_3 \sqcup C\}$ of $\mathcal{V}^d$. Recall that the clause $: A \sqcup : G \sqcup \exists r.C$ was generated by the Role Propagation rule in Figure 4.3. The elimination of this redundancy requires a modification to *Deductive Reduction* component of our framework. Instead of performing the *Role Propagation* rule then removing the set D^d from $\mathcal{O}^{int}$ to generate $\mathcal{O}^d$, the set D^d would be identified in $\mathcal{O}^{int}$ in a first step. In a second step, the subset of D^d that would be preserved is extracted, and used to control the application of the *Role Propagation* rule in a third step. When all Role Propagation inferences have been performed, the remaining D^d clauses are removed from $\mathcal{O}^{int}$ in a fourth and final step.

Algorithm 3 shows an outline of our fine-grained forgetting method. Transformation to clausal form and computation of $\mathcal{O}^{int}$ are performed in lines 1 and 2 as described in Section 4.3. Line 3 identifies D^d but does not exclude it from $\mathcal{O}^{int}$. Line 4 runs an

Algorithm 3: Updated algorithm of fine-grained forgetting

Input: Ontology $\mathcal{O}$, forgetting signature $\mathbf{F}$
Output: Fine-grained forgetting view $\mathcal{V}^d$
// Initialized all used sets of clauses to the empty set
Initialize: $\mathcal{O}^{clausal}; \mathbf{D}^d; \mathbf{D}^p; \mathcal{O}^{int}; \mathcal{O}^d; \mathcal{V}^d := \emptyset$
1 $\mathcal{O}^{clausal}$:= TransformToClausalForm($\mathcal{O}; \mathbf{F}$)
2 $\mathcal{O}^{int}$:= EliminateSymbolsByResolution($\mathcal{O}^{clausal}; \mathbf{F}$)
// Identify the set $\mathbf{D}^d$ of clauses that use two or more negative definers
3 $\mathbf{D}^d$:= IdentifyDelta($\mathcal{O}^{int}$)
// Run an external process to find the clauses of $\mathbf{D}^d$ preserved in the fine-grained forgetting view
4 $\mathbf{D}^p$:= FindPreservedClausesFromDelta($\mathcal{O}^{int}; \mathbf{D}^d$)
5 $\mathcal{O}^d$:= (PerformRolePropagation($\mathcal{O}^{int}; \mathbf{D}^p$)$n\mathbf{D}^d$) $[$ $\mathbf{D}^p$
6 $\mathcal{V}^d$:= EliminateDefiners($\mathcal{O}^d$)
Return: $\mathcal{V}^d$

external process that returns a set $\mathbf{D}^p$ of clauses that will be preserved in the final forgetting view. Line 5 performs role propagation and removes the non-preserved clauses of $\mathbf{D}^d$. The method *PerformRolePropagation* takes an input $\mathbf{D}^p$ which will be used to control the execution of the Role Propagation rule. Line 6 eliminates the definer symbols from $\mathcal{O}^d$ to obtain a final forgetting view $\mathcal{V}^d$.

The method PerformRolePropagation used in Algorithm 3 is outlined in Algorithm 4. Lines 2 and 3 loops on clauses from $\mathbf{D}^d$ and use them as a first premise to the Role Propagation rule if at most one negative definer from p_1 occurs positively in $\mathcal{O}^{int}$ under existential role restrictions. Lines 4, 5, and 6 collect the second, third, and fourth premises of the Role Propagation rule. Line 7 prevents the application of the Role Propagation rule when all of its premises will be preserved in $\mathcal{O}^{int}$. Line 8 performs role propagation as described in Section 4.3.1.

Algorithm 4: Outline of method PerformRolePropagation

Input: Ontology O^{int}, a set of clauses D^p
Output: An ontology which extends O^{int} with conclusions of the Role Propagation rule
Initialize: $D^d; p_2; p_4 := \emptyset; p_1; p_3; C_{rp} := NULL$

```
1 D^d := IdentifyDelta(O^int)
2 for clause p1 2 D^d do
      // At most one -ve definer from p1 occurs positively in O^int under
         existential role restriction
3     if validPremise(p1) then
          // Get the clauses representing the second, third, and fourth
             premises of the Role Propagation rule
4         p2 := GetPremise2(O^int; p1)
5         p3 := GetPremise3(O^int; p1)
6         p4 := GetPremise4(O^int; p1)
          // Perform role propagation if any of the D^d clauses used in the
             premises is not preserved
7         if (p2 [ fp1g) \ D^d 6 D^p then
              // Perform the Role Propagation rule and capture the conclusion
                 in the clause C_rp
8             C_rp := computeConclusionOfRolePropagation(p1; p2; p3; p4)
9             O^int := O^int [ fC_rpg
```

Return: O^{int}

Chapter 5

Query Forgetting

In the last chapter we established a fine-grained forgetting framework. The inputs to the framework are an $\mathcal{ALC}$ ontology and a forgetting signature consisting of concept names. The framework computes a representation of the semantic forgetting views of the input ontology with respect to the forgetting signature, and expresses it in a normal form that allows for extracting a spectrum of fine-grained forgetting views with customized content. Key to this is the customization described in Sections 4.4 and 4.5, whereby the final forgetting view coincides with the deductive forgetting views of $\mathcal{O}$ with respect to $\mathbf{F}$.

In this chapter we develop another customization, namely the query forgetting customization. One output of this customization is a representation of the query forgetting view of the input ontology with respect to the forgetting signature. Recall that an ontology $\mathcal{V}^q$ is a query forgetting view of an ontology $\mathcal{O}$ with respect to a forgetting signature $\mathbf{F}$, if $\mathcal{V}^q$ and $\mathcal{O}$ coincide on all answers to all conjunctive queries over $sig(\mathcal{O}) \cap \mathbf{F}$ against an arbitrary ABox $\mathcal{A}$. A formal definition of query forgetting is given in Definition 3.5.15. The customization generates two representations of the forgetting view. The first version is an $\mathcal{ALC}$ ontology that may use definer symbols. The second version is an $\mathcal{ALCI}$ forgetting view in which non-cyclic definers are eliminated.

As discussed in Chapter 4, we only use cyclic definers for existence and practical reasons. When the computed forgetting view uses cyclic definers, it can be seen as providing a faithful representation of the query forgetting view of the input ontology with respect to the forgetting signature. Having an $\mathcal{ALC}$ representation of the query forgetting view satisfies users who only have access to $\mathcal{ALC}$ query answering systems, and is useful in applications requiring the knowledge to be stored in $\mathcal{ALC}$ format. Users concerned about the presence of definers may use the $\mathcal{ALCI}$ representation

instead. There are two additional competitive advantages of the $\mathcal{ALC}$ representation compared to the $\mathcal{ALCI}$ representation. They are:

1. It can be used as deductive forgetting view of the input ontology with respect to the forgetting signature. It is therefore helpful in demanding applications where classification as well as query answering are needed.

2. We compute a set of axioms that indicates the content difference between the input ontology and the $\mathcal{ALC}$ representation relative to the non-forgetting signature. The set can be used as described in Section 4.8 to obtain more informative forgetting views between the query forgetting view and the semantic forgetting view of the input ontology with respect to the forgetting signature.

The research aims addressed in this chapter are:

1. With the query forgetting customization, we present the first query forgetting method in the context of the description logic $\mathcal{ALC}$.

2. Our forgetting framework will be capable of computing representations of different forgetting views of $\mathcal{ALC}$ ontologies. These are the semantic forgetting views, the deductive forgetting views, the query forgetting views, as well as the fine-grained forgetting views in between. This asserts practically the extensibility and the completeness properties of our framework. The framework is extensible because different customization methods can be plugged in to obtain forgetting views with different contents, and it is complete because it can compute forgetting views in between the three main forgetting views.

3. Obtaining the axioms indicating the content difference between the representations of the semantic and query forgetting views will allow for comparing the content of query forgetting views and deductive forgetting views.

The chapter is structured as follows. Section 5.1 gives an overview of the query forgetting customization. Section 5.2 computes the $\mathcal{ALC}$ representation of the query forgetting views. For clarity, the underlying mathematics will be discussed separately in Section 5.3. Section 5.4 eliminates the definers present in the $\mathcal{ALC}$ representation and obtains the $\mathcal{ALCI}$ representation of the query forgetting views.

5.1 Outline

The query forgetting customization presented in this chapter is depicted by Figure 5.1. The input to the customization is the ontology $\mathcal{O}^{int}$ obtained by the process in Section 4.3. Recall that $\mathcal{O}^{int}$ is computed by transforming the input ontology to the normal form, then eliminating the forgetting symbols one after the other. A forgetting symbol is eliminated by performing unrestricted binary resolution on the clauses where the symbol occurs positively, and the clauses where it occurs negatively. When all possible resolution steps have been performed, the clauses where the symbol occurs are removed, and $\mathcal{O}^{int}$ is obtained. Lemma 4.3.3 shows that $\mathcal{O}^{int}$ and the input ontology $\mathcal{O}$ are model inseparable with respect to the non-forgotten symbols.

The first process in the customization is *query reduction*. Two outputs are expected from this customization. The first is the *query reduced* ontology $\mathcal{O}^q$, and the second is the ontology D^q. The two ontologies, $\mathcal{O}^q$ and D^q, play parallel roles to $\mathcal{O}^d$ and D^d from Chapter 4. The query reduced ontology $\mathcal{O}^q$ is query inseparable to the input ontology $\mathcal{O}$ with respect to the non-forgotten symbols $sig(\mathcal{O})nF$, and D^q indicates the content difference between $\mathcal{O}^q$ and the input ontology $\mathcal{O}$ with respect to the non-forgotten symbols $sig(\mathcal{O})nF$.

The query reduction process follows the algorithm presented in Section 4.8. As such, we identify the subset D^d of $\mathcal{O}^{int}$ that is specified by (4.90), and extract only a slice of it. The ontology D^q is the extracted slice. We use it in *Role Propagation* inferences, and append the conclusions of the inferences to the remainder of $\mathcal{O}^{int}$. The *query reduced* ontology $\mathcal{O}^q$ is the subset of axioms that remain in $\mathcal{O}^{int}$ after removing the subset D^q, and adding the conclusions of the *Role Propagation* inferences. We use the symbol D^u to denote the clauses of D^d that are not in D^q.

The task of eliminating definers from $\mathcal{O}^q$ is more complex than the one performed in the deductive customization in Chapter 4. The complexity of the task comes from the clauses of D^u that were preserved by the query reduction process. We split the

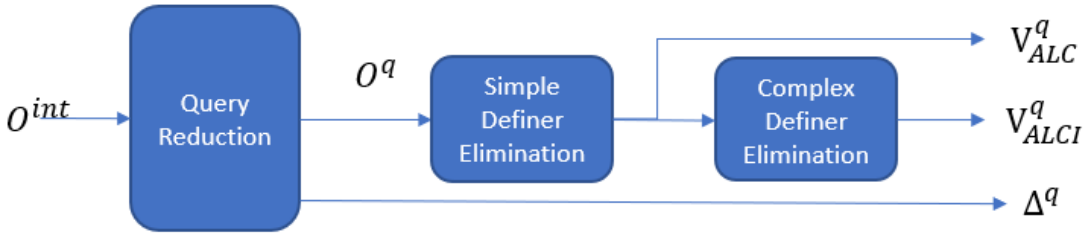

Figure 5.1: Query forgetting customization.

task in two processes, the *simple definer elimination* process, and the *complex definer elimination* process.

The *simple definer elimination* process eliminates *simple* definers from $\mathcal{O}^q$. These are a subset of the definers that do not occur with negative polarity in D^u. They are called *simple* because they are relatively easy to eliminate. The definer elimination process described in Section 4.5 is used to eliminate them. The output of this process is the $\mathcal{ALC}$ representation $\mathcal{V}^q_{\mathcal{ALC}}$ of the query forgetting views of the input ontology with respect to the forgetting signature. From the construction of $\mathcal{O}^q$, and the fact that elimination of simple definers preserves model inseparability (see Theorem 4.5.5), it follows that $\mathcal{V}^q_{\mathcal{ALC}}$ is concept inseparable from the input ontology $\mathcal{O}$ with respect to the forgetting signature $\mathcal{F}$. That is, both ontologies, $\mathcal{O}$ and $\mathcal{V}^q_{\mathcal{ALC}}$ coincide on all entailed subsumption axioms over the non-forgotten signature $sig(\mathcal{O}) \cap \mathcal{F}$.

The *complex definer elimination* process eliminates the remainder of the non-cyclic definers from $\mathcal{O}^q$. They are called *complex* because they require an elimination calculus which may introduce new $\mathcal{ALCI}$ axioms. The output of this process is the $\mathcal{ALCI}$ representation $\mathcal{V}^q_{\mathcal{ALCI}}$ of the query forgetting view of the input ontology with respect to the forgetting symbols.

5.2 Query Reduction

In this section we give an informal description of the query reduction method. Technical details and proofs are deferred to the next section.

The input to the query reduction method is the intermediate ontology $\mathcal{O}^{int}$ obtained from the input ontology $\mathcal{O}$ by the *resolution* stage of our forgetting framework. The outputs are the two sets, $\mathcal{O}^q$ and D^q, explained earlier. The ontology $\mathcal{O}^q$ enhances $\mathcal{O}^d$ with axioms from D^d and the *Role Propagation* conclusions that would be otherwise lost by the exclusion of the clauses not preserved from D^d (see Section 4.8 for more explanation). The trick is to identify the subset D^u of D^d required to attain query inseparability from the input ontology $\mathcal{O}$ with respect to the non-forgotten symbols $sig(\mathcal{O}) \cap \mathcal{F}$. We can see from Lemmas 3.5.7 and 3.5.10 regarding transitivity of inseparability relations that D^u needs only preserve query inseparability with $\mathcal{O}^{int}$ with respect to the non-forgotten symbols $sig(\mathcal{O}) \cap \mathcal{F}$, because $\mathcal{O}$ and $\mathcal{O}^{int}$ are model inseparable with respect to the same set of symbols.

The following three examples motivate a characterization of the clauses of D^u. Each example considers an intermediate ontology $\mathcal{O}^{int}$, such that the subset of D^d of

$\mathcal{O}^{int}$ comprises only one clause. The query reduced ontology $\mathcal{O}^{q}$ will be assumed to be the ontology extracted from the intermediate ontology by excluding the D^{d} clause, and preserving the conclusion of a *Role Propagation* inference triggered by the excluded clause when applicable. Thus, $\mathsf{D}^{u} = \emptyset$ in the three examples. Finally, the answers of $\mathcal{O}^{q}$ to queries will be compared to the answers of the intermediate ontology given an ABox $\mathcal{A}$.

Example 1. All negative definers occur positively below universal role restriction:

Consider the intermediate ontology $\mathcal{O}_1^{int}$ comprising the following axioms.

$$\top \sqsubseteq A_1 \sqcup \forall r.D_1 \tag{5.1}$$

$$\top \sqsubseteq A_2 \sqcup \forall r.D_2 \tag{5.2}$$

$$\top \sqsubseteq \neg D_1 \sqcup \neg D_2 \sqcup C \tag{5.3}$$

where D_1 and D_2 are definers. Let $\mathcal{A}$ the following ABox.

$$A_1(a_1) \tag{5.4}$$

$$A_2(a_2) \tag{5.5}$$

$$r(a_1, b) \tag{5.6}$$

$$r(a_2, b); \tag{5.7}$$

Recall that D^{d} is the subset of $\mathcal{O}^{int}$ where two or more definers occur with negative polarity. In this example, D^{d} is the set comprising the clause (5.3). Let q_1 and q_2 be the two following conjunctive queries.

$$q_1(x) = C(x) \tag{5.8}$$

$$q_2 = \exists y.C(y) \tag{5.9}$$

For the knowledge base $(\mathcal{O}_1^{int}, \mathcal{A})$ the answer to the query $q_1(x)$ is 'b', because (5.1) together with (5.4) and (5.6) imply that b is an instance of D_1. Also, (5.2) together with (5.5) and (5.7) imply that b is an instance of D_2. And, the two conclusions, together with (5.3) imply that b is an instance of C. In the same way, the answer to the query q_2 is *true*.

Let $\mathcal{O}^{q}$ be the ontology consisting of (5.1) and (5.2), and enhanced with the conclusions of the possible *Role Propagation* inferences but excluding (5.3). In this example, a *Role Propagation* inference can be performed with the trigger clause (5.3), and its

conclusion is the following axiom.

$$:A_1 \sqcap :A_2 \sqsubseteq \forall r.C \tag{5.10}$$

So, in total, $\mathcal{O}^q$ is the following ontology

$$:A_1 \sqsubseteq \forall r.D_1$$
$$:A_2 \sqsubseteq \forall r.D_2$$
$$:A_1 \sqcap :A_2 \sqsubseteq \forall r.C$$

Answers to $q_1(x)$ can only be obtained from (5.10). But since there are no common instances of A_1 and A_2 in the ABox $\mathcal{A}$, no answer would be obtained from $(\mathcal{O}^q;\mathcal{A})$ to $q_1(x)$. Furthermore, *false* would be obtained as the answer to q_2. But the answers obtained from $(\mathcal{O}_1^{int};\mathcal{A})$ are b and *true*. So, we see that the clause (5.3) is required to attain query inseparability with respect to S.

Example 2. One negative definer occurs positively below existential role restriction:

Consider the intermediate ontology $\mathcal{O}_2^{int}$ comprising the following axioms.

$$:A_1 \sqsubseteq \exists r.D_1 \tag{5.11}$$
$$:A_2 \sqsubseteq \forall r.D_2 \tag{5.12}$$
$$:D_1 \sqcap :D_2 \sqsubseteq C \tag{5.13}$$

where D_1 and D_2 are definers.

In this example, D_1 occurs below existential role restriction in $\mathcal{O}_2^{int}$, but it occured below universal role restriction in $\mathcal{O}_1^{int}$.

Consider the ABox $\mathcal{A}$ from the previous example. There are no answers to $q_1(x)$ in $(\mathcal{O}_2^{int};\mathcal{A})$. Also, $(\mathcal{O}_2^{int};\mathcal{A})$ answers q_2 with *false*, because models of $(\mathcal{O}_2^{int};\mathcal{A})$, where b is not an instance of D_1, can be constructed.

As in the previous example, $\mathcal{O}^q$ is the ontology consisting of the axioms (5.11), (5.12), but excluding (5.13), and enhanced with the conclusions of the possible Role Propagation inference, which in this example is the following clause.

$$:A_1 \sqcap :A_2 \sqsubseteq \exists r.C \tag{5.14}$$

Altogether, $\mathcal{O}^q$ is the following ontology.

$$\begin{aligned}
&: A_1 \sqsubseteq \exists r.D_1 \\
&: A_2 \sqsubseteq \forall r.D_2 \\
&: A_1 \sqcap \neg A_2 \sqsubseteq \exists r.C
\end{aligned}$$

We can see that the answers to $q_1(x)$ and q_2 for $(\mathcal{O}^q; \mathcal{A})$ coincides with those for $(\mathcal{O}_2^{int}; \mathcal{A})$. So, the clause (5.13) was not required.

There is an important case when $A_2(a_1)$ is added to the ABox $\mathcal{A}$. In this case, no answers would be obtained from $(\mathcal{O}_2^{int}; \mathcal{A})$ for $q_1(x)$, but *true* would be obtained for q_2. Interestingly, $(\mathcal{O}^q; \mathcal{A})$ would obtain the expected answers for both queries. In particular, the answer *true* would be obtained for q_2 due to the clause (5.14) computed by the *Role Propagation* inference.

We summarize the above observations as follows. When one negative definer occurring in (5.13) is used positively in the intermediate ontology below existential role restriction, we observe the following.

1. Explicit instances of C, that is, individuals occurring in $\mathcal{A}$, cannot be obtained.

2. When in the ABox a_1 is an instance of A_1 and A_2, implicit instance of C is obtained. The *Role Propagation* inference is crucial to discover this instance.

3. The clause (5.13) is not required to obtain the correct answers for $q_1(x)$ and q_2 in the two considered versions of the ABox.

Example 3. Two or more negative definer occur positively below existential role restriction:

Consider the intermediate ontology $\mathcal{O}_3^{int}$ comprising the following axioms.

$$: A_1 \sqsubseteq \exists r.D_1 \tag{5.15}$$
$$: A_2 \sqsubseteq \exists r.D_2 \tag{5.16}$$
$$: D_1 \sqcap \neg D_2 \sqsubseteq C \tag{5.17}$$

where D_1 and D_2 are definers.

The definers D_1 and D_2 occur in $\mathcal{O}_3^{int}$ below existential role restrictions in (5.15) and (5.16), respectively. In $\mathcal{O}_2^{int}$, only D_1 occurs below existential role restriction, and in $\mathcal{O}_1^{int}$ the two definers occur below universal role restrictions.

Consider the ABox $\mathcal{A}$ from the first example. There are no answers to $q_1(x)$ in

		D_1	
		Existential	Universal
D_2	Existential	✗	✗
	Universal	✗	✓

Table 5.1: Summary of observations regarding the requirement of (5.3) to obtain correct answers to queries in the three examples.

$(\mathcal{O}_3^{int}; A)$. Furthermore, $(\mathcal{O}_3^{int}; A)$ answer q_2 with *false*, because we can construct models of $(\mathcal{O}_3^{int}; A)$ where the individual b is not an instance of D_1 and D_2.

Let $\mathcal{O}^q$ be the ontology consisting of the two clauses (5.15) and (5.16), but not (5.17). Observe that there are no possible *Role Propagation* inferences because the two negative definers D_1 and D_2 occur positively in $\mathcal{O}_3^{int}$ below existential role restrictions. Yet, we see that $(\mathcal{O}^q; A)$ and $(\mathcal{O}_3^{int}; A)$ have the same answers to $q_1(x)$ and q_2. Moreover, the answers of the two knowledge bases to q_2 remains *false* when updating A in the way done in the second example above. So the correct answers to $q_1(x)$ and q_2 could be obtained without preserving the clause (5.3) in $\mathcal{O}^q$.

Table 5.1 summarizes the observations made in the three examples. A check mark indicates that correct answers to queries are only obtained when the clause $: D_1\ t\ : D_2\ t\ C$ is included in $\mathcal{O}^q$. A cross mark indicates that the correct answers to queries are obtained from $(\mathcal{O}^q; A)$ when the clause $: D_1\ t\ : D_2\ t\ C$ is removed. The headers of the columns and the rows specify the positive occurrences of the definers D_1 and D_2. The terms *existential* and *universal* in the headers mean the following.

Definition 5.2.1. *Let $\mathcal{O}$ be an ontology, and $D \in N_d$ a definer occurring in $\mathcal{O}$. We say D is an* existential definer *if it occurs with positive polarity only below existential role restrictions. Likewise, we say D is an* universal definer *if it occurs with positive polarity only below universal role restrictions.*

It should be understood from Table 5.1 that when D_1 and D_2 are universal, in other words, occurring positively only below universal role restrictions, the clause (5.3) is needed to be preserved in order to obtain the correct answers to the queries $q_1(x)$ and q_2. In the remaining cases, the clause (5.3) is not required to obtain the correct answers to $q_1(x)$ and q_2.

The above can be formulated by uplifting the terms *existential* and *universal* to clauses, as shown in the next definition.

Definition 5.2.2. *Let $\mathcal{O}$ be an ontology, and $C \in \mathcal{O}$ a clause with two or more negative*

Algorithm 5: Outline of method ComputeQueryReducedOntology

Input: Ontology $\mathcal{O}^{int}$, and two sets of clauses D^u, D^q
Output: A query reduced ontology $\mathcal{O}^q$
Initialize: $\mathcal{O}^{rp} := \emptyset; \mathcal{O}^q := \emptyset$
// Compute the Role Propagation inferences using Algorithm 4
1 $\mathcal{O}^{rp} :=$ PerformRolePropagation($\mathcal{O}^{int}; \mathsf{D}^u$)
// Exclude the clauses from D^q
2 $\mathcal{O}^q := \mathcal{O}^{rp} \cap \mathsf{D}^q$
Return: $\mathcal{O}^q$

definers. We say that C is existential *if it contains at least one existential definer with negative polarity. We say that C is* universal *if all definers with negative polarity in C are universal.*

It can be seen from the normal form of $\mathcal{O}^{int}$ defined in Section 4.2 that each definer is either existential or universal. Likewise, each clause from D^d is either existential or universal. We can now suggest the following definitions of D^u and D^q based on the above observations.

Definition 5.2.3. *Let $\mathcal{O}^{int}$ be an intermediate ontology computed as explained in Chapter 4. The ontologies D^u and D^q are defined as follow.*

1. *D^u is the subset of $\mathcal{O}^{int}$ consisting of universal clauses.*
2. *D^q is the subset of $\mathcal{O}^{int}$ consisting of existential clauses.*

An algorithm to compute the query reduced ontology $\mathcal{O}^q$ is shown in Algorithm 5. The inputs of the algorithm are the intermediate ontology $\mathcal{O}^{int}$, the set D^u of clauses from D^d that would be preserved in $\mathcal{O}^q$, and the set D^q of clauses that would be removed. Line 1 perform role propagation inferences by invoking Algorithm 4 explained in Section 4.8. The call to Algorithm 4 takes the inputs $\mathcal{O}^{int}$ and D^u. Algorithm 4 computes the set D^d, and performs *Role Propagation* inferences, where an inference is performed only if any of its premises occurs in $\mathsf{D}^d \cap \mathsf{D}^u$. That is, if any premise occurs in the set D^q. The output $\mathcal{O}^{rp}$ of Algorithm 4 is the union of $\mathcal{O}^{int}$ and the conclusions of the *Role Propagation* inferences. Line 2 removes the D^q clauses from $\mathcal{O}^{rp}$, the remainder of the axioms form the reduced ontology $\mathcal{O}^q$.

The following theorem asserts the query inseparability property of $\mathcal{O}^q$

Theorem 5.2.4. *Let $\mathcal{O}$ be an ontology, and F a forgetting signature. Let $\mathcal{O}^{int}$ be the intermediate ontology obtained from $\mathcal{O}$ with respect to F, and D^u be the subset of $\mathcal{O}^{int}$*

defined in Definition 5.2.3. Suppose $\mathcal{O}^q$ is an ontology computed from $\mathcal{O}^{int}$ by invoking Algorithm 5 with the inputs $\mathcal{O}^{int}$ and D^u. Then, $\mathcal{O}^q$ is query inseparable from $\mathcal{O}^{int}$ with respect to the symbols $sig(\mathcal{O}) \cap \mathsf{F}$, in symbols $\mathcal{O}^{int} \overset{Q}{\underset{sig(\mathcal{O})\cap\mathsf{F}}{\equiv}} \mathcal{O}^q$.

5.3 Technical Proof

In this section we prove Theorem 5.2.4. The proof follows the outline of the proof of Theorem 4.4.6.

Suppose $\mathsf{S} = sig(\mathcal{O}) \cap \mathsf{F}$ and $\mathsf{S}^\emptyset = sig(\mathcal{O}^{int}) \cap \mathsf{N}_d$. Recall that the method of computing $\mathcal{O}^{int}$ may eliminate clauses following exhaustive resolution, or through the purification rule. Symbols from $sig(\mathcal{O}) \cap \mathsf{F}$ may get eliminated by deletion of redundant clauses or purification. Therefore, there might be non-forgetting symbols in S not present in $\mathsf{S}^\emptyset$. As taking account of this technicality would unnecessarily complicate the proof, we assume in this section only that $\mathcal{O}^{int}$ is incremented with the axioms $A \sqsubseteq \top$ for every concept name $A \in \mathsf{S} \cap \mathsf{S}^\emptyset$, and $\top \sqsubseteq \forall r.\top$ for every role name $r \in \mathsf{S} \cap \mathsf{S}^\emptyset$. The axioms are only added so that the signature of $\mathcal{O}^{int}$ modulo definers coincides with the non-forgetting symbols $sig(\mathcal{O}) \cap \mathsf{F}$ in the original ontology. Because these axioms do not change the content of $\mathcal{O}^{int}$, $\mathcal{O}$ and $\mathcal{O}^{int}$ are model inseparable with respect to $sig(\mathcal{O}) \cap \mathsf{F}$.

Let $\mathcal{O}^{rp}$ be the extension of $\mathcal{O}^{int}$ with the conclusions of the *Role Propagation* inferences performed at line 1 of Algorithm 5. This implies $\mathcal{O}^q$ is a subset of $\mathcal{O}^{rp}$ and

$$\mathcal{O}^{rp} = \mathcal{O}^q \cup \mathsf{D}^q \tag{5.18}$$

We have the following lemma.

Lemma 5.3.1. *Consider $\mathcal{O}^{int}$ and $\mathcal{O}^{rp}$ from above. The following are equivalent.*

$$\mathcal{O}^{int} \overset{Q}{\underset{sig(\mathcal{O})\cap\mathsf{F}}{\equiv}} \mathcal{O}^q \tag{5.19}$$

$$\mathcal{O}^{rp} \overset{Q}{\underset{sig(\mathcal{O})\cap\mathsf{F}}{\equiv}} \mathcal{O}^q \tag{5.20}$$

Proof. We can see that $\mathcal{O}^{rp}$ is logically equivalent to $\mathcal{O}^{int}$ because $\mathcal{O}^{int}$ is a subset of $\mathcal{O}^{rp}$, and the remaining clauses of $\mathcal{O}^{rp}$ are conclusions of the clauses of $\mathcal{O}^{int}$. Recall from Lemma 3.5.7 that query inseparability is transitive. We have the following conclusions. The last two proves the lemma.

1. Since $\mathcal{O}^{rp}$ and $\mathcal{O}^{int}$ are logically equivalent as shown above, we see from Lemmas 3.5.8 and 3.5.12 that $\mathcal{O}^{rp}$ and $\mathcal{O}^{int}$ are model inseparable with respect to $sig(\mathcal{O}) \cap \mathbf{F}$.

2. From the previous conclusion and Lemma 3.5.10 which says that model inseparability implies query inseparability, we have that $\mathcal{O}^{rp}$ and $\mathcal{O}^{int}$ are query inseparable with respect to $sig(\mathcal{O}) \cap \mathbf{F}$.

3. Assume (5.19) is true, from the previous conclusion and Lemma 3.5.7 regarding the transitivity of query inseparability, we conclude that (5.20) is also true. In other words, (5.19) implies (5.20).

4. In the same way of 3, we see that (5.20) implies (5.19).

□

To prove Theorem 5.2.4 we prove (5.20) instead of (5.19). Suppose we have $q(\vec{x})$ as an arbitrary conjunctive query. We need to prove the following.

1. An answer $\vec{a}$ for $q(\vec{x})$ in $(\mathcal{O}^q; \mathcal{A})$ is also an answer to the same query in $(\mathcal{O}^{rp}; \mathcal{A})$. In symbols, $(\mathcal{O}^q; \mathcal{A}) \models q(\vec{a})$ implies $(\mathcal{O}^{rp}; \mathcal{A}) \models q(\vec{a})$.

2. An answer $\vec{a}$ for $q(\vec{x})$ in $(\mathcal{O}^{rp}; \mathcal{A})$ is also an answer to the same query in $(\mathcal{O}^q; \mathcal{A}))$. In symbols, $(\mathcal{O}^{rp}; \mathcal{A}) \models q(\vec{a})$ implies $(\mathcal{O}^q; \mathcal{A}) \models q(\vec{a})$.

The first statement is always true because $\mathcal{O}^q$ is a subset of $\mathcal{O}^{rp}$ and the logic considered is monotonic. It can be proved by contradiction. Suppose that $\vec{a}$ is an answer for $q(\vec{x})$ in $(\mathcal{O}^q; \mathcal{A})$, and there is a model $\mathcal{I}$ of $(\mathcal{O}^{rp}; \mathcal{A})$ where $q(\vec{a})$ is not true. Recall that $\mathcal{I}$ is a model of the knowledge base $(\mathcal{O}^{rp}; \mathcal{A})$ if and only if $\mathcal{I}$ is a model of $\mathcal{O}^{rp}$, and $\mathcal{I}$ is a model of $\mathcal{A}$. We observe from this that $\mathcal{I}$ is a model of $\mathcal{O}^q$, because $\mathcal{O}^q \subseteq \mathcal{O}^{rp}$. But then we have that $\mathcal{I}$ is a model of the knowledge base $(\mathcal{O}^q; \mathcal{A})$. We have now established the following, which yield a contradiction.

1. $\mathcal{I} \not\models q(\vec{a})$ by assumption.

2. $\mathcal{I} \models q(\vec{a})$ because $\mathcal{I} \models (\mathcal{O}^q; \mathcal{A})$ and $(\mathcal{O}^q; \mathcal{A}) \models q(\vec{a})$.

Therefore, there is no such model of $(\mathcal{O}^{rp}; \mathcal{A})$ where $q(\vec{a})$ is not true. In other words, $(\mathcal{O}^q; \mathcal{A}) \models q(\vec{a})$ implies $(\mathcal{O}^{rp}; \mathcal{A}) \models q(\vec{a})$. This proves the following lemma.

Lemma 5.3.2. *$\mathcal{O}^q$ and $\mathcal{O}^{rp}$ are not query inseparable if and only if $(\mathcal{O}^{rp}; \mathcal{A}) \models q(\vec{a})$ and $(\mathcal{O}^q; \mathcal{A}) \not\models q(\vec{a})$ where $\vec{a}$ is an answer to a query $q(\vec{x})$.*

The following lemma finds the root cause of query inseparability from a model theoretic perspective when D^q consists of one clause.

Lemma 5.3.3. *Suppose* $\mathsf{D}^q = \{ \top \sqsubseteq D_1 \sqcup \cdots \sqcup D_n \mid E \}$ *where* D_1 *is existential. If we have* $\mathcal{O}^{rp} \not\equiv^{\mathcal{Q}}_{sig(\mathcal{O})nF} \mathcal{O}^q$ *then there is a model* $\mathcal{I}$ *of* $(\mathcal{O}^q; \mathcal{A})$ *where* $D_1 \sqcup \cdots \sqcup D_n \sqcup \top \sqsubseteq E$ *is satisfied in* $\mathcal{I}$.

Proof. We prove the lemma by contradiction. Assume $\mathcal{O}^{rp} \not\equiv^{\mathcal{Q}}_{sig(\mathcal{O})nF} \mathcal{O}^q$, and $\top \sqsubseteq D_1 \sqcup \cdots \sqcup D_n \sqcup E$ is true in every model of $(\mathcal{O}^q; \mathcal{A})$. From the first assumption and Lemma 5.3.2 we have that $(\mathcal{O}^{rp}; \mathcal{A}) \models q(\mathbf{a})$ and $(\mathcal{O}^q; \mathcal{A}) \not\models q(\mathbf{a})$ where $\mathbf{a}$ is an answer to $q(\mathbf{x})$. Let $\mathcal{I}$ be an arbitrary model of $(\mathcal{O}^q; \mathcal{A})$ but $q(\mathbf{a})$ is not true in $\mathcal{I}$. We have that $\mathcal{I}$ is a model of $\mathcal{O}^q$ because $\mathcal{I} \models (\mathcal{O}^q; \mathcal{A})$. Furthermore, $\mathcal{I} \models \top \sqsubseteq D_1 \sqcup \cdots \sqcup D_n \sqcup E$ by assumption. Putting both together, we have $\mathcal{I} \models \mathcal{O}^{rp}$. So, $\mathcal{I}$ is a model of the knowledge base $(\mathcal{O}^{rp}; \mathcal{A})$ which yields a contradiction because both the following are true in $\mathcal{I}$.

1. $\mathcal{I} \not\models q(\mathbf{a})$ by construction of $\mathcal{I}$.

2. $\mathcal{I} \models q(\mathbf{a})$ because $\mathcal{I} \models (\mathcal{O}^{rp}; \mathcal{A})$ and $(\mathcal{O}^{rp}; \mathcal{A}) \models q(\mathbf{a})$.

Therefore, either $\mathcal{O}^{rp} \not\equiv^{\mathcal{Q}}_{sig(\mathcal{O})nF} \mathcal{O}^q$ or $D_1 \sqcup \cdots \sqcup D_n \sqcup \top \sqsubseteq E$ is true in $\mathcal{I}$. If $\mathcal{O}^{rp} \not\equiv^{\mathcal{Q}}_{sig(\mathcal{O})nF} \mathcal{O}^q$, it must be that $D_1 \sqcup \cdots \sqcup D_n \sqcup \top \sqsubseteq E$ is true in $\mathcal{I}$. □

We prove that for every model $\mathcal{I}$ of $(\mathcal{O}^q; \mathcal{A})$ where $D_1 \sqcup \cdots \sqcup D_n \sqcup \top \sqsubseteq E$ is satisfied, there is a model $\mathcal{J}$ of $(\mathcal{O}^{rp}; \mathcal{A})$ that coincides with $\mathcal{I}$ on all answers to queries over the non-forgotten symbols $sig(\mathcal{O})nF$. Proving this statement together with Lemma 5.3.2, imply query inseparability between $\mathcal{O}^{rp}$ and $\mathcal{O}^q$ with respect to $sig(\mathcal{O})nF$. This part is of the proof parallels the proof of Lemma 4.4.19, where the model $\mathcal{J}$ was constructed using model transformations applied on $\mathcal{I}$.

There are two concerns that require settling before building this proof. One concern is the data structure used to represent the models of knowledge bases. The other concern is the comparison method that ensures coincidence on the answers to queries. While tree models and bisimulation were used in of Lemma 4.4.19, they are not suitable when an ABox is present. A new algebraic structure, namely *pseudo forests*, is needed. Furthermore, the definition of bisimulation is adapted to use pseudo forests instead of pointed interpretations. The following example shows why tree models are not convenient when an ABox is present.

Example 5.3.4. *Consider a knowledge base $K = (\emptyset; A)$ consisting of the empty ontology and the following ABox A.*

$$r(a_1; b) \tag{5.21}$$

$$r(a_2; b) \tag{5.22}$$

Let I be a model of K over the domain $\Delta^I = \{a_1; a_2; b\}$ with the following interpretations.

$$a_1^I = a_1 \tag{5.23}$$

$$a_2^I = a_2 \tag{5.24}$$

$$b^I = b \tag{5.25}$$

$$r^I = \{(a_1; b); (a_2; b)\} \tag{5.26}$$

We can visualize I with the graph in Figure 5.2, where domain elements are nodes in the graph, and a directed edge is drawn between two nodes x and y if $(x; y) \in r^I$ for a role name r. If we use the tree unravelling algorithm explained in Section 4.4, we would obtain the tree model in Figure 5.3. We can observe that individual b is represented by the two domain elements $a_1{:}r{:}b$ and $a_2{:}r{:}b$ in the unravelled model. This representation is wrong as the individuals a_1 and a_2 should be mapped to a single domain element b in the interpretation.

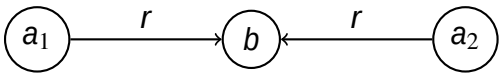

Figure 5.2: Graph representation of I

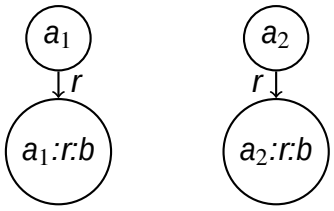

Figure 5.3: Tree models obtained from I by tree unravelling

Pseudo forest can intuitively be described as several separate trees with the following being satisfied.

1. Individuals from the ABox A occur only as root nodes.

2. Edges connecting different trees can only be between the roots.

For instance, the model I in Figure 5.2 is a pseudo forest, because it consists of three tree structures, and each tree comprises only one node which is the root of the tree.

We explain the construction of pseudo forests in the following. We first define the notion of sub-interpretation rooted at an individual $a \in ind(A)$.

Definition 5.3.5. *Let I be a model, and $a, d \in \Delta^I$ be two domain elements in I. We say there is a* path from a to d along $\bigcup_{r \in N_r} r^I$ *if any of the following is true.*

1. $(a, d) \in r^I$ *where* $r \in N_r$.
2. *There are* $r, r_1, r_2, \ldots, r_n \in N_r$ *and* $e_1, e_2, \ldots, e_n \in \Delta^I$ *such that* $(a, e_1) \in r_1^I$, $(e_i, e_{i+1}) \in r_{i+1}^I$, *and* $(e_n, d) \in r^I$, *where* $n \geq 1$ *and* $1 \leq i \leq n - 1$.

Definition 5.3.6. *Let $K = (O, A)$ be a knowledge base, and I a model of K where $x^I = x$ for every individual $x \in ind(A)$. Further, let $a \in ind(A)$ be an individual. For a model J, we say J is a* sub-interpretation of I rooted at a *if the following are true about J.*

1. $\Delta^J = \{a\} \cup \{d \in \Delta^I \mid d \in ind(A)$ *and there is a path from a to d along* $\bigcup_{r \in N_r} r^I\}$,
2. $a^J = a$,
3. $A^J = A^I \cap \Delta^J$,
4. $r^J = r^I \cap (\Delta^J \times \Delta^J)$,

By M we denote the set of all sub-interpretations of I rooted at the individuals from $ind(A)$.

Recall from Section 4.4 the tree unravelling algorithm for constructing tree interpretations. We use this algorithm to construct a tree interpretation for each sub-interpretation in M.

Definition 5.3.7. *Let I be a model of a knowledge base $K = (O, A)$ as in Definition 5.3.6. Let M be the set of all sub-interpretations of I rooted at the individuals from $ind(A)$. Let $tree(I)$ be a function that unravels an interpretation I into a tree interpretation I^t. By $M^t = \{I^t = tree(I) \mid I \in M\}$ we denote the set of tree interpretations corresponding to the sub-interpretations in M.*

A pseudo forest is defined using Definition 5.3.7 as follows.

Definition 5.3.8. *Let I be a model of a knowledge base $K = (O;A)$ as in Definitions 5.3.6 and 5.3.7. Let M^t be the set of tree interpretations constructed from the sub-interpretations of I rooted at the individuals from $ind(A)$. Suppose there are n tree models $I_1;:::;I_n$ with $n \quad 1$ in M^t. A* pseudo forest model *J is the model constructed from the tree models in M^t as follows. The operator t used below means the disjoint union.*

1. $\Delta^J = \Delta^{I_1} t \quad t \Delta^{I_n}$.
2. $A^J = A^{I_1} t \quad t A^{I_n}$ *for every concept name* $A \in N_c \cap N_d$.
3. $r^J = r^{I_1} t \quad t r^{I_n} \cup \{(x;y) \in r^I \mid x;y \in ind(A)\}$ *for every role name* $r \in N_r$.

We explain the above definition of a pseudo forest with the following example.

Example 5.3.9. *Consider the following ontology O.*

$$A_1 \sqsubseteq \exists r.B \tag{5.27}$$

$$A_2 \sqsubseteq \exists r.B \tag{5.28}$$

Let A be the following ABox.

$$A_1(a_1) \tag{5.29}$$

$$A_2(a_2) \tag{5.30}$$

Suppose we have a model I of the knowledge base $(O;A)$. Assume the domain of interpretation of I is the set $\{a_1;a_2;b\}$, and the interpretation function is defined by the following.

$$a_1^I = a_1 \tag{5.31}$$

$$a_2^I = a_2 \tag{5.32}$$

$$A_1^I = \{a_1\} \tag{5.33}$$

$$A_2^I = \{a_2\} \tag{5.34}$$

$$B^I = \{b\} \tag{5.35}$$

$$r^I = \{(a_1;b);(a_2;b);(a_1;a_2)\} \tag{5.36}$$

The model I is visualized in the lift graph of Figure 5.4. Let I_1 be the sub-interpretation of I rooted at a_1, and I_2 the sub-interpretation of I rooted at a_2. They are defined as

follows.

$$\begin{array}{ll} \Delta^{\mathcal{I}_1} = \{a_1; b\} & \Delta^{\mathcal{I}_2} = \{a_2; b\} \\ a_1^{\mathcal{I}_1} = a_1 & a_2^{\mathcal{I}_2} = a_2 \\ A_1^{\mathcal{I}_1} = \{a_1\} & A_1^{\mathcal{I}_2} = \emptyset \\ A_2^{\mathcal{I}_1} = \emptyset & A_2^{\mathcal{I}_2} = \{a_2\} \\ B^{\mathcal{I}_1} = \{b\} & B^{\mathcal{I}_2} = \{b\} \\ r^{\mathcal{I}_1} = \{(a_1; b)\} & r^{\mathcal{I}_2} = \{(a_2; b)\} \end{array}$$

The two models $\mathcal{I}_1$ and $\mathcal{I}_2$ are tree models. So tree unravelling is not required in this case, and we may define the set $\mathcal{M}^t$ as the set consisting of $\mathcal{I}_1$ and $\mathcal{I}_2$. The pseudo forest model $\mathcal{J}$ is constructed as the disjoint union of $\mathcal{I}_1$ and $\mathcal{I}_2$. This means that common domain elements of $\mathcal{I}_1$ and $\mathcal{I}_2$ are copied twice to $\Delta^{\mathcal{J}}$ as different elements. Let $a_1{:}r{:}b$ be the image of $b \in \Delta^{\mathcal{I}_1}$ in $\Delta^{\mathcal{J}}$, and $a_2{:}r{:}b$ be the image of $b \in \Delta^{\mathcal{I}_2}$ in $\Delta^{\mathcal{J}}$. The pseudo forest model $\mathcal{J}$ is defined as follows.

$$\Delta^{\mathcal{J}} = \{a_1; a_2; a_1{:}r{:}b; a_2{:}r{:}b\} \tag{5.37}$$
$$a_1^{\mathcal{J}} = a_1 \tag{5.38}$$
$$a_2^{\mathcal{J}} = a_2 \tag{5.39}$$
$$A_1^{\mathcal{J}} = \{a_1\} \tag{5.40}$$
$$A_2^{\mathcal{J}} = \{a_2\} \tag{5.41}$$
$$B^{\mathcal{J}} = \{a_1{:}r{:}b; a_2{:}r{:}b\} \tag{5.42}$$
$$r^{\mathcal{J}} = \{(a_1; a_1{:}r{:}b); (a_2; a_2{:}r{:}b); (a_1; a_2)\} \tag{5.43}$$

The pseudo forest model $\mathcal{J}$ is depicted in the right graph of Figure 5.4.

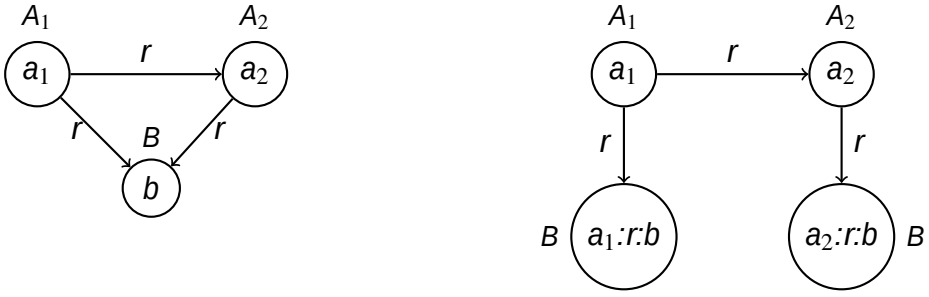

Figure 5.4: Tree model obtained from $\mathcal{I}$ by tree unravelling

Definitions 5.3.6, 5.3.7, and 5.3.8 require that individuals are interpreted as themselves. Thus, we make the unique name assumption [BLR+19]. It is known from the literature that query answering in $\mathcal{ALC}$ is not impacted by the unique name assumption. Satisfiability in knowledge bases can be confusing when working with pseudo forests. We define it in the following.

Definition 5.3.10. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{A})$ be a knowledge base, and $\mathcal{I}$ a pseudo forest model of $\mathcal{K}$. We write $\mathcal{I} \models q(\mathbf{a})$ if and only if $q(\mathbf{a})$ is true at any root in $\mathcal{I}$, and $\mathcal{I} \not\models q(\mathbf{a})$ if and only if $q(\mathbf{a})$ is false at every root in $\mathcal{I}$, where the roots of $\mathcal{I}$ are the individuals from $\mathsf{ind}(\mathcal{A})$; $q(\mathbf{x})$ is a conjunctive query, and $\mathbf{a}$ is any tuple over $\mathsf{ind}(\mathcal{A})$ with the same length as $\mathbf{x}$.*

The following lemma allows us to focus our proofs on pseudo forest models which is true because every model of $(\mathcal{O}, \mathcal{A})$ can be transformed to a pseudo-forest. The proof of the lemma can be found in [GHLS07, BKR+16, BLR+19] for example.

Lemma 5.3.11. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{A})$ be a knowledge base, and $q(\mathbf{x})$ a conjunctive query. $\mathcal{K} \models q(\mathbf{a})$ if and only if there is a pseudo forest model $\mathcal{I}$ of $\mathcal{K}$ but $\mathcal{I} \not\models q(\mathbf{a})$, for any tuple $\mathbf{a}$ over $\mathsf{ind}(\mathcal{A})$ with the same length as $\mathbf{x}$.*

We discuss bisimulation as a comparison method that ensures coincidence of models on answers to queries. Definition 4.4.11 of bisimulation considers pointed interpretations which have one root element. We adapt the definition to pseudo forests.

Definition 5.3.12. *Let g be a set of individual names, and S a signature. Suppose $\mathcal{I}$ and $\mathcal{J}$ are two pseudo forest models. $\mathcal{I}$ and $\mathcal{J}$ are* bisimilar with respect to S and g, *in symbols $\mathcal{I} \overset{\mathsf{g}}{\underset{\mathsf{S}}{\leftrightarrow}} \mathcal{J}$, if and only if there is a relation $R \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ where $(a, a) \in R$ for every individual in g, and for every $(d, d') \in R$ the following hold:*

1. *$d \in A^{\mathcal{I}}$ iff $d' \in A^{\mathcal{J}}$ for all concept names $A \in \mathsf{S}$.*
2. *if $(d, e) \in r^{\mathcal{I}}$ then there is $e' \in \Delta^{\mathcal{J}}$ such that $(d', e') \in r^{\mathcal{J}}$ for every role name $r \in \mathsf{S}$ and $(e, e') \in R$.*
3. *if $(d', e') \in r^{\mathcal{J}}$ then there is $e \in \Delta^{\mathcal{I}}$ such that $(d, e) \in r^{\mathcal{I}}$ for every role name $r \in \mathsf{S}$ and $(e, e') \in R$.*

Bisimulation ensures coincidence on answers to arbitrary conjunctive queries. The following lemma proves this statement.

Lemma 5.3.13. *Let $\mathcal{I}$ and $\mathcal{J}$ be two bisimilar models with respect to a set S of symbols, and a set g of individuals. Suppose $q(\vec{x})$ is a conjunctive query where $sig(q) \subseteq \mathsf{S}$, and $\vec{a}$ is an answer to $q(\vec{x})$ where $\vec{a} \subseteq \mathsf{g}$. Then $q(\vec{a})$ is true in $\mathcal{I}$ if and only if $q(\vec{a})$ is true in $\mathcal{J}$ for any answer $\vec{a}$ of $q(\vec{x})$.*

Proof. Suppose $\mathcal{I} \models q(\vec{a})$, then there is a mapping p that maps each anonymous variable in q, that is, a variable in q that is not in $\vec{x}$, to an element in $\Delta^{\mathcal{I}}$, and each answer variable x at position i in $\vec{x}$ to an individual in $\vec{a}$ at the same position in $\vec{a}$. Furthermore, for every two variables u and v occurring in q, the following must be true.

1. If $A(u)$ is an atom in q, then $\mathsf{p}(u) \in A^{\mathcal{I}}$, for any $A \in \mathsf{S}$.
2. If $r(u,v)$ is an atom in q, then $(\mathsf{p}(u),\mathsf{p}(v)) \in r^{\mathcal{I}}$, for any $r \in \mathsf{S}$.

Let s be a function that maps a domain element from $\Delta^{\mathcal{I}}$ to a bisimilar element in $\Delta^{\mathcal{J}}$. The function s is a total function, because each domain element in $\Delta^{\mathcal{I}}$ is either an individual a which has a map in R, or an element reachable from a through $\bigcup_S r^{\mathcal{I}}$ for every $r \in N_r$. The latter elements are guaranteed to have a map in R by Conditions 2 and 3 of Definition 5.3.12.

Consider the mapping $\mathsf{b} = \mathsf{s} \circ \mathsf{p}$ which is the composition of p and s. We can see that following about b.

1. b maps each answer variable x at position i in $\vec{x}$ to an individual in $\vec{a}$ at the same position in $\vec{a}$, because $a = \mathsf{p}(x)$ and $a = \mathsf{s}(a)$.
2. b maps each anonymous variable occurring in q to a domain element in $\Delta^{\mathcal{J}}$, because s is a total function.
3. If $A(u)$ is an atom in q, then $\mathsf{b}(u) \in A^{\mathcal{J}}$, because $\mathsf{p}(u) \in A^{\mathcal{I}}$, and $\mathsf{p}(u)$ must be mapped through s to an element in the extension of A in $\mathcal{J}$.
4. If $r(u,v)$ is an atom in q. then $(\mathsf{b}(u),\mathsf{b}(v)) \in r^{\mathcal{J}}$, because $\mathsf{p}(u)$ must be mapped through s to an element $d \in \Delta^{\mathcal{J}}$, and v to an element e where $(d,e) \in r^{\mathcal{J}}$.

So, if $q(\vec{a})$ is true in $\mathcal{I}$, then $q(\vec{a})$ is true in $\mathcal{J}$. The same argument shows that if $q(\vec{a})$ is true in $\mathcal{J}$, then $q(\vec{a})$ is true in $\mathcal{I}$. □

Example 5.3.14. *Consider the following ontology $\mathcal{O}$.*

$$A \sqsubseteq \forall r.B_1 \sqcap \forall r.B_2 \tag{5.44}$$

$$B_1 \sqcap B_2 \sqsubseteq \exists s.C \tag{5.45}$$

$$B_1 \sqcap B_2 \sqsubseteq \bot \tag{5.46}$$

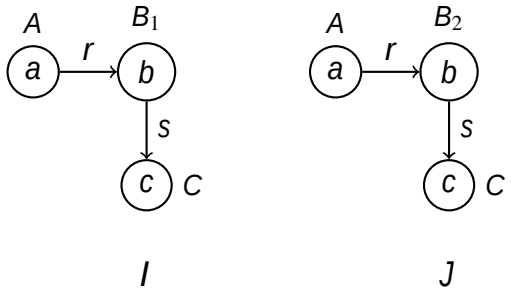

Figure 5.5: I and J are bisimilar with respect to $\{A, r\}$ and $\{a, b\}$.

and the following ABox A

$$A(a) \tag{5.47}$$

$$r(a, b) \tag{5.48}$$

Two models I, J *of the knowledge base* $K = (O, A)$ *are shown in Figure 5.5. Both models are interpretations over the domain* $\{a, b, c\}$*, where* $a, b \in ind(A)$*. Let* $q_1(x)$ *and* $q_2(x)$ *be the following queries.*

$$q_1(x) = \exists y, z.A(x) \wedge r(x, y) \wedge s(y, z) \wedge C(z) \tag{5.49}$$

$$q_2(x) = B_1(x) \tag{5.50}$$

The two models coincide on the answer 'a' for q_1 *because they are bisimilar with respect to the symbols* $\{A, C, r, s\}$ *and the individuals* $\{a, b\}$*. The bisimulation relation is* $R = \{(a, a), (b, b), (c, c)\}$*. However the two models do not coincide on their answers to* q_2 *as* $I \not\models B_1(b)$ *but no answers for* q_2 *exist in* J*. The models* I *and* J *are different on the answers of* $q_2(x)$ *because they are not bisimilar with respect to any signature containing* B_1 *and the individuals* $\{a, b\}$*.*

The following lemma asserts important properties of bisimulation.

Lemma 5.3.15. *Bisimulation is a transitive operation. Furthermore, if* S_1 *and* S_2 *are two sets of symbols such that* $\mathsf{S}_1 \subseteq \mathsf{S}_2$ *then* $I \overset{\mathfrak{g}}{\underset{\mathsf{S}_2}{\leftrightarrow}} J$ *implies* $I \overset{\mathfrak{g}}{\underset{\mathsf{S}_1}{\leftrightarrow}} J$*, where* I *and* J *are pseudo forest models, and* $\mathfrak{g}$ *is a set of individuals.*

We have shown by Lemma 5.3.2 that O^{rp} and O^q are query separable if $(O^{rp}, A) \models q(\mathfrak{a})$ and $(O^q, A) \not\models q(\mathfrak{a})$, where $q(x)$ is a conjunctive query over the non-forgotten symbols $sig(O) \cap F$, and $\mathfrak{a}$ is an answer to $q(x)$. When D^q contains only the clause $\bot \leftarrow D_1(t) \wedge \cdots \wedge D_n(t) \wedge E$ where D_1 is an existential definer, Lemma 5.3.3 showed that O^{rp}

and $\mathcal{O}^q$ are query separable if and only if there is a model $\mathcal{I}$ of $(\mathcal{O}^q; \mathcal{A})$ where $D_1 \sqcup \cdots \sqcup D_n(x) : E$ is true. We may assume that $\mathcal{I}$ is a pseudo forest model by Lemma 5.3.11. We show that there is a model $\mathcal{J}$ of $(\mathcal{O}^{rp}; \mathcal{A})$ that coincides with $\mathcal{I}$ on all answers to queries over the non-forgotten symbols $sig(\mathcal{O}) \cap \mathbf{F}$. This will cause a contradiction because the following will be true about $\mathcal{J}$.

1. $\mathcal{J} \not\models q(a)$, because $\mathcal{J}$ coincides with $\mathcal{I}$ on all answers to queries over $sig(\mathcal{O}) \cap \mathbf{F}$.
2. $\mathcal{J} \models q(a)$ because $\mathcal{J}$ is a model of $(\mathcal{O}^{rp}; \mathcal{A})$ and $(\mathcal{O}^{rp}; \mathcal{A}) \models q(a)$.

The contradiction implies that $\mathcal{O}^{rp}$ and $\mathcal{O}^q$ are query inseparable. The case when $\mathbf{D}^q$ contains n clauses where n is arbitrary, can be generalized from the above by induction. We construct a sequence of n ontologies where each ontology excludes only one clause from $\mathbf{D}^q$ such that the first ontology coincides with $\mathcal{O}^{rp}$ and the last coincides with $\mathcal{O}^q$. The above argument shows that every two consecutive ontologies in this sequence are query inseparable, and by transitivity of query inseparability (see Lemma 3.5.7) we prove query inseparability of $\mathcal{O}^{rp}$ and $\mathcal{O}^q$ with respect to $sig(\mathcal{O}) \cap \mathbf{F}$.

The central part as discussed above is finding the model $\mathcal{J}$. We construct $\mathcal{J}$ from $\mathcal{I}$ by eliminating all occurrences of domain elements satisfying $D_1 \sqcup \cdots \sqcup D_n(x) : E$. Suppose there are m domain elements in $\mathcal{I}$ satisfying $D_1 \sqcup \cdots \sqcup D_n(x) : E$. We construct m models $\mathcal{I}_1, \mathcal{I}_2, \ldots$ where each model eliminates exactly one domain element satisfying $D_1 \sqcup \cdots \sqcup D_n(x) : E$ from the previous model, where $\mathcal{I}_0 = \mathcal{I}$ and $\mathcal{I}_m = \mathcal{J}$. Each two consecutive models in this sequence are bisimilar with respect to $sig(\mathcal{O}) \cap \mathbf{F}$ and $ind(\mathcal{A})$. By transitivity of bisimulation, we obtain that $\mathcal{I}$ and $\mathcal{J}$ are bisimilar with respect to $sig(\mathcal{O}) \cap \mathbf{F}$ and $ind(\mathcal{A})$.

Lemma 5.3.16. *Let $\mathcal{K} = (\mathcal{O}; \mathcal{A})$ be a knowledge base. Let $\mathcal{O}^{rp}$ and $\mathcal{O}^q$ be as in Lemma 5.3.3 where $\mathcal{O}^{rp} \cap \mathcal{O}^q = \{ : D_1 \sqcup \cdots \sqcup : D_n \sqcup E \mid D_1, \ldots, D_n$ are definers, and D_1 is an existential definer. Suppose $\mathcal{I}$ is a pseudo forest model of $(\mathcal{O}^q; \mathcal{A})$ and $D_1 \sqcup \cdots \sqcup D_n(x) : E$ is true in $\mathcal{I}$. There is a model $\mathcal{J}$ with the following properties:*

1. *$\mathcal{J} \models \mathcal{O}^q$;*
2. *There is no $e \in \Delta^{\mathcal{J}}$ such that $e \in D_1^{\mathcal{J}} \setminus \cdots \setminus D_n^{\mathcal{J}} \setminus (: E)^{\mathcal{J}}$; and*
3. *$\mathcal{I}$ and $\mathcal{J}$ are bisimilar with respect to $sig(\mathcal{O}) \cap \mathbf{F}$ and $ind(\mathcal{A})$.*

Proof. Recall from the normal form of $\mathcal{O}^q$ that a definer occurs with positive polarity only below existential role restrictions, or only below universal role restrictions. It

also occurs with negative polarity only as a top level negated disjunct. Let I_0 be a model that coincides with I on everything but reinterprets every definer D as below. We assume $O^q \models G \sqsubseteq \exists r.D$ or $O^q \models G \sqsubseteq \forall r.D$ where G is an $\mathcal{ALC}$ concept. If O^q has several consequences of these forms, e.g., $O^q \models G_i \sqsubseteq \exists r.D$ with $1 \leq i \leq n$ and $n > 1$, then we assume G is the conjunction of all such G_i's. The definer D is interpreted in I_0 by the following.

$$D^{I_0} := D^I \setminus \{y \in \Delta^I \mid \exists x.(x,y) \in r^I \text{ and } x \notin G^I \text{ where } (O^q \models G \sqsubseteq \exists r.D, \text{ or } O^q \models G \sqsubseteq \forall r.D)\} \quad (5.51)$$

where $D \in N_d, r \in N_r$, and G is an $\mathcal{ALC}$ concept. The following two properties are true about I_0.

1. I_0 is a model of (O^q, A). The is true because all clauses of O^q of the forms $G \sqsubseteq \exists r.D$ or $G \sqsubseteq \forall r.D$ are true in I_0 by the side conditions of (5.51). Also, since (5.51) only removes elements from the extension of D, clauses of the form $\neg D \sqsubseteq H$, where H is an $\mathcal{ALC}$ concept, remain satisfied in I_0.

2. I and I_0 are bisimilar with respect to $sig(O) \cap N_F$ and the individuals $ind(A)$. This is true because only the interpretations of definers are modified by (5.51).

Equation (5.51) imposes conditions on the predecessors of elements in the extension of D. Suppose e is such an element, the second conjunct in the right-hand side of (5.51) requires e to have at least one predecessor e_{pre} via a role r such that there is a logical consequence $G \sqsubseteq Q\, r.D$ of O^q where $Q \in \{\exists, \forall\}$ and $e_{pre} \notin G^{I_0}$.

We now construct a sequence of interpretations $I_1, I_2, \ldots$. The interpretations are constructed such that I_{k+1} eliminates one domain element $e \in D_1^{I_k} \setminus \ldots \setminus D_n^{I_k} \setminus (\neg E)^{I_k}$ where $k \geq 0$, and I_k and I_{k+1} are bisimilar with respect to $sig(O) \cap N_F$ and $ind(A)$. The limit of this sequence is J. The construction considers the following conditions.

1. Whether e is an individual from $ind(A)$.

2. Whether the definers $D_1, D_2, \ldots, D_n$ occur positively below the same role in O^q.

3. Whether only D_1 is existential.

Five cases are obtained as different combinations of the above conditions. Figure 5.6 visualizes them in more detail. Case I assumes that $e \in ind(A)$ and the definers $D_1, D_2, \ldots, D_n$ do not occur positively below the same role. Case II assumes that

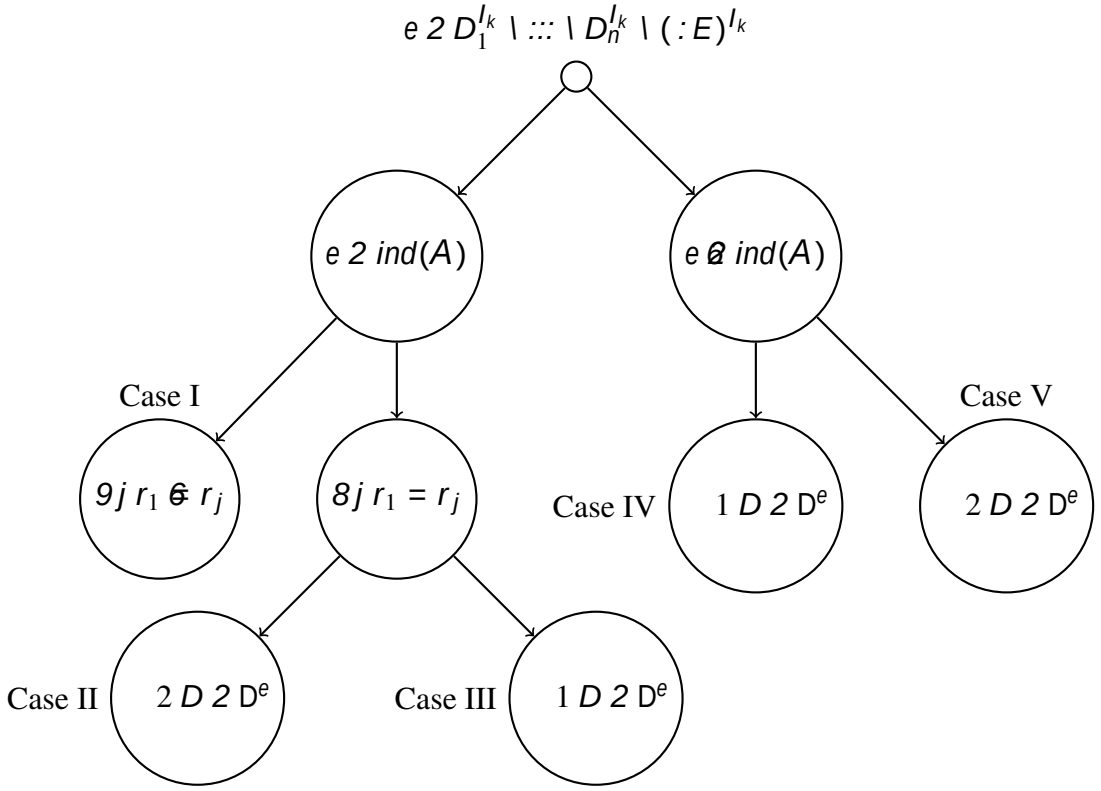

Figure 5.6: Cases satisfied by e in the construction of I_{k+1}

$e \in ind(A)$, the definers $D_1, D_2, \ldots, D_n$ occur positively below the same role, and two or more definers, including D_1, are existential. Case III assumes that $e \in ind(A)$, the definers $D_1, D_2, \ldots, D_n$ occur positively below the same role, and only D_1 is existential. Case IV assumes that $e \notin ind(A)$, and only D_1 is existential. Case V assumes that $e \notin ind(A)$, and two or more definers, including D_1, are existential. Note that Cases IV and V implicitly assume that the definers $D_1, D_2, \ldots, D_n$ occur positively below the same role because e is not an individual from $ind(A)$ and so there can only be one incoming edge to e in I labeled with one role. For Case I it is irrelevant whether there is only one existential definer or more than one. We consider all five cases. A transformation which ensures $e \notin D_1^{I_{k+1}} \cup \ldots \cup D_n^{I_{k+1}} \cup (\neg E)^{I_{k+1}}$ is shown for each case.

Suppose first that $e \in ind(A)$, and consider the cases I, II, and III. As discussed in the comment after the transformation in Equation (5.51), e has n predecessors e^i_{pre} such that $O^q \not\models G \sqsubseteq Q\, r_i.D_i$, $e^i_{pre} \notin G_i^{I_k}$, and $(e^i_{pre}, e) \in r_i^{I_k}$, where $1 \leq i \leq n$. Denote by P the set of predecessors $e^i_{pre} \notin G_1^{I_k}$ such that $(e^i_{pre}, e) \in r_1^{I_k}$. We construct I_{k+1} as follows.

Case I: $e \in ind(A)$ and $r_1 \not\sim r_j$ for some j where $1 \leq j \leq n$:

Extend I_k to I_{k+1} as follows. For every $d \in P$, a new domain element e' is introduced in $\Delta^{I_{k+1}}$. Additionally, the following updates are performed:

1. $e' \in A^{I_{k+1}}$ if $e \in A^{I_k}$ for every concept name $A \in sig(\mathcal{O}) \cap F$;
2. $(d, e') \in r_1^{I_{k+1}}$;
3. $(e', f) \in r^{I_{k+1}}$ if $(e, f) \in r^{I_k}$ for every role name $r \in sig(\mathcal{O}) \cap F$;
4. $e' \in D^{I_{k+1}}$ if $d \in (\forall r_1.D)^{I_k}$;
5. $D_1^{I_{k+1}} := (D_1^{I_k} \cap \{e\}) \cup \{e'\}$.

First, We prove that I_{k+1} is a model of $\mathcal{O}^q$. Two changes in I_{k+1} are implied by the above transformations. The first change is a new r_1-successor for every $d \in P$ added to the interpretation of D_1. The second change is removing e from the interpretation of D_1. For the first change, we observe from Steps 1, 2, 3, and 4 of the transformation that the e' elements mirror the domain element e in everything before removing e from the interpretation of D_1. So if only this change introducing the e' domain elements has been performed in I_{k+1}, then $I_{k+1} \models \mathcal{O}^q$ because duplicating domain elements does not invalidate the models of $\mathcal{ALC}$ ontologies. For the other change, it suffices to verify that $d \in (\exists r_1.D_1)^{I_{k+1}}$. This is verified by observing that when e is removed from the interpretation of D_1, the e' elements are simultaneously added to the interpretation of D_1 and by Step 2 of the transformation we have $(d, e') \in r_1^{I_{k+1}}$. So, $I_{k+1} \models \mathcal{O}^q$.

Second, it follows from Step 5 of the transformation that $e \notin D_1^{I_{k+1}} \cup \cdots \cup D_n^{I_{k+1}}$. We also observe that the new elements e' are not in $D_1^{I_{k+1}} \cup \cdots \cup D_n^{I_{k+1}}$ because they were introduced only as r_1 successors, and since for some j where $1 \leq j \leq n$ we have that $r_1 \sqsubseteq r_j$, there must be a definer $D \in \{D_2, \ldots, D_n\}$ where $e' \notin D^{I_{k+1}}$.

Third, we prove that I_{k+1} and I_k are bisimilar with respect to $sig(\mathcal{O}) \cap F$ and $ind(\mathcal{A})$. To this purpose it suffices to have the following bisimulation relation.

$$R = \{(d, d) \in \Delta^{I_k} \times \Delta^{I_{k+1}} \mid d \in \Delta^{I_k} \setminus \Delta^{I_{k+1}}\} \cup \{(e', e) \mid \text{for every } e' \text{ introduced by the transformation}\} \quad (5.52)$$

We explain the above transformation through the example depicted in Figure 5.7. The left graph of the figure shows a model I_k where an individual b has a predecessor a via r, and a predecessor c via s. The individual b is labelled with two definers D_1 and D_2. The definer D_1 is existential, and D_2 is universal. The domain element a is in the interpretation of $\exists r.D$, and c is in the interpretation of $\forall s.D_2$.

In the model I_{k+1}, a new successor e of a via r is introduced by the above transformation. The edges represent the updates to the interpretations of role symbols defined by the third transformation. Finally, e substitutes b in the interpretation of D_1.

The dotted lines indicate bisimilar elements in the relation R. For instance, the dotted line between e and b indicates that $(e;b)\ 2\ R$.

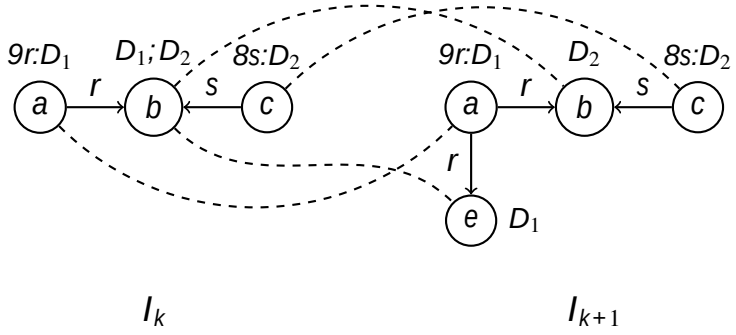

Figure 5.7: Example of the transformation of Case I

Case II: $e\ 2\ ind(A)$, all definers occur below r_1, and two or more are existential:

When $D_1;D_2;:::;D_n$ occur positively below the same role, and at least two of them are existential, we repeat the same transformation in Case I.

We can prove that I_{k+1} is a model of O^q with a similar argument to that used in Case I. We also have that the bisimulation relation in (5.52) remains valid in this case, and e is not in $D_1^{I_{k+1}} \setminus \quad \setminus D_n^{I_{k+1}}$ because of Step 5. It remains to show that the introduced domain elements $e^{\emptyset}$ are not in the set $D_1^{I_{k+1}} \setminus \quad \setminus D_n^{I_{k+1}}$. This can be seen because by assumption there is a definer $D\ 2\ fD_2;:::;D_ng$ which is existential, and at no time in the transformation we make $e^{\emptyset}$ in the interpretation of D.

We explain this transformation through the example in Figure 5.8. The example shows a slight modification of that in Figure 5.7. The individual c is in the interpretation of $9r:D_2$ instead of $8s:D_2$. The output of the transformation remains unchanged. A new r-successor e of a is introduced. The interpretation of D_1 is updated by substituting e for b.

Case III: $e\ 2\ ind(A)$, all definers occur below r_1, and only D_1 is existential:

Since all definers, except D_1, are universal, it must be true that $O^q\ j=\ G_j\ t\ 8r_1:D_j$ for every $2 \quad j \quad n$. Further, it must be true that $O^q\ j=\ G_1\ t\ 9r_1:D_1$, because D_1 is existential. But then the clause $G_1\ t\ :::G_n\ t\ 9r_1:(E\ u\ \bigsqcup_{s=1}^{l} E_s)$ must have been preserved in O^q by a *Role Propagation* inference, where $O^{rp}\ j=\ P_s\ t\ E_s$, and P_s is any concept in $:D_1\ t \quad t\ :D_n$ (The clauses $P_s\ t\ E_s$ are the clauses of the type-2 premise

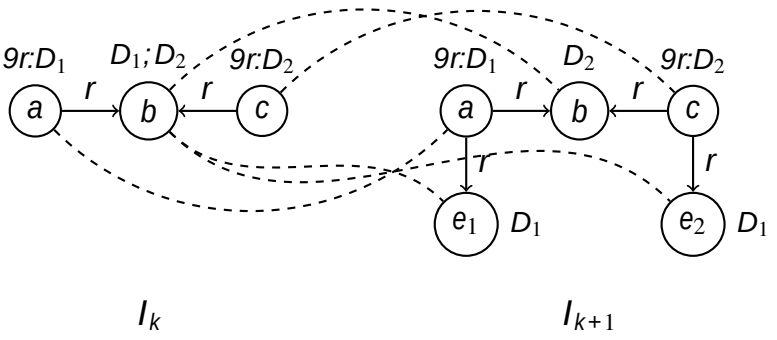

Figure 5.8: Example of the transformation of Case II

of the *Role Propagation* rule).

Denote by P' the predecessors e_{pre} from P such that $e_{pre} \in G_j^{I_k}$ with $2 \leq j \leq n$. The set P' includes all domain elements e_{pre} whose r_1 successors would be in the interpretations of $D_1, \ldots, D_n$ if the transformation in Case I is performed. Since $O^q \models G_1 \sqcap \ldots \sqcap G_n \sqcap \exists r_1.(E \sqcup \bigsqcup_{s=1}^{l} E_s)$, every $p_{pre} \in P'$ must be in the interpretation of $\exists r_1.(E \sqcup \bigsqcup_{s=1}^{l} E_s)$ in I_k. So, there is a child e' of e_{pre} via r_1, such that $e' \in D_i^{I_k}$ where $2 \leq i \leq n$, and $e' \in E^{I_k}$.

We extend I_k to I_{k+1} by applying the following transformations.

1. For every e' identified above, we set $e' \in D_1^{I_{k+1}}$.
2. For every $e_{pre} \in P$ and $e_{pre} \notin P'$ we apply the transformation described in Case I.
3. $D_1^{I_{k+1}} := D_1^{I_k} \setminus \{e\}$.

The argument from Case I shows that $I_{k+1} \models (O^q, A)$. Moreover, we see that I_{k+1} and I_k are bisimilar with respect to $sig(O) \cap \mathbf{F}$ and $ind(A)$. The relation R defined by (5.52) is the bisimulation relation between I_{k+1} and I_k.

Cases IV and V: $e \notin ind(A)$:

When e is not an individual from $ind(A)$, we have $r_j = r_1$ for every j where $1 \leq j \leq n$ because I_k is a pseudo forest. Also, since e has only one predecessor e_{pre}, the sets P and P' would contain only e_{pre}. We use the transformation discussed for Case III in Case IV, and the transformation discussed for Case II in Case V. □

We have now constructed a model J of O^q which is bisimilar to I with respect to $sig(O) \cap \mathbf{F}$ and $ind(A)$ such that no domain element $e \in \Delta^J$ satisfying $e \in D_1^J \sqcap \ldots \sqcap D_n^J \sqcap (\neg E)^J$. The last property of J means that the clause $\neg D_1 \sqcup \ldots \sqcup \neg D_n \sqcup E \in \Delta^q$ is true in J. If this clause is the only clause in Δ^q then J is a model of O^{rp} because

$\mathcal{O}^{rp} = \mathcal{O}^q \cup \mathsf{D}^q$. This gives the contradiction noted before Lemma 5.3.16 and implies that $\mathcal{O}^{rp}$ and $\mathcal{O}^q$ are query inseparable. The same result is reachable if D^q contains several clauses as argued in the discussion before Lemma 5.3.16. This completes the proof of Theorem 5.2.4.

5.4 Eliminating Definers

In Section 5.2 we extracted from the intermediate ontology $\mathcal{O}^{int}$ two ontologies, a query reduced ontology $\mathcal{O}^q$, and the ontology D^q. There are two properties satisfied by $\mathcal{O}^q$.

1. It does not use symbols from the forgetting signature $\mathbf{F}$.
2. It is query inseparable from $\mathcal{O}^{int}$ with respect to the non-forgotten signature $sig(\mathcal{O}) \setminus \mathbf{F}$.

Because $\mathcal{O}^q$ may use definers, it can be seen as a representation or an approximation to the query forgetting view of the input ontology $\mathcal{O}$ with respect to the forgetting signature $\mathbf{F}$. Three types of definers may occur in $\mathcal{O}^q$.

1. Cyclic definers.
2. Simple definers.
3. Complex definers.

Recall from Section 4.5, a definers is cyclic if it occurs both positively and negatively in a single non-tautological clause. Cyclic definers indicate syntactic cycles in the input ontology over some of the forgetting symbols. We saw in Section 4.5 that elimination of cyclic definers may in some cases produce an infinite forgetting view, and deciding the existence of a finite forgetting view has double exponential time complexity. Therefore, we do not eliminate cyclic definers to for practical concerns. For a more detailed discussion, see Section 4.6.

Simple definers are those which can be eliminated without relying on languages with higher complexity. They are simple because they can be eliminated using syntactic transformations. On the contrary, complex definers require a more complex language. They are complex because reasoning is required in order to eliminate them.

Our aim in this section is to eliminate simple and complex definers, such that query inseparability from the input ontology $\mathcal{O}$ with respect to $sig(\mathcal{O}) \setminus \mathbf{F}$ is preserved. We eliminate simple definers first, and complex definers second. Two representations of

the query forgetting view will be obtained by eliminating simple and complex definers. The first is an $\mathcal{ALC}$ representation which is computed by eliminating simple definers. The second is an $\mathcal{ALCI}$ representation which is computed by eliminating simple and complex definers.

5.4.1 Simple and Complex Definers

Let us first characterize simple and complex definers.

Recall that $\mathcal{O}^q$ does not contain clauses which use two or more negative definers where at least one of them is existential. We can categorize the remaining clauses that contain negative definers in two groups. The first group consists of the clauses containing only one negative definer. Let us call this group *Type 1 definition clauses*. The second group is the set D^u of clauses which contain two or more negative universal definers. Complex and simple definers are defined as follows.

Definition 5.4.1. *We say D is a* complex definer*, if D is not a cyclic definer, and one of the following is true about D.*

1. *D occurs with negative polarity in a clause from the set D^u; or*
2. *There is a clause of the following form in $\mathcal{O}^q$*

$$C \sqcap \neg D \sqcap \forall r_1.D_1 \sqcap \cdots \sqcap \forall r_n.D_n \tag{5.53}$$

where $D_1, D_2, \ldots, D_n$ are complex definers.

We use N_d^+ to denote the set of complex definers.

Definition 5.4.2. *Let D be a non-cyclic definer. We say D is a* simple definer *if it occurs negatively only in Type 1 definition clauses, and 2 from Definition 5.4.1 does not apply for D.*

Example 5.4.3. *Consider the following ontology $\mathcal{O}$.*

$$A \sqsubseteq \exists r.C \sqcup \exists r.E \tag{5.54}$$

$$A \sqcap C \sqsubseteq \forall r.B \sqcap \forall r.\neg B \tag{5.55}$$

$$E \sqsubseteq F \tag{5.56}$$

Let $F = \{B, C, E\}$ be a forgetting signature. The query reduced ontology O^q consists of the following clauses.

$$: A \sqcup \exists r.D_1 \tag{5.57}$$

$$: A \sqcup \exists r.D_2 \tag{5.58}$$

$$: A \sqcup : D_1 \sqcup \forall r.D_3 \sqcup \forall r.D_4 \tag{5.59}$$

$$: D_3 \sqcup : D_4 \tag{5.60}$$

$$: D_2 \sqcup F \tag{5.61}$$

where D_1, D_2, D_3 and D_4 are definers. D_1 and D_2 are existential definers. D_3 and D_4 are universal definers. The set D^u consists of the clause (5.60). The set N_d^+ is the set containing D_3 and D_4, because they occur negatively in (5.60). N_d^+ also contains the definer D_1, because it occurs negatively in (5.59). The clause (5.61) is a Type 1 definition clause because it contains only one negative definer. The definer D_2 occurs negatively only in (5.61). Furthermore, none of the definers D_1, D_3, and D_4 occur positively in (5.61). Therefore, D_2 is a simple definer.

There is an interesting remark about Definition 5.4.1. The complex definers satisfying Condition 1 are necessarily universal, whereas the complex definers satisfying Condition 2 can be existential or universal. An existential complex definer is not eliminated with the same elimination process as universal complex definers. We group them under the same umbrella, however, because the technical proofs of their elimination processes are closely related.

5.4.2 Eliminating Simple Definers

First, we eliminate simple definers. We reuse the definer elimination process from Section 4.5 to eliminate simple definers. The input to this process is the ontology O^q. The output is an $\mathcal{ALC}$ representation $V^q_{\mathcal{ALC}}$ of the query forgetting views of the input ontology O with respect to the forgetting signature F.

Figure 5.9 shows the calculus rules used to eliminate simple definers. The rules in Figure 5.9 are very similar to the rules in Figure 4.8. The only difference is the *Non-cyclic Definer Elimination* rule, which is adapted to take into account the definition of simple definers.

The elimination process of simple definers follows Algorithm 6. The algorithm

Non-cyclic Definer Elimination

$$\frac{\mathcal{O} \cup \{ f : D \sqcap C_1; \ldots : D \sqcap C_n g}{\mathcal{O}[D/C]}$$

where $C = \bigsqcup_{i=1}^{n} C_i$, $\mathcal{O}$ does not contain D negatively, C does not contain a negative definer, $D \in sig(C)$, and $sig(C) \setminus N_d^+ = \emptyset$.

Positive Definer Purification

$$\frac{\mathcal{O}}{\mathcal{O}[D/\top]}$$

where $D \in N_d$ is a definer symbol, and D does not occur negatively in $\mathcal{O}$.

Negative Definer Purification

$$\frac{\mathcal{O}}{\mathcal{O}[D/?]}$$

where $D \in N_d$ is a definer symbol, and D does not occur positively in $\mathcal{O}$.

Figure 5.9: Simple definer elimination rules

iterates over simple definers, and eliminates them using the positive and negative definer purification rules, if possible. If a definer occurs both positively and negatively in $\mathcal{V}_{ALC}^{q}$, we eliminate it using the *Non-cyclic Definer Elimination* rule.

The following example explains the elimination of a simple definer, and the extraction of $\mathcal{V}_{ALC}^{q}$ from $\mathcal{O}^{q}$.

Example 5.4.4. *Consider $\mathcal{O}^{q}$ from Example 5.4.3. There was one simple definer used in $\mathcal{O}^{q}$, namely D_2. The definer D_2 occurred negatively only in the clause $\neg D_2 \sqcup F$, and positively only in (5.58). We eliminate D_2 by replacing it with F in (5.58). The ontology $\mathcal{V}_{ALC}^{q}$ is therefore the ontology consisting of the clauses (5.57), (5.59), (5.60), and the following clause.*

$$\neg A \sqcup \exists r.F \tag{5.62}$$

The Non-cyclic Definer Elimination rule in Figure 5.10 is almost identical to the one in Figure 4.8. However, the last side condition $sig(C) \setminus N_d^+ = \emptyset$ is new. It is added

Algorithm 6: Algorithm for eliminating simple de ners

Input: Ontology $\mathcal{O}^q$, set of de ners N_d, set of de ners N_d^+
Output: Ontology $\mathcal{V}^q_{ALC}$ where simple de ners are eliminated
Initialize: $\mathcal{V}^q_{ALC} := \mathcal{O}^q$, $C = \top$

```
for simple de ner D ∈ N_d ∩ N_d^+ do
    if D occurs only positively in V^q_ALC then
        // Replace every occurrence of D in V^q_ALC with ⊤
        V^q_ALC := V^q_ALC[D ⇒ ⊤]
    else if D occurs only negatively in V^q_ALC then
        // Replace every occurrence of D in V^q_ALC with ⊥
        V^q_ALC := V^q_ALC[D ⇒ ⊥]
    else
        // Perform the Non-cyclic Definer Elimination rule
        for clause C ∈ V^q_ALC do
            if C has form: D ⊔ E, and D ∉ sig(E) then
                C := C ⊓ E
        if sig(C) \ N_d^+ = ∅ and D ∉ sig(C) then
            V^q_ALC := V^q_ALC[D ⇒ C]
        C = ⊤
```

Return: $\mathcal{V}^q_{ALC}$

to prevent eliminating complex de ners satisfying Condition 2 in De nition 5.4.1. Although ignoring this condition allows for eliminating more de ners, the obtained ontology would not be ready for eliminating complex de ners. So, we respect this condition only if complex de ners will be eliminated in a later step. To explain, consider the following ontology $\mathcal{O}$.

$$A \sqsubseteq \forall r.\forall r.(B \sqcap (\forall r.C \sqcup \forall r.\top \sqsubseteq C)) \tag{5.63}$$

$$B \sqsubseteq E \tag{5.64}$$

Let $\Gamma = \{B, C\}$ be a forgetting signature. The query reduced ontology $\mathcal{O}^q_\Gamma$ of $\mathcal{O}$ with respect to Γ is the following ontology.

$$\top \sqsubseteq A \sqcup \forall r.D_1 \tag{5.65}$$

$$\top \sqsubseteq D_1 \sqcup \forall r.D_2 \tag{5.66}$$

$$\top \sqsubseteq D_2 \sqcup E \tag{5.67}$$

$$\top \sqsubseteq D_2 \sqcup \forall r.D_3 \sqcup \forall r.D_4 \tag{5.68}$$

$$: D_3 \sqcap : D_4 \tag{5.69}$$

The set $\mathcal{D}^u$ consists of the clause (5.69). So D_3 and D_4 are complex de ners because they occur negatively in (5.69). The de ner D_2 is complex because it occurs negatively in (5.68). Furthermore, the de ner D_1 is also complex because it occurs negatively in the clause (5.66) which uses the complex de ner D_2 in the concept $\forall r.D_2$.

Consider applying the Non-cyclic De ner Elimination rule in Figure 5.10, and ignore the last side condition of the rule. The output would be the following ontology which uses less de ners.

$$: A \sqcap \forall r.(\forall r.(E \sqcup (\forall r.D_3 \sqcap \forall r.D_4))) \tag{5.70}$$

$$: D_3 \sqcap : D_4 \tag{5.71}$$

Although the ontology obtained when the last side condition of the Non-cyclic De ner Elimination rule uses less de ners, it is not in a ready state for eliminating the complex de ners D_3 and D_4. For instance, to represent the predecessors of elements in the extensions of $D_3 \sqcap D_4$, it is easy to use the concept $\forall r.(: D_2)$ derived from (5.68), but it is not easy to derive a representation from the two clauses above.

Lemma 5.4.5. Let $\mathcal{O}$ be an ontology, and $\mathbf{F}$ a forgetting signature. Suppose $\mathcal{O}^q$ is a query reduced ontology obtained from $\mathcal{O}$ as shown before, and $\mathcal{V}^q_{ALC}$ be the ontology obtained from $\mathcal{O}^q$ by applying the Non-cyclic de ner elimination rule in Figure 5.10 exhaustively. Then $\mathcal{V}^q_{ALC}$ and $\mathcal{O}$ are query inseparable with respect to $sig(\mathcal{O}) \cap \mathbf{F}$.

Proof. The proof can be constructed from the following.

1. It can be seen from the proof of Theorem 4.5.5 that $\mathcal{O}^q$ and $\mathcal{V}^q_{ALC}$ are model inseparable with respect to $sig(\mathcal{O}) \cap \mathbf{F}$.

2. The result in Item 1 and Lemma 3.5.10 imply that $\mathcal{O}^q$ and $\mathcal{V}^q_{ALC}$ are query inseparable with respect to $sig(\mathcal{O}) \cap \mathbf{F}$.

3. From Theorem 5.2.4 we know that $\mathcal{O}$ and $\mathcal{O}^q$ are query inseparable with respect to the non-forgotten signature $sig(\mathcal{O}) \cap \mathbf{F}$.

4. $\mathcal{O} \overset{Q}{\equiv}_{sig(\mathcal{O}) \cap \mathbf{F}} \mathcal{V}^q_{ALC}$ follows from the two previous points and Lemma 3.5.7.

□

5.4.3 Eliminating Complex De ners

The second step is eliminating complex de ners.

There are two types, Type 1 and Type 2 complex de ners. Prioritizing the elimination of Type 1 complex de ners can simplify the elimination of Type 2 complex de ners. The two types are de ned as follows.

De nition 5.4.6. Let D_1 be a complex de ner. We say D is a *Type 2 complex de ner* if and only if there is a clause of the following form

$$C \sqcup \forall r_1.D_1 \sqcup \forall r_2.D_2 \sqcup \cdots \sqcup \forall r_n.D_n \tag{5.72}$$

where C is a concept, $r_1, \ldots, r_n \in N_r$ are role names, $D, D_1, D_2, \ldots, D_n$ are complex de ners, $n \geq 2$, and one of the following is true.

1. D is a de ner from the set $D_1, \ldots, D_n$, or
2. D occurs with negative polarity in C.

A complex de ner that is not Type 2, is said to be a *Type 1 complex de ner*.

We use the rules from Figure 5.10 to eliminate the complex de ners. The *Lower Bound Extraction* rule, the *LB* rule for short, computes clauses where a complex de ner occurs positively as a top level literal. In axiom form, the conclusion of a LB inference can be represented as $E \sqcup D_i$, where D_i is a de ner, and E is the negation of the concept $\forall r_i.(C \sqcup \bigsqcup_{j=1, j\neq i}^{n} \forall r_j.C_j)$. The concept E can be seen as a lower bound on D_i because a domain element in the extension of E is necessarily in the extension of D_i.

The first premise of a LB inference is a clause where the complex de ners $D_1, \ldots, D_n$ occur with positive polarity below universal role restrictions. We require the concept C to be free of de ners. This restriction on C is important because $\forall^q_{ALC}$ may contain a clause of the following form.

$$\neg D \sqcup \forall r_1.D_1 \sqcup \cdots \sqcup \forall r_n.D_n \tag{5.73}$$

where $D, D_1, \ldots, D_n$ are complex de ners. For instance, the first representation of $\forall^q_{ALC}$ obtained in the example before Lemma 5.4.5 contained the clause (5.68) which has the form of (5.73). When C contains a de ner D', the restriction on C becomes a blocking constraint forcing either the elimination of D' if it is universal, or performing a set of inferences such that $D_1, \ldots, D_n$ are no longer complex de ners. We will examine the first case in Example 5.4.10, and the second case in Example 5.4.11.

Lower Bound Extraction (LB)

$$\frac{C \sqsubseteq \forall r_1.D_1 \sqcap \ldots \sqcap \forall r_n.D_n, \quad \bigsqcup_{k=1,k\neq i}^{n} \{\, D_k \sqcap C_k \}}{D_i \sqsubseteq \forall r_i.(C \sqcap \bigsqcap_{j=1,j\neq i}^{n} \forall r_j.C_j)}$$

where $1 \leq i \leq n$, $n \geq 1$, $\{D_1, \ldots, D_n\} \subseteq N_d^+$, D_i does not occur in C, and negative D_i does not occur in D_k.

Upper Bound Extraction (UB)

$$\frac{C_1 \sqsubseteq \forall r_1.D \sqcap \forall r_2.D', \quad C_2 \sqsubseteq \forall s_1.D_1 \sqcap \ldots \sqcap \forall s_m.D_m, \quad \bigsqcup_{j=1}^{m} \neg D \sqcap \neg D_j \sqcap E_j}{\neg D \sqcap \forall s_1.\neg C_2 \sqcap \ldots \sqcap \forall s_m.\neg C_2 \sqcap E_1 \sqcap \ldots \sqcap E_m}$$

where $D, D', D_1, \ldots, D_m$ are complex Type 2 de ners, $m \geq 2$, negative D does not occur in C_1, and D does not occur in C_2.

Relaxing

$$\frac{C \sqsubseteq \forall r_1.D_1 \sqcap \ldots \sqcap \forall r_n.D_n, \quad \neg D_1 \sqcap E}{C \sqsubseteq \forall r_1.E \sqcap \forall r_2.D_2 \sqcap \ldots \sqcap \forall r_n.D_n}$$

where C and E are concepts that do not use de ners.

Figure 5.10: Complex de ner elimination rules

The second premise of a LB inference is a set of $n-1$ clauses of the form $\neg D_k \sqcap C_k$ where $1 \leq k \leq n$ and $k \neq i$. Here, i is the subscript of the de ner in the first disjunct of the conclusion. The excluded clause $\neg D_i \sqcap C_i$ is not needed to compute the conclusion, and the information that it brings is obtainable anyway when the clauses is resolved with the conclusion on $\neg D_i$. When several clauses $\neg D_k \sqcap C_k^l$ are present for one value of k, where $l \geq 2$, we combine these clauses as in (5.74).

$$\neg D_k \sqcap \Big(\bigsqcup_{j=1}^{l} C_k^j\Big) \tag{5.74}$$

Clauses from the set D^u where two or more negative de ners occur, cannot be used in the second premise. This restriction prevents negative de ners from occurring below universal role restriction in the conclusion of the rule. If for some D_k, with $1 \leq k \leq n$, no clause with the required form exists, or if D_k occurs negatively only in clauses from D^u, we use the clause $\neg D_k \sqcap \top$.

The conclusion of a LB inference is a clause where the complex de ner D_i occurs positively as a top level literal, and $1 \leq i \leq n$. The second disjunct in the conclusion

is an $\mathcal{ALCI}$ concept because the inverse role r_i^- is used. The conclusion of a LB inference is best understood when the premises and the conclusion are in axiom form. Suppose we set $n = 1$.

$$\neg C \sqsubseteq \forall r_1.D_1 \sqcup \cdots \sqcup \forall r_n.D_n \tag{5.75}$$

$$\bigsqcup_{i=2}^{n} \{D_i \sqsubseteq C_i\} \tag{5.76}$$

$$\exists r_1.(\neg C \sqcup \bigsqcup_{i=2}^{n} \exists r_i.\top \sqsubseteq C_i) \sqsubseteq D_1 \tag{5.77}$$

The axiom (5.75) is the axiom form of the first premise. (5.76) is the axiom form of the second premise, and (5.77) is the axiom form of the conclusion. Observe that we ignored the clause $D_1 \sqsubseteq C_1$ from (5.76) because it is not required to compute the conclusion.

Suppose $\mathcal{I}$ is an interpretation, and a domain element e is in the interpretation of the left hand side of (5.77). Then there is a predecessor of e through r_1. That is, there is a domain element d such that $(d, e) \in r_1^{\mathcal{I}}$. Furthermore, the following is true about d:

1. d is not in the interpretation of C.
2. For each i where $2 \leq i \leq n$, d has a successor e_i through r_i, and e_i is not in the interpretation of C_i.

From Item 1 above and (5.75) we know that there is j where $1 \leq j \leq n$ where every r_j successor of d is in the interpretation of D_j. From Item 2 and (5.76) we know that j is not in the range from 2 to n. Therefore, d is in the interpretation of $\forall r_1.D_1$. Since e is a r_1 successor of d, we get that e is in the interpretation of D_1.

Another rule in Figure 5.10 is the *Upper Bound Extraction* rule, or the *UB* rule for short. In axiom form, the conclusion of the UB inferences have the form $C \sqsubseteq D$, where the definer D is on the left hand side of the subsumption, and C is the concept $\forall s_1.C_2 \sqcup \cdots \sqcup \forall s_m.C_2 \sqcup E_1 \sqcup \cdots \sqcup E_m$. The concept C can be seen as an upper bound on D, because if a domain element is in the extension of D then it is also in the extension of C, and if it is not in the extension of C, then it is not in the extension of D. The UB rule takes three premises. The first and second premises are almost identical to the first premise of the *Lower Bound Extraction* rule. The difference is that the two premises use at least two complex definers below universal role restrictions. The third premise is a set of m clauses. Each clause in the third premise uses, with negative polarity, the

de ner D from the first premise, and a de ner from the second premise. Suppose that D is a de ner occurring below universal role restriction in the first premise, and D_1 a de ner occurring below universal role restriction in the second premise. If several clauses of the form $\neg D \sqcup \neg D_1 \sqcup C_i$ are present, we combine them as we did for the second premise of the Lower Bound Extraction rule. The combined clause is used in the third premise instead of the original ones.

The conclusion of a UB inference is best explained through the axiom form of the premises and conclusion.

$$\neg C_1 \sqcup \forall r_1.D \sqcup \forall r_2.D' \tag{5.78}$$

$$\neg C_2 \sqcup \forall s_1.D_1 \sqcup \cdots \sqcup \forall s_m.D_m \tag{5.79}$$

$$\bigcup_{j=1}^{m} \{D \sqcup D_j \sqcup E_j\} \tag{5.80}$$

$$D \sqcup \exists s_1.C_2 \sqcup \cdots \sqcup \exists s_m.C_2 \sqcup E_1 \sqcup \cdots \sqcup E_m \tag{5.81}$$

(5.78) is the axiom form of the first premise, (5.79) is the axiom form of the second premise, (5.80) is the axiom form of the third premise, and (5.81) is the axiom form of the conclusion. Suppose $\mathcal{I}$ is a model, and $d \in \Delta^{\mathcal{I}}$ is a domain element. Assume d is in the interpretation of the concept on the left hand side of the subsumption in (5.81). From the conjuncts $\exists s_i.C_2$ with $1 \leq i \leq m$, there must exist a domain element $e \in \Delta^{\mathcal{I}}$ such that $d \notin C_2^{\mathcal{I}}$ and $(d,e) \in s_i^{\mathcal{I}}$. From (5.79) we know that e is in the interpretation of one of $D_1, D_2, \ldots,$ or D_m. Since additionally $e \in \Delta^{\mathcal{I}}$, we get from (5.80) that $e \in (E_1 \sqcup E_2 \sqcup \cdots \sqcup E_m)^{\mathcal{I}}$, which is expressed by the right hand side of the subsumption in (5.81).

The Relaxing rule has two premises. The first premise has the same form of the first premise in the LB rule, and the second premise is a clause $D_1 \sqcup E$. The conclusion of the rule substitutes E for D_1 in the first premise. When there are several clauses $\neg D_1 \sqcup E_i$ where $i \geq 2$, we combine these clauses and use the combined clause as the second premise. Note that we require that C and E do not use de ner symbols.

The first step of the de ner elimination process, is the elimination of Type 1 complex de ners. Type 1 complex de ners are eliminated using Algorithm 7. Line 1 in Algorithm 7 iterates over the Type 1 complex de ners. Line 2 iterates over clauses C which use D below universal role restriction. Line 4 performs a LB inference using C as the first premise. The clause C is removed from $\mathcal{V}$ at line 5. When all clauses that use D positively below universal role restriction have been processed in LB inferences,

Algorithm 1 from Chapter 4 is invoked at Line 6. The algorithm resolves exhaustively the conclusions of the LB inferences with the clauses that use D negatively, and then removes the clauses where D occurs. The elimination of Type 1 de ners may make some other de ners simple. So, at line 7, after the elimination of Type 1 de ners have been completed, we call Algorithm 6 to eliminate any simple de ners present in $\mathcal{V}$.

Algorithm 7: Algorithm for eliminating complex de ners of Type 1

Input: Ontology $\mathcal{O}$, set of de ners $\mathbf{N}$
Output: Ontology $\mathcal{V}$ where Type 1 de ners have been eliminated
Initialize: $\mathcal{V} := \mathcal{O}$

```
1 for Type 1 de ner D ∈ N do
2     for Clause C ∈ O of the form C' ∨ r:D do
3         if no de ner occurs in C' then
              // Perform the LB rule in Figure 5.10.
4             V := V ∪ LB(C, D)
              // Remove the clause C.
5             V := V \ {C}
      // Eliminate D using Algorithm 1.
6     V := Algorithm 1(V, {D})
  // Eliminate Simple Definers.
7 V := Eliminate_Simple_De ners(V, N)
  Return: V
```

We will show later in Lemma 5.4.18 that the elimination of Type 1 complex de ners preserves model inseparability with respect to $\mathrm{sig}(\mathcal{O}) \cap \mathbf{F}$. For this reason, the elimination of Type 1 de ners is prioritized and eagerly performed. We will see that as Type 2 de ners are being eliminated, some Type 2 de ners become Type 1.

The second step eliminates Type 2 de ners. The main idea is to eliminate the clauses of $D^{\sqcup}$ that remain after the elimination of Type 1 de ners. When these clauses are eliminated the de ners occurring in the ontology become simple de ners that can be eliminated using Algorithm 6. A general process for eliminating Type 2 de ners can be described as follows.

1. We perform UB inferences to obtain Type 1 de nition clauses, where de ners occur as top level negated literals.

2. When all possible UB inferences have been performed, we use the computed Type 1 de nition clauses in LB inferences to obtain clauses where de ners occur as top level positive disjuncts.

3. We resolve the clauses that use de ners positively with the clauses that use them negatively using the binary resolution rule in Figure 4.3. When resolution has been performed, the clauses used as premises to resolution inferences are removed. This step in the process may eliminate clauses from $\mathcal{D}^u$ as they would be resolved with the conclusions of the LB inferences.

4. The elimination of clauses from $\mathcal{D}^u$ may consequently convert some de ners to Type 1 or simple de ners. If this happens, we eliminate these de ners using Algorithms 7 and 6.

5. At the end of the process, we remove all clauses from $\mathcal{D}^u$ that remain, and eliminate the remaining de ners using the simple de ner elimination process in Algorithm 6.

A dif culty in the above process is that all possible UB conclusions should be computed in order to be used in LB inferences. However, in some cases, UB inferences may not be performed until some de ners have been eliminated in Step 4. Thus, the ordering by which de ners are eliminated matters, and may trigger new LB inferences. Finding the optimal ordering remains an open problem that we did not solve. In other cases, some UB inferences can only be performed when Relaxing inferences have been performed.

We explain the elimination of complex de ners through a series of examples. The following example considers only universal de ners of Type 1, and covers Algorithm 7.

Example 5.4.7. Consider the following ontology $\mathcal{O}$.

$$A_1 \sqsubseteq \forall r.B \tag{5.82}$$

$$A_2 \sqsubseteq \forall s.\neg B \tag{5.83}$$

Let $\mathsf{F} = \{B\}$ be the forgetting signature. The reduced ontology of $\mathcal{O}$ with respect to F is the following ontology.

$$\neg A_1 \sqcup \forall r.D_1 \tag{5.84}$$

$$\neg A_2 \sqcup \forall s.D_2 \tag{5.85}$$

$$\neg D_1 \sqcup \neg D_2 \tag{5.86}$$

where D_1 and D_2 are de ners. The set $\mathcal{D}^u$ consists of the clause (5.86). So, the de ners D_1 and D_2 are complex. From De nition 5.4.6 we see that D_1 and D_2 are Type 1

de ners.

Suppose we eliminate D_1 in the rst iteration of the loop at line 1 in the algorithm. We perform a LB inference to extract a clause where D_1 occurs with positive polarity as a top level literal. The rst premise of the inference is the clause (5.84), and the second premise is $\emptyset$. The conclusion is the following clause.

$$D_1 \sqcup \forall r.\top \sqsubseteq A_1 \tag{5.87}$$

Line 5 of Algorithm 7 removes the clause (5.84). As no more clauses use D_1 below universal role restriction, Algorithm 1 is called. It resolves the clauses (5.87) and (5.86) on D_1, and obtains the following clause.

$$\neg D_2 \sqcup \forall r.\top \sqsubseteq A_1 \tag{5.88}$$

Next, the call to Algorithm 1 removes the clauses (5.86) and (5.87) used in Resolution.

In the second iteration, D_2 is eliminated. A LB inference is performed. The rst premise of the inference is the clause (5.85), and the second premise is the empty set. The conclusion is the following clause.

$$D_2 \sqcup \forall s.\top \sqsubseteq A_2 \tag{5.89}$$

The clause (5.85) is removed. Then, the clauses (5.89) and (5.88) are resolved on D_2 through the call to Algorithm 1. The following conclusion is obtained.

$$\forall r.\top \sqsubseteq A_1 \sqcup \forall s.\top \sqsubseteq A_2 \tag{5.90}$$

Next, the algorithm removes the clauses (5.88) and (5.89).

Since there are no more de ners to eliminate, we obtain the query forgetting view $\mathcal{V}^{q}_{\mathcal{ALCI}}$ of $\mathcal{O}$ with respect to $\mathcal{F}$ as the ontology consisting of the clause (5.90).

In the next example we consider only Type 2 de ners.

Example 5.4.8. Consider the following ontology $\mathcal{O}$.

$$A \sqcup \forall r.B \sqcup \forall s.\top \sqsubseteq B \tag{5.91}$$

Let $\mathcal{F} = \{B\}$ be a forgetting signature. The reduced ontology with respect to $\mathcal{F}$ is

as follows.

$$: A \sqcup \forall r.D_1 \sqcup \forall s.D_2 \tag{5.92}$$

$$: D_1 \sqcup : D_2 \tag{5.93}$$

where D_1 and D_2 are de ners. The set $\mathcal{D}^u$ consists of the clause (5.93). So, D_1 and D_2 are complex de ners. They are Type 2 because they occur together positively in (5.97).

There are not any UB inferences that can be performed. Two LB inferences should be performed to compute positive occurrences of D_1 and D_2. The first premise in the two inferences would be (5.92). The second premise of the first inference is the set $\{: D_2 \sqcup \bot\}$, and the second premise of the second inference is the set $\{: D_1 \sqcup \bot\}$. Observe that in the two sets we ignored the clauses which are not required in the computation of the conclusion. The two sets consist of technical clauses, because D_1 and D_2 occur with negative polarity only in the clause (5.93) which has two negative de ners.

The clause (5.94) below is the conclusion of the first LB inference, and (5.95) is the conclusion of the second. However, the two conclusions are tautologies and can be eliminated.

$$D_1 \sqcup \forall r.(: A \sqcup \forall s.\bot) \tag{5.94}$$

$$D_2 \sqcup \forall s.(: A \sqcup \forall r.\bot) \tag{5.95}$$

Next, the clause (5.93) is removed, which makes D_1 and D_2 simple de ners. The two de ners are puri ed by the simple de ner elimination process, and the empty ontology $\emptyset$ is obtained as the final query forgetting view $\mathcal{V}^q_{\mathcal{ALCI}}$ of $\mathcal{O}$ with respect to $\mathcal{F}$.

There is an important difference between Example 5.4.7 and Example 5.4.8. The interpretations of D_1 and D_2 can be captured in terms of A and r in Example 5.4.7. Below, (5.96) de nes D_1, and (5.97) de nes D_2 in an arbitrary model $\mathcal{I}$.

$$D_1^{\mathcal{I}} = \{y \in \Delta^{\mathcal{I}} \mid \exists x.(x,y) \in r^{\mathcal{I}} \wedge x \in A_1^{\mathcal{I}}\} \tag{5.96}$$

$$D_2^{\mathcal{I}} = \{y \in \Delta^{\mathcal{I}} \mid \exists x.(x,y) \in s^{\mathcal{I}} \wedge x \in A_2^{\mathcal{I}}\} \tag{5.97}$$

Compare (5.96) and (5.97) to the following de nitions of D_1 and D_2 in Example 5.4.8.

$$D_1^{\mathcal{I}} = \{y \in \Delta^{\mathcal{I}} \mid \exists x.(x,y) \in r^{\mathcal{I}} \wedge x \in A^{\mathcal{I}} \wedge y \notin D_2^{\mathcal{I}}\} \tag{5.98}$$

$$D_2^{\mathcal{I}} = \{y \in \Delta^{\mathcal{I}} \mid \exists x.(x,y) \in s^{\mathcal{I}} \wedge x \in A^{\mathcal{I}} \wedge y \notin D_1^{\mathcal{I}}\} \tag{5.99}$$

The two definitions (5.98) and (5.99) of D_1 and D_2 depend on each other. Because of this dependency, the LB inferences in Example 5.4.8 could not obtain information about D_1 and D_2. More information would need be present in the input ontology to obtain non-trivial information about the complex definers. The following example explains this case.

Example 5.4.9. Consider the following ontology $\mathcal{O}$

$$A \sqsubseteq \forall r.B_1 \sqcap \forall r.B_2 \tag{5.100}$$
$$B_1 \sqcap B_2 \sqsubseteq C \tag{5.101}$$
$$B_1 \sqsubseteq \neg G_1 \tag{5.102}$$
$$B_2 \sqsubseteq \neg G_2 \tag{5.103}$$

Let $\mathbf{F} = \{B_1, B_2\}$ be a forgetting signature. The query reduced ontology $\mathcal{O}^{\mathbf{F}}_{\forall}$ of $\mathcal{O}$ with respect to $\mathbf{F}$ is:

$$\top \sqsubseteq A \sqcup \forall r.D_1 \sqcup \forall r.D_2 \tag{5.104}$$
$$\top \sqsubseteq D_1 \sqcup \neg D_2 \sqcup C \tag{5.105}$$
$$\top \sqsubseteq D_1 \sqcup \neg G_1 \tag{5.106}$$
$$\top \sqsubseteq D_2 \sqcup \neg G_2 \tag{5.107}$$

where D_1 and D_2 are definers. The set $\mathcal{D}^u$ consists of the clause (5.105). So, D_1 and D_2 are complex definers. Moreover, they are Type 2 because they occur in (5.104).

No UB inferences can be performed on $\mathcal{O}^{\mathbf{F}}_{\forall}$. We perform two LB inferences to obtain clauses where D_1 and D_2 occur with positive polarity as top level literals.

The first premise of the two inferences is the clause (5.104). The second premise of the first inference is the set $\{D_2 \sqsubseteq \neg G_2\}$, and the second premise of the second inference is the set $\{D_1 \sqsubseteq \neg G_1\}$. The conclusion of the first inference is the clause (5.108) below. The conclusion of the second inference is the clause (5.109).

$$D_1 \sqsubseteq \forall r.(\neg A \sqcup \forall r.\neg G_2) \tag{5.108}$$
$$D_2 \sqsubseteq \forall r.(\neg A \sqcup \forall r.\neg G_1) \tag{5.109}$$

We perform resolution exhaustively on the clauses that use D_1 and D_2. The following clauses would be obtained.

$$\neg D_2(r) \vee G_1(r) \tag{5.110}$$

$$\neg D_1(r) \vee G_2(r) \tag{5.111}$$

$$\neg D_2(r) \vee \neg D_1(r) \vee C \tag{5.112}$$

Then, the clauses (5.105), (5.106), (5.107), (5.108), and (5.109) which have been used in resolution inferences are removed. The definitions of D_1 and D_2 remain used only in the clause (5.104). The two definitions are purified by the simple definition elimination process, which substitutes the clause $\neg D_1(r) \vee \neg D_2(r)$ for (5.104). The new clause is removed because it is a tautology.

The query forgetting view of $\mathcal{O}$ with respect to $\mathbf{F}$ would be the ontology comprising the clauses (5.110), (5.111), and (5.112).

In the above example, the presence of the axioms (5.102) and (5.103) allows for inferring non-trivial information about D_1 and D_2 when the interpretations of G_1 and G_2 are known. For instance, suppose we have the following ABox

$$A(a_1) \tag{5.113}$$

$$A(a_2) \tag{5.114}$$

$$r(a_1, b_1) \tag{5.115}$$

$$r(a_2, b_2) \tag{5.116}$$

$$G_2(b_1) \tag{5.117}$$

$$G_1(b_2) \tag{5.118}$$

$$r(a_1, b) \tag{5.119}$$

$$r(a_2, b) \tag{5.120}$$

Recall the ontology $\mathcal{O}$ and forgetting view $V^q_{\mathcal{ALCI}}$ from Example 5.4.9. A visualization of the knowledge base $(\mathcal{O}, \mathcal{A})$ is in Figure 5.11. Suppose we have the query $q(x) = C(x)$. As shown in the figure, we have that $(\mathcal{O}, \mathcal{A}) \models C(b)$. The same answer can be obtained from the knowledge base $\mathcal{K} = (V^q_{\mathcal{ALCI}}, \mathcal{A})$ as follows. First, we have $\mathcal{A} \models (A \sqcap \exists r.G_2)(a_1)$, because $\mathcal{A} \models r(a_1, b_1)$ and $\mathcal{A} \models G_2(b_1)$. Hence $\mathcal{A} \models \exists r.(A \sqcap \exists r.G_2)(b)$ because $\mathcal{A} \models r(a_1, b)$. Second, we have $\mathcal{A} \models (A \sqcap \exists r.G_1)(a_2)$, because $\mathcal{A} \models r(a_2, b_2)$ and $\mathcal{A} \models G_1(b_2)$. Hence $\mathcal{A} \models \exists r.(A \sqcap \exists r.G_1)(b)$ because $\mathcal{A} \models$

Figure 5.11: Visualization of the knowledge base $(\mathcal{O}_3, \mathcal{A})$. Small letters with subscripts represent domain elements. Circles labeled with big letters with subscripts represent concept names. Two elements d_1 and d_2 are connected by an arrow if $(\mathcal{O}_3, \mathcal{A}) \models r(d_1, d_2)$.

(a_2, b). The two conclusions and (5.117) imply that $(\mathcal{O}_{ALCI}^{q}, \mathcal{A}) \models C(b)$.

The two clauses (5.110) and (5.111) are logically equivalent, because each of them is logically equivalent to the following clause (see the proof of Lemma 5.4.17).

$$\top \sqsubseteq \exists r.G_1 \sqcap \forall r.G_2 \tag{5.121}$$

A LB inference can be blocked if a negative definer occurs in the first premise. If this definer is universal, then it must be eliminated before the LB inference can be performed. We discuss this situation in the following example.

Example 5.4.10. Consider the following ontology $\mathcal{O}$.

$$A \sqsubseteq \forall r.(E \sqcap \forall r.B_1 \sqcap \forall r.B_2) \tag{5.122}$$

$$B_1 \sqsubseteq \neg B_2 \tag{5.123}$$

$$B_1 \sqsubseteq G_1 \tag{5.124}$$

$$B_2 \sqsubseteq G_2 \tag{5.125}$$

Let $\mathbf{F} = \{B_1, B_2\}$ be a forgetting signature. The query reduced ontology of $\mathcal{O}$ with respect to $\mathbf{F}$ is as follows.

$$\top \sqsubseteq \exists r.D_1 \tag{5.126}$$

$$: D_1 \sqcup \exists r{:}D_2 \sqcup \forall r{:}D_3 \tag{5.127}$$

$$: D_2 \sqcup \neg D_3 \tag{5.128}$$

$$: D_2 \sqcup G_1 \tag{5.129}$$

$$: D_3 \sqcup G_2 \tag{5.130}$$

where D_1, D_2 and D_3 are de ners. The set D^u is a singleton set consisting of the clause (5.128). So, D_2 and D_3 are complex de ners. The de ner D_1 is complex because it occurs negatively in the clause (5.127) where as D_2 and D_3 occur positively. We can also see that D_1; D_2 and D_3 are Type 2 de ners.

The clause (5.127) cannot be used in a LB inference because it contains the de ner D_1 with negative polarity. So, D_1 needs to be eliminated before D_2 and D_3. We perform a LB inference with (5.126) as the first premise, and the empty set as the second premise. The conclusion of this inference is the following clause.

$$D_1 \sqsubseteq \forall r :: A \tag{5.131}$$

We then resolve the clauses where D_1 occurs. The following clause is consequently computed.

$$\exists r :: A \sqcup \forall r{:}D_2 \sqcup \forall r{:}D_3 \tag{5.132}$$

Finally, we eliminate the clauses (5.127), and (5.131) that were premises to the resolution inference which computed (5.132). D_1 now occurs only positively in (5.126), we purify it by replacing it with $\bot$, and remove the clause (5.126) as it becomes a tautology. The remaining clauses are (5.132), (5.128), (5.129), and (5.130).

We perform two LB inferences to obtain clauses where D_2 and D_3 occur as top level literals. The first premise in the two inferences is the clause (5.132). The second premise of the first inference is the clause (5.130), and the second premise of the second inference is the clause (5.129). The two conclusions of the two inferences are the clauses (5.133) and (5.134) below.

$$D_2 \sqsubseteq \forall r :: (\exists r :: A \sqcup \forall r{:}G_2) \tag{5.133}$$

$$D_3 \sqsubseteq \forall r :: (\exists r :: A \sqcup \forall r{:}G_1) \tag{5.134}$$

We next resolve the clauses where D_2 and D_3 occur, and obtain the following clauses.

$$\forall r.(\exists t.\forall r.A :: \forall r.G_2) \sqcup \forall r.(\exists t.\forall r.A :: \forall r.G_1) \tag{5.135}$$

$$G_1 \sqcup \forall r.(\exists t.\forall r.A :: \forall r.G_2) \tag{5.136}$$

$$G_2 \sqcup \forall r.(\exists t.\forall r.A :: \forall r.G_1) \tag{5.137}$$

Finally, the clauses using D_2 and D_3 are removed. The query forgetting view of $\mathcal{O}$ with respect to $\mathbf{F}$ is the ontology consisting of the three clauses (5.135), (5.136), and (5.137).

Another case that blocks LB inferences, is the presence of a negative existential definer in the first premise of the inference. The following example discusses this case.

Example 5.4.11. Consider the following ontology $\mathcal{O}$.

$$A \sqsubseteq \exists r.(\exists t.\forall r.B_1 \sqcup \forall r.\neg B_2) \tag{5.138}$$

$$B_1 \sqsubseteq \neg B_2 \tag{5.139}$$

$$B_1 \sqsubseteq G_1 \tag{5.140}$$

$$B_2 \sqsubseteq G_2 \tag{5.141}$$

Let $\mathbf{F} = \{B_1, B_2\}$ be a forgetting signature. The query reduced ontology of $\mathcal{O}$ with respect to $\mathbf{F}$ is as follows.

$$\top \sqsubseteq \forall.\exists r.D_1 \tag{5.142}$$

$$\top \sqsubseteq D_1 \sqcup \exists t.\forall r.D_2 \sqcup \forall r.D_3 \tag{5.143}$$

$$\top \sqsubseteq D_2 \sqcup \neg D_3 \tag{5.144}$$

$$\top \sqsubseteq D_2 \sqcup G_1 \tag{5.145}$$

$$\top \sqsubseteq D_3 \sqcup G_2 \tag{5.146}$$

where D_1, D_2 and D_3 are definers. The set $\mathcal{D}^u$ is a singleton set consisting of the clause (5.144). So, D_2 and D_3 are complex definers. The definer D_1 is complex because it occurs negatively in the clause (5.143) where D_2 and D_3 occur positively. Furthermore, D_1, D_2, and D_3 are Type 2.

The clause (5.142) cannot be used in a LB inference, because D_1 is existential. Furthermore, the clause (5.143) cannot be used in LB inferences, because it uses the

existential de ner D_2 with negative polarity.

We remove the clause (5.144). This makes the de ners D_1, D_2, and D_3 simple de ners. We eliminate them using the simple de ner elimination process, and obtain the following query forgetting view $\mathcal{V}^q_{\mathcal{ALCI}}$ of $\mathcal{O}$ with respect to $\mathbf{F}$.

$$: A \sqcup \exists r.(\forall r.G_1 \sqcup \forall r.G_2) \tag{5.147}$$

In the previous examples, UB inferences were not possible. The next two examples perform UB inferences.

Example 5.4.12. Consider the following ontology $\mathcal{O}$.

$$A_1 \sqsubseteq \forall r.B_1 \sqcup \forall r.B_2 \tag{5.148}$$
$$A_2 \sqsubseteq \forall r.B_3 \sqcup \forall r.B_4 \tag{5.149}$$
$$B_1 \sqcap B_3 \sqsubseteq C_1 \tag{5.150}$$
$$B_1 \sqcap B_4 \sqsubseteq C_2 \tag{5.151}$$
$$B_2 \sqcap B_3 \sqsubseteq C_3 \tag{5.152}$$
$$B_2 \sqcap B_4 \sqsubseteq C_4 \tag{5.153}$$
$$C_1 \sqcup C_2 \sqcup C_3 \sqcup C_4 \sqsubseteq C \tag{5.154}$$

Let $\mathbf{F} = \{B_1, B_2, B_3, B_4\}$ be a forgetting signature.

The query reduced ontology $\mathcal{O}^q$ of $\mathcal{O}$ with respect to $\mathbf{F}$ is:

$$: A_1 \sqcup \forall r.D_1 \sqcup \forall r.D_2 \tag{5.155}$$
$$: A_2 \sqcup \forall r.D_3 \sqcup \forall r.D_4 \tag{5.156}$$
$$: D_1 \sqcup : D_3 \sqsubseteq C_1 \tag{5.157}$$
$$: D_1 \sqcup : D_4 \sqsubseteq C_2 \tag{5.158}$$
$$: D_2 \sqcup : D_3 \sqsubseteq C_3 \tag{5.159}$$
$$: D_2 \sqcup : D_4 \sqsubseteq C_4 \tag{5.160}$$
$$C_1 \sqcup C_2 \sqcup C_3 \sqcup C_4 \sqsubseteq C \tag{5.161}$$

where D_1, D_2, D_3, and D_4 are complex de ners. We can see that the four de ners are Type 2 de ners. Observe the axiom (5.161) is in axiom form, because the normal of $\mathcal{O}^q$ does not normalize axioms that do not use forgetting symbols.

Four UB inferences can be performed. The rst and second inferences use (5.155)

as a first premise, (5.156) as a second premise. The third premise of the first inference is the set consisting of (5.157) and (5.158). The third premise of the second inference is the set consisting of (5.159) and (5.160). The conclusion of the first inference is (5.162), and the conclusion of the second inference is (5.163). The third and fourth inferences use (5.156) as a first premise, (5.155) as a second premise. The third premise of the third inference is the set consisting of (5.157) and (5.159). The third premise of the second inference is the set consisting of (5.158) and (5.160). The conclusion of the third inference is (5.164), and the conclusion of the fourth inference is (5.165).

$$: D_1 \mathrel{t} \forall\, r \;::\; A_2 \mathrel{t} C_1 \mathrel{t} C_2 \tag{5.162}$$

$$: D_2 \mathrel{t} \forall\, r \;::\; A_2 \mathrel{t} C_3 \mathrel{t} C_4 \tag{5.163}$$

$$: D_3 \mathrel{t} \forall\, r \;::\; A_1 \mathrel{t} C_1 \mathrel{t} C_3 \tag{5.164}$$

$$: D_4 \mathrel{t} \forall\, r \;::\; A_1 \mathrel{t} C_2 \mathrel{t} C_4 \tag{5.165}$$

We use the clauses from (5.167) to (5.170) in LB inferences. The following four conclusions are obtained.

$$D_1 \mathrel{t} \forall\, r \;:(:\, A_1 \mathrel{t} \forall\, r{:}(\forall r \;::\; A_2 \mathrel{t} C_3 \mathrel{t} C_4)) \tag{5.166}$$

$$D_2 \mathrel{t} \forall\, r \;:(:\, A_1 \mathrel{t} \forall\, r{:}(\forall r \;::\; A_2 \mathrel{t} C_1 \mathrel{t} C_2)) \tag{5.167}$$

$$D_3 \mathrel{t} \forall\, r \;:(:\, A_2 \mathrel{t} \forall\, r{:}(\forall r \;::\; A_1 \mathrel{t} C_2 \mathrel{t} C_4)) \tag{5.168}$$

$$D_4 \mathrel{t} \forall\, r \;:(:\, A_2 \mathrel{t} \forall\, r{:}(\forall r \;::\; A_1 \mathrel{t} C_1 \mathrel{t} C_3)) \tag{5.169}$$

We resolve the clauses (5.166), (5.162), (5.157), and (5.158) exhaustively on D_1 and remove them when resolution has been performed. The following resolvent clauses are obtained.

$$\forall r \;::\; A_2 \mathrel{t} \forall\, r \;:(:\, A_1 \mathrel{t} \forall\, r{:}(\forall r \;::\; A_2 \mathrel{t} C_3 \mathrel{t} C_4)) \mathrel{t} C_1 \mathrel{t} C_2 \tag{5.170}$$

$$: D_3 \mathrel{t} \forall\, r \;:(:\, A_1 \mathrel{t} \forall\, r{:}(\forall r \;::\; A_2 \mathrel{t} C_3 \mathrel{t} C_4)) \mathrel{t} C_1 \tag{5.171}$$

$$: D_4 \mathrel{t} \forall\, r \;:(:\, A_1 \mathrel{t} \forall\, r{:}(\forall r \;::\; A_2 \mathrel{t} C_3 \mathrel{t} C_4)) \mathrel{t} C_2 \tag{5.172}$$

In the same way as above, we resolve the clauses (5.167), (5.163), (5.159), and (5.160) exhaustively on D_2 and remove them when resolution has been performed. The following resolvents are obtained.

$$\forall r \;::\; A_2 \mathrel{t} \forall\, r \;:(:\, A_1 \mathrel{t} \forall\, r{:}(\forall r \;::\; A_2 \mathrel{t} C_1 \mathrel{t} C_2)) \mathrel{t} C_3 \mathrel{t} C_4 \tag{5.173}$$

$$: D_3 \sqcap \forall r.\neg(\neg A_1 \sqcap \forall r.(\forall r.A_2 \sqcup C_1 \sqcup C_2)) \sqcup C_3 \tag{5.174}$$

$$: D_4 \sqcap \forall r.\neg(\neg A_1 \sqcap \forall r.(\forall r.A_2 \sqcup C_1 \sqcup C_2)) \sqcup C_4 \tag{5.175}$$

Next, we resolve on D_3 the clauses (5.168), (5.164), (5.171), and (5.174), and remove them when the resolution inferences have been performed. We get the following resolvents.

$$\forall r.A_1 \sqcap \forall r.\neg(\neg A_2 \sqcap \forall r.(\forall r.A_1 \sqcup C_2 \sqcup C_4)) \sqcup C_1 \sqcup C_3 \tag{5.176}$$

$$\forall r.\neg(\neg A_2 \sqcap \forall r.(\forall r.A_1 \sqcup C_2 \sqcup C_4)) \sqcap \forall r.\neg(\neg A_1 \sqcap \forall r.(\forall r.A_2 \sqcup C_3 \sqcup C_4)) \sqcup C_1 \tag{5.177}$$

$$\forall r.\neg(\neg A_2 \sqcap \forall r.(\forall r.A_1 \sqcup C_2 \sqcup C_4)) \sqcap \forall r.\neg(\neg A_1 \sqcap \forall r.(\forall r.A_2 \sqcup C_1 \sqcup C_2)) \sqcup C_3 \tag{5.178}$$

Finally, we resolve exhaustively on D_4 the clauses (5.169), (5.165), (5.172), and (5.175), and obtain the following resolvents.

$$\forall r.\neg(\neg A_2 \sqcap \forall r.(\forall r.A_1 \sqcup C_1 \sqcup C_3)) \sqcap \forall r.A_1 \sqcup C_2 \sqcup C_4 \tag{5.179}$$

$$\forall r.\neg(\neg A_2 \sqcap \forall r.(\forall r.A_1 \sqcup C_1 \sqcup C_3)) \sqcap \forall r.\neg(\neg A_1 \sqcap \forall r.(\forall r.A_2 \sqcup C_3 \sqcup C_4)) \sqcup C_2 \tag{5.180}$$

$$\forall r.\neg(\neg A_2 \sqcap \forall r.(\forall r.A_1 \sqcup C_1 \sqcup C_3)) \sqcap \forall r.\neg(\neg A_1 \sqcap \forall r.(\forall r.A_2 \sqcup C_1 \sqcup C_2)) \sqcup C_4 \tag{5.181}$$

We remove the clauses used in resolution. The de ners D_1, D_2, D_3, and D_4 would consequently occur only positively in the two clauses (5.155) and (5.156). We purify the four de ners by substituting $\top$ for them. Purifying the de ners makes (5.155) and (5.156) tautologies, and so we remove them.

The nal query forgetting view is the following ontology.

$$\forall r.A_2 \sqcap \forall r.\neg(\neg A_1 \sqcap \forall r.(\forall r.A_2 \sqcup C_3 \sqcup C_4)) \sqcup C_1 \sqcup C_2$$

$$\forall r.A_2 \sqcap \forall r.\neg(\neg A_1 \sqcap \forall r.(\forall r.A_2 \sqcup C_1 \sqcup C_2)) \sqcup C_3 \sqcup C_4$$

$$\forall r.A_1 \sqcap \forall r.\neg(\neg A_2 \sqcap \forall r.(\forall r.A_1 \sqcup C_2 \sqcup C_4)) \sqcup C_1 \sqcup C_3$$

$$\forall r.A_1 \sqcap \forall r.\neg(\neg A_2 \sqcap \forall r.(\forall r.A_1 \sqcup C_1 \sqcup C_3)) \sqcup C_2 \sqcup C_4$$

$$\forall r.\neg(\neg A_2 \sqcap \forall r.(\forall r.A_1 \sqcup C_2 \sqcup C_4)) \sqcap \forall r.\neg(\neg A_1 \sqcap \forall r.(\forall r.A_2 \sqcup C_3 \sqcup C_4)) \sqcup C_1$$

$$\forall r.\neg(\neg A_2 \sqcap \forall r.(\forall r.A_1 \sqcup C_2 \sqcup C_4)) \sqcap \forall r.\neg(\neg A_1 \sqcap \forall r.(\forall r.A_2 \sqcup C_1 \sqcup C_2)) \sqcup C_3$$

$$\forall r.\neg(\neg A_2 \sqcap \forall r.(\forall r.A_1 \sqcup C_1 \sqcup C_3)) \sqcap \forall r.\neg(\neg A_1 \sqcap \forall r.(\forall r.A_2 \sqcup C_3 \sqcup C_4)) \sqcup C_2$$

$$\forall r.(\neg A_2 \sqcup \forall r.(\forall r.C_1 \sqcup C_3)) \sqcup \forall r.(\neg A_1 \sqcup \forall r.(\forall r.C_2 \sqcup C_1) \sqcup C_4$$
$$C_1 \sqcup C_2 \sqcup C_3 \sqcup C_4 \sqsubseteq C$$

Example 5.4.13. Consider the following ontology $\mathcal{O}$.

$$A_1 \sqsubseteq \forall s.B_1 \sqcup \forall s.B_2 \tag{5.182}$$
$$A_2 \sqsubseteq \forall r.B_3 \sqcup \forall r.B_4 \tag{5.183}$$
$$A_3 \sqsubseteq \forall t.B_5 \sqcup \forall t.B_6 \tag{5.184}$$
$$B_1 \sqcap B_3 \sqsubseteq C_1 \tag{5.185}$$
$$B_2 \sqcap B_3 \sqsubseteq C_2 \tag{5.186}$$
$$B_5 \sqcap B_4 \sqsubseteq C_3 \tag{5.187}$$
$$B_6 \sqcap B_4 \sqsubseteq C_4 \tag{5.188}$$
$$B_1 \sqsubseteq \neg G_1 \tag{5.189}$$

Let $\mathbf{F} = \{B_1, B_2, B_3, B_4, B_5, B_6\}$ be a forgetting signature. The query reduced ontology $\mathcal{O}^q$ of $\mathcal{O}$ with respect to $\mathbf{F}$ has the following clauses.

$$\neg A_1 \sqcup \forall s.D_1 \sqcup \forall s.D_2 \tag{5.190}$$
$$\neg A_2 \sqcup \forall r.D_3 \sqcup \forall r.D_4 \tag{5.191}$$
$$\neg A_3 \sqcup \forall t.D_5 \sqcup \forall t.D_6 \tag{5.192}$$
$$\neg D_1 \sqcup \neg D_3 \sqcup C_1 \tag{5.193}$$
$$\neg D_2 \sqcup \neg D_3 \sqcup C_2 \tag{5.194}$$
$$\neg D_5 \sqcup \neg D_4 \sqcup C_3 \tag{5.195}$$
$$\neg D_6 \sqcup \neg D_4 \sqcup C_4 \tag{5.196}$$
$$\neg D_1 \sqcup \neg G_1 \tag{5.197}$$

where D_1, D_2, D_3, D_4, D_5, and D_6 are definers. The six definers are complex Type 2 definers.

We perform two UB inferences. The first premise of the two inferences is (5.191). The second premise is (5.190) in the first inference, and (5.192) in the second inference. The third premise of the first inference consists of the clauses (5.193) and (5.194). The third premise of the second inference consists of the clauses (5.195) and (5.196). The clause (5.198) is the conclusion of the first inference, and (5.199) is the conclusion of

the second inference.

$$: D_3 t 8\ s\ ::\ A_1 t\ C_1 t\ C_2 \tag{5.198}$$

$$: D_4 t 8\ t\ ::\ A_3 t\ C_3 t\ C_4 \tag{5.199}$$

We perform LB inferences and obtain the following conclusions.

$$D_2 t 8\ s\ :(: A_1 t 8\ s :: G_1) \tag{5.200}$$

$$D_3 t 8\ r\ :(: A_2 t 8\ r :(8 t\ ::\ A_3 t\ C_3 t\ C_4)) \tag{5.201}$$

$$D_4 t 8\ r\ :(: A_2 t 8\ r :(8 s\ ::\ A_1 t\ C_1 t\ C_2)) \tag{5.202}$$

Resolution is not applicable on D_1 because there is not a clause that uses D_1 with positive polarity as a top-level literal.

We resolve exhaustively on D_2 the clauses (5.194), and (5.200), and obtain the following clause.

$$: D_3 t 8\ s\ :(: A_1 t 8\ s :: G_1) t\ C_2 \tag{5.203}$$

The clauses (5.194) and (5.200) are removed.

Next, we resolve the clauses (5.193), (5.194), (5.198), (5.201), and (5.203) on D_3. The following resolvents would be computed.

$$: D_1 t 8\ r\ :(: A_2 t 8\ r :(8 t\ ::\ A_3 t\ C_3 t\ C_4)) t\ C_1 \tag{5.204}$$

$$: D_2 t 8\ r\ :(: A_2 t 8\ r :(8 t\ ::\ A_3 t\ C_3 t\ C_4)) t\ C_2 \tag{5.205}$$

$$8 r\ :(: A_2 t 8\ r :(8 t\ ::\ A_3 t\ C_3 t\ C_4)) t 8\ s\ ::\ A_1 t\ C_1 t\ C_2 \tag{5.206}$$

$$8 r\ :(: A_2 t 8\ r :(8 t\ ::\ A_3 t\ C_3 t\ C_4)) t 8\ s\ :(: A_1 t 8\ s :: G_1) t\ C_2 \tag{5.207}$$

We remove the clauses (5.193), (5.194), (5.198), (5.201), and (5.203).

The de ners D_1 and D_2 become simple de ners because the clauses (5.193), (5.194) has been removed. We resolve D_1 and D_2 using the simple de ner elimination process, which substitutes the clause (5.208) below for (5.190), (5.204), and (5.205).

$$\begin{aligned} : A_1 t 8\ s :(: G_1 u\ (8 r\ :(: A_2 t 8\ r :(8 t\ ::\ A_3 t\ C_3 t\ C_4)) t\ C_1)) \\ t 8\ s :(8 r\ :(: A_2 t 8\ r :(8 t\ ::\ A_3 t\ C_3 t\ C_4)) t\ C_2) \end{aligned} \tag{5.208}$$

In the nal resolution iteration, we resolve the clauses (5.195), (5.196), (5.199),

and (5.202) on D_4. The following resolvents would be obtained.

$$: D_5 \sqcup \forall r.(\neg A_2 \sqcup \forall r.(\forall s.\neg A \sqcup \neg C_1 \sqcup \neg C_2)) \sqcup C_3 \tag{5.209}$$

$$: D_6 \sqcup \forall r.(\neg A_2 \sqcup \forall r.(\forall s.\neg A \sqcup \neg C_1 \sqcup \neg C_2)) \sqcup C_4 \tag{5.210}$$

$$\forall r.(\neg A_2 \sqcup \forall r.(\forall s.\neg A_1 \sqcup \neg C_1 \sqcup \neg C_2)) \sqcup \forall t.\neg A_3 \sqcup \neg C_3 \sqcup \neg C_4 \tag{5.211}$$

As resolution has been performed, we remove the clauses (5.195), (5.196), (5.199), and (5.202) where were used as premises to the resolution inferences.

The de ners D_3, D_4, D_5, and D_6 are now simple de ners. The two de ners D_3 and D_4 are eliminated using the de ner puri cation rule, which substitutes $\top$ for D_3 and D_4. The clause (5.191) is removed as it becomes a tautology by the puri cation. The two de ners D_5 and D_6 are eliminated using the simple de ner elimination process, which removes the clauses (5.192), (5.209), and (5.210), and introduces the following clause.

$$: A_3 \sqcup \forall t.(\forall r.(\neg(\neg A_2 \sqcup \forall r.(\forall s.\neg A \sqcup \neg C_1 \sqcup \neg C_2)) \sqcup C_3) \sqcup \forall t.(\forall r.(\neg(\neg A_2 \sqcup \forall r.(\forall s.\neg A \sqcup \neg C_1 \sqcup \neg C_2)) \sqcup C_4) \tag{5.212}$$

The query forgetting view $\mathcal{V}^q_{\mathcal{ALCI}}$ of $\mathcal{O}$ with respect to $\mathcal{F}$ is the ontology consisting of the clauses (5.206), (5.207), (5.208), (5.211), and (5.212). That is, $\mathcal{V}^q_{\mathcal{ALCI}}$ is the following ontology.

$$\forall r.(\neg A_2 \sqcup \forall r.(\forall t.\neg A_3 \sqcup \neg C_3 \sqcup \neg C_4)) \sqcup \forall s.\neg A_1 \sqcup \neg C_1 \sqcup \neg C_2$$

$$\forall r.(\neg A_2 \sqcup \forall r.(\forall t.\neg A_3 \sqcup \neg C_3 \sqcup \neg C_4)) \sqcup \forall s.(\neg A_1 \sqcup \forall s.\neg G_1) \sqcup \neg C_2$$

$$\forall r.(\neg A_2 \sqcup \forall r.(\forall s.\neg A \sqcup \neg C_1 \sqcup \neg C_3)) \sqcup \forall t.\neg A_3 \sqcup \neg C_3 \sqcup \neg C_4$$

$$: A_1 \sqcup \forall s.(\neg G_1 \sqcup \forall u.(\forall r.(\neg(\neg A_2 \sqcup \forall r.(\forall t.\neg A_3 \sqcup \neg C_3 \sqcup \neg C_4)) \sqcup \neg C_1)) \sqcup \forall s.(\forall r.(\neg(\neg A_2 \sqcup \forall r.(\forall t.\neg A_3 \sqcup \neg C_3 \sqcup \neg C_4)) \sqcup \neg C_2)$$

$$: A_3 \sqcup \forall t.(\forall r.(\neg(\neg A_2 \sqcup \forall r.(\forall s.\neg A \sqcup \neg C_1 \sqcup \neg C_2)) \sqcup C_3) \sqcup \forall t.(\forall r.(\neg(\neg A_2 \sqcup \forall r.(\forall s.\neg A \sqcup \neg C_1 \sqcup \neg C_2)) \sqcup C_4)$$

In the previous examples, UB inferences were performed at the beginning of the elimination process. One dif culty is that UB inferences may only be possible after de ners have been eliminated. The following example explains this issue.

Example 5.4.14. Consider the following ontology $\mathcal{O}$.

$$B \sqsubseteq \forall s.B_3 \tag{5.213}$$
$$A_1 \sqsubseteq \forall s.B_1 \sqcap \forall s.B_2 \sqcup B \tag{5.214}$$
$$A_2 \sqsubseteq \forall r.B_4 \sqcap \forall r.B_5 \tag{5.215}$$
$$A_3 \sqsubseteq B \sqcap \forall u.B_6 \tag{5.216}$$
$$B_1 \sqcap B_4 \sqsubseteq C_1 \tag{5.217}$$
$$B_1 \sqcap B_5 \sqsubseteq C_2 \tag{5.218}$$
$$B_2 \sqcap B_4 \sqsubseteq C_3 \tag{5.219}$$
$$B_2 \sqcap B_5 \sqsubseteq C_4 \tag{5.220}$$
$$B_3 \sqcap B_6 \sqsubseteq C_5 \tag{5.221}$$
$$B_6 \sqsubseteq \neg G \tag{5.222}$$

Let $\mathbf{F} = \{B, B_1, B_2, B_3, B_4, B_5, B_6\}$ be a forgetting signature.

The query reduced ontology $\mathcal{O}^q$ of $\mathcal{O}$ with respect to $\mathbf{F}$ consists of the following.

$$\neg A_1 \sqcup \forall s.D_1 \sqcup \forall s.D_2 \sqcup \forall s.D_3 \tag{5.223}$$
$$\neg A_2 \sqcup \forall r.D_4 \sqcup \forall r.D_5 \tag{5.224}$$
$$\neg A_3 \sqcup \forall s.D_3 \sqcup \forall r.D_6 \tag{5.225}$$
$$\neg D_1 \sqcup \neg D_4 \sqcup C_1 \tag{5.226}$$
$$\neg D_1 \sqcup \neg D_5 \sqcup C_2 \tag{5.227}$$
$$\neg D_2 \sqcup \neg D_4 \sqcup C_3 \tag{5.228}$$
$$\neg D_2 \sqcup \neg D_5 \sqcup C_4 \tag{5.229}$$
$$\neg D_3 \sqcup \neg D_6 \sqcup C_5 \tag{5.230}$$
$$\neg D_3 \sqcup \neg G \tag{5.231}$$

where D_1, D_2, D_3, D_4, D_5, and D_6 are de ners. All six de ners are Type 2 complex de ners.

Observe that no UB inferences are possible. Moreover, the conclusions of the LB inferences which have D_1, D_2, D_3, D_4, and D_5 as top-level literals are tautologies because they would use the technical axiom $\neg Dist \sqsubseteq \top$.

We perform one LB inference. The first premise of the inference is the clause (5.225), the second premise is the set consisting of (5.231). The following clause is

obtained from the inference.

$$D_6(t) \sqcup (\neg A_3(t) \sqcup \neg G) \tag{5.232}$$

We resolve the clauses (5.230) and (5.232), and obtain the following resolvent.

$$\neg D_3(t) \sqcup (\neg A_3(t) \sqcup \neg G) \sqcup C_5 \tag{5.233}$$

Next, we remove the clauses (5.230) and (5.232).

As the clauses are removed, the de ners D_3 and D_6 become simple de ners. The de ner D_6 is eliminated by de ner puri cation because it occurs only positively in (5.225). As D_6 is puri ed away, the clause (5.225) becomes a tautology and is removed. The de ner D_3 is eliminated by the simple de ner elimination process, which substitutes the clause (5.234) for (5.223), (5.231), and (5.233).

$$\neg A_1(t) \sqcup \neg D_1(t) \sqcup \neg D_2(t) \sqcup \neg(\neg G \sqcup (\forall u.(\neg A_3(t) \sqcup \neg G) \sqcup C_5)) \tag{5.234}$$

The ontology $\mathcal{V}^q_{\mathcal{ALCI}}$ now consists of the following clauses.

$$\neg A_1(t) \sqcup \neg(\neg G \sqcup (\forall u.(\neg A_3(t) \sqcup \neg G) \sqcup C_5))(t) \sqcup \neg D_1(t) \sqcup \neg D_2$$
$$\neg A_2(t) \sqcup \neg D_4(t) \sqcup \neg D_5$$
$$\neg D_1(t) \sqcup \neg D_4(t) \sqcup C_1$$
$$\neg D_1(t) \sqcup \neg D_5(t) \sqcup C_2$$
$$\neg D_2(t) \sqcup \neg D_4(t) \sqcup C_3$$
$$\neg D_2(t) \sqcup \neg D_5(t) \sqcup C_4$$

In this form, UB and LB inferences can now be performed on $\mathcal{V}^q_{\mathcal{ALCI}}$ to eliminate the complex Type 2 de ners D_1, D_2, D_4, D_5 as done in Example 5.4.12.

In the nal example, we show the use of the Relaxing rule to trigger UB inferences.

Example 5.4.15. Consider the following ontology $\mathcal{O}$.

$$B \sqsubseteq \forall s.B_3 \tag{5.235}$$
$$A_1 \sqsubseteq \forall s.B_1(t) \sqcup \forall s.B_2(t) \sqcup B \tag{5.236}$$
$$A_2 \sqsubseteq \forall r.B_4(t) \sqcup \forall r.B_5 \tag{5.237}$$
$$A_3 \sqsubseteq \forall t.B_6(t) \sqcup \forall t.B_7(t) \sqcup B \tag{5.238}$$

$$B_1 \sqcap B_4 \sqsubseteq C_1 \quad (5.239)$$
$$B_1 \sqcap B_5 \sqsubseteq C_2 \quad (5.240)$$
$$B_2 \sqcap B_4 \sqsubseteq C_3 \quad (5.241)$$
$$B_2 \sqcap B_5 \sqsubseteq C_4 \quad (5.242)$$
$$B_6 \sqcup B_7 \sqsubseteq C_5 \quad (5.243)$$
$$B_7 \sqcup B_3 \sqsubseteq C_6 \quad (5.244)$$
$$B_3 \sqsubseteq \neg G \quad (5.245)$$

Let $\mathbf{F} = \{B, B_1, B_2, B_3, B_4, B_5, B_6, B_7\}$ be a forgetting signature. The query reduced ontology $\mathcal{O}^q$ of $\mathcal{O}$ with respect to $\mathbf{F}$ is the following ontology

$$\neg A_1 \sqcup \forall s.D_1 \sqcup \forall s.D_2 \sqcup \forall s.D_3 \quad (5.246)$$
$$\neg A_2 \sqcup \forall r.D_4 \sqcup \forall r.D_5 \quad (5.247)$$
$$\neg A_3 \sqcup \forall t.D_6 \sqcup \forall t.D_7 \sqcup \forall s.D_3 \quad (5.248)$$
$$\neg D_1 \sqcup \neg D_4 \sqsubseteq C_1 \quad (5.249)$$
$$\neg D_1 \sqcup \neg D_5 \sqsubseteq C_2 \quad (5.250)$$
$$\neg D_2 \sqcup \neg D_4 \sqsubseteq C_3 \quad (5.251)$$
$$\neg D_2 \sqcup \neg D_5 \sqsubseteq C_4 \quad (5.252)$$
$$\neg D_6 \sqcup \neg D_7 \sqsubseteq C_5 \quad (5.253)$$
$$\neg D_7 \sqcup \neg D_3 \sqsubseteq C_7 \quad (5.254)$$
$$\neg D_3 \sqcup \neg G \quad (5.255)$$

where $D_1, D_2, D_3, D_4, D_5, D_6$, and D_7 are de ners. All de ners are Type 2 complex de ners.

Observe that UB inferences cannot be performed. Moreover, the conclusions of all possible LB inferences would be tautologies. We use the Relaxing rule to substitute the following clause for (5.246) and (5.255).

$$\neg A_1 \sqcup \forall s.D_1 \sqcup \forall s.D_2 \sqcup \forall s.\neg G \quad (5.256)$$

In this form, we can perform four UB inferences on the clauses (5.256), (5.247), (5.249), (5.250), (5.251), and (5.252) in the same way done in Example 5.4.12.

Lemma 5.4.16. The conclusion of the Lower Bound Extraction rule is entailed by the

premises.

Proof. We prove the lemma by contradiction. We show that a contradiction can be obtained from the premises and the negation of the conclusion. In rst-order logic notations, the rst premise of the rule is translated to the following formula.

$$8x: (C(x) \_ \bigvee_{i=1}^{n} 8y_i: (: r_i(x; y_i) \_ D_i(y_i))) \tag{5.257}$$

The second premise is translated to the following rst-order formula.

$$8x \bigwedge_{i=2}^{n} (: D_i(x) \_ C_i(x)) \tag{5.258}$$

The negated conclusion translated to the conjunction of the following formulas:

$$: D_1(a) \tag{5.259}$$

$$r_1(f(a); a) \tag{5.260}$$

$$: C_1(f(a)) \tag{5.261}$$

$$\bigwedge_{i=2}^{n} (r_i(f(a); g_i(f(a)))) \tag{5.262}$$

$$\bigwedge_{i=2}^{n} : C_i(g_i(f(a))) \tag{5.263}$$

where a is a witness instance, and f and g_i are Skolem functions.

We now have:

$$\bigwedge_{i=2}^{n} : D_i(g_i(f(a))) \quad \text{Res}(5.258; 5.263) \tag{5.264}$$

$$\_\_{i=1}^{n} (: r_i(f(a); y_i) \_ D_i(y_i)) \quad \text{Res}(5.257; 5.261) \tag{5.265}$$

$$D_1(a) \_\_ \_\_{i=2}^{n} (: r_i(f(a); y_i) \_ D_i(y_i)) \quad \text{Res}(5.260; 5.265) \tag{5.266}$$

$$\_\_{i=2}^{n} (: r_i(f(a); y_i) \_ D_i(y_i)) \quad \text{Res}(5.258; 5.266) \tag{5.267}$$

$$\_\_{i=2}^{n} D_i(g_i(f(a))) \quad \text{Res}(5.262; 5.267) \tag{5.268}$$

$$? \quad \text{Res}(5.264; 5.268) \tag{5.269}$$

where $\text{Res}(x; y)$ means that the formula on the left is obtained by resolving the formulas x and y. □

We can prove a stronger result than that of Lemma 5.4.16 when the rst premise

has only one de ner below universal role restriction.

Lemma 5.4.17. Let C_1 be the clause $C \sqcup \forall r.D$ be a clause, where C is a concept, r a role name, and D a de ner. Let C_2 be the conclusion $\forall r. \neg C \sqcup D$ obtained from a LB inference, where the rst premise of the inference is C_1 and the second premise is $\mathcal{A}$. Then the two ontologies $\{C_1\}$ and $\{C_2\}$ are logically equivalent.

Proof. Suppose we have the following.

$$\mathcal{O}_1 = \{C \sqcup \forall r.D\} \tag{5.270}$$

$$\mathcal{O}_2 = \{D \sqcup \forall r.\neg C\} \tag{5.271}$$

To prove logical equivalence we need to show the following.

$$\mathcal{O}_1 \models \mathcal{O}_2 \tag{5.272}$$

$$\mathcal{O}_2 \models \mathcal{O}_1 \tag{5.273}$$

From Lemma 5.4.16 we know that (5.272) is true.

We show that (5.273) by showing that $\mathcal{O}_2, \neg\mathcal{O}_1 \models \bot$. Let a be an individual, the ontology $\mathcal{O}_2, \neg\mathcal{O}_1$ is equivalent to:

$$D \sqcup \forall r.\neg C \tag{5.274}$$

$$\neg C(a) \tag{5.275}$$

$$(\exists r.\neg D)(a) \tag{5.276}$$

From (5.276) there must be an instance b satisfying the following.

$$r(a,b) \tag{5.277}$$

$$\neg D(b) \tag{5.278}$$

From (5.274) and (5.278), we get the following.

$$(\forall r.\neg C)(b) \tag{5.279}$$

From (5.277) and (5.279) we get:

$$C(a) \tag{5.280}$$

A contradiction is now obtained from (5.280) and (5.275). □

Lemma 5.4.17 can be used to prove that the elimination process of Type 1 de ners, described in Algorithm 7, preserves model equivalence with respect to the non-forgotten signature $sig(O) \cap F$.

Lemma 5.4.18. Let O be an ontology, and N a set of Type 1 de ners occurring in O. Let V be the ontology obtained by eliminating the Type 1 de ners from O using the process explained in Algorithm 7. The two ontologies O and V are model inseparable with respect to the signature $sig(O) \cap N$.

Proof. We can describe Algorithm 7 as follows. Algorithm 7 iterates over the Type 1 de ners. Every iteration eliminates one Type 1 de ner. An iteration eliminating a de ner D replaces every clause of the following form with the conclusion of the LB inference performed on the clause.

$$C \vee t \quad 8\, r.D \tag{5.281}$$

When all clauses of the form (5.281) are replaced, Algorithm 1 is called to eliminate D. When all Type 1 de ners have been eliminated, simple de ners occurring in O are eliminated using the process explained in Algorithm 6.

Lemma 4.3.3 shows that the call to Algorithm 1 preserves model inseparability with respect to $sig(O) \cap N$. Lemma 5.4.5 shows that the call to Algorithm 6 preserves model inseparability with respect to $sig(O) \cap N$. We show that the substitution of the conclusions of the LB inferences for the premises of the inferences preserves logical equivalence.

Consider rst the substitution of the conclusion of a LB inference for a clause of the form (5.281). Let C_1 be the clause of the form (5.281) being substituted for, and C_2 the conclusion of the LB inference. Let O_1 be a snapshot of the ontology before the substitution, and O_2 be the snapshot after the substitution, and assume that the two ontologies are consistent. We show that O_1 and O_2 are logically equivalent by showing that every model of O_1 is a model of O_2, and every model of O_2 is a model of O_1. Let O_0 be the subset of O_1 which does not contain C_1. The following is true about O_1 and O_2.

$$O_1 = O_0 \cup \{C_1\} \tag{5.282}$$

$$O_2 = O_0 \cup \{C_2\} \tag{5.283}$$

Let $\mathcal{I}$ be a model of $\mathcal{O}_1$, then the following is true about $\mathcal{I}$.

$$\mathcal{I} \models \mathcal{O}_0 \tag{5.284}$$

$$\mathcal{I} \models \{C_1 \sqsubseteq C_2\} \tag{5.285}$$

From (5.285) and Lemma 5.4.17, we get that $\mathcal{I}$ is a model of $\{C_2 \sqsubseteq C_2\}$. Because $\mathcal{I}$ is a model of $\mathcal{O}_0$ as well, we get that $\mathcal{I}$ is a model of $\mathcal{O}_2$. The same argument proves that a model $\mathcal{I}$ of $\mathcal{O}_2$ is a model of $\mathcal{O}_1$. □

Theorem 5.4.19. Let $\mathcal{O}$ be an ontology, and $\mathbf{F}$ a forgetting signature. Suppose $\mathcal{O}^q$ is the reduced ontology obtained from $\mathcal{O}$ with respect to $\mathbf{F}$, and $\mathcal{V}^q_{\mathcal{ALCI}}$ is the ontology obtained by eliminating simple and complex de ners from $\mathcal{O}^q$. We have:

1. $\mathcal{O} \equiv^{Q}_{sig(\mathcal{O})\cap\mathbf{F}} \mathcal{V}^q_{\mathcal{ALCI}}$, and
2. If $sig(\mathcal{V}^q_{\mathcal{ALCI}}) \setminus \mathbf{N}_d = \emptyset$, then $\mathcal{V}^q_{\mathcal{ALCI}}$ is a query forgetting view of $\mathcal{O}$ with respect to $\mathbf{F}$.

The proof of the theorem is delivered through the discussions and lemmas in the remainder of this section. Note that 2 in Theorem 5.4.19 directly follows from 1 and De nition 3.5.15. So we only need to prove 1.

We know from Theorem 5.2.4 that $\mathcal{O}$ and $\mathcal{O}^q$ are query inseparable with respect to $\mathbf{F}$. The ontology $\mathcal{V}^q_{\mathcal{ALC}}$ obtained from $\mathcal{O}^q$ by eliminating simple de ners is also query inseparable from $\mathcal{O}$ with respect to $\mathbf{F}$. A proof for this was given in Lemma 5.4.5. Furthermore, the elimination of Type 1 de ners preserves model inseparability with respect to the non-forgotten signature $sig(\mathcal{O}) \cap \mathbf{F}$. From Lemma 3.5.10 we can see that the elimination of Type 1 de ners also preserves query inseparability with respect to $sig(\mathcal{O}) \cap \mathbf{F}$. It remains to show that the complex de ner elimination process preserves query inseparability. It remains to consider the elimination of Type 2 de ners.

We analyse each component in elimination process of Type 2 de ners separately. There are ve components to consider.

1. The LB and the UB inferences: This component preserves logical equivalence, because it only makes some implicit entailments of $\mathcal{V}^q_{\mathcal{ALCI}}$ explicit. From Lemmas 3.5.8 and 3.5.10, we see that logical equivalence implies query inseparability. So this component preserves query inseparability with respect to $sig(\mathcal{O}) \cap \mathbf{F}$.

2. Simple de ner elimination. This component relies only on syntactic transformation as well as puri cation rules. Lemma 5.4.5 showed that elimination of simple de ners preserves query inseparability with respect to $sig(\mathcal{O}) \cap \mathbf{F}$.

3. Resolution: Resolution makes implicit information explicit, so it preserves logical equivalence.

4. Elimination of resolution premises: Theorem 4.3.3 showed that removing the premises of resolution inferences preserves model inseparability. However, the result of the theorem cannot be admitted in our case because it was obtained assuming a specific normal form of the input $\mathcal{O}$ $^{\text{clausal}}$. In particular, the eliminated symbols are assumed to occur only as top level, possibly negated, disjuncts, whereas Type 2 definers can additionally be used below role restrictions. Therefore, query inseparability with respect to $\text{sig}(\mathcal{O}) \cap \mathbf{F}$ would need to be verified when the premises of resolution inferences are removed.

5. Elimination of $\mathcal{D}^u$ clauses: Query inseparability with respect to $\text{sig}(\mathcal{O}) \cap \mathbf{F}$ is needed to be verified for this component.

Based on the above analysis we focus on proving query inseparability with respect to $\text{sig}(\mathcal{O}) \cap \mathbf{F}$ when clauses are removed in the components 4 and 5.

We consider 4 first. Resolution is performed iteratively. Each iteration resolves the clauses that use a certain definer, and eliminates the premises of the resolution inferences when resolution has completed. We consider a single iteration where the clauses that use a definer D_1 are resolved together on D_1, then eliminated. Let $\mathcal{V}^{\text{pre}}$ be the snapshot of $\mathcal{V}^{\text{q}}_{\text{ALCI}}$ before removing the clauses. Suppose $\mathcal{V}^{\text{pre}}$ has a set of clauses $C_i \sqcup D_1$ with $1 \leq i \leq n$ and $n \geq 1$. We replace these clauses with their combined form $C \sqcup D_1$ where $C = \bigsqcup_{i=1}^{n} C_i$. In the same way, if there are m clauses of the form $E_j \sqcup \neg D_1$ with $1 \leq j \leq m$ in $\mathcal{V}^{\text{pre}}$, we replace them with the combined form $E \sqcup \neg D_1$ where $E = \bigsqcup_{j=1}^{m} E_j$. Combining clauses is a technique that we have used in numerous places, and was shown to preserve logical equivalence, see for instance the proof of Lemma 4.3.3. Note that the clauses $C_i \sqcup D_1$ are the conclusions of LB inferences. The concept $C_i = \forall r_1.(G \sqcap \bigsqcap_{j=2}^{n} \forall r_j.F_j)$ is a conjunct in C if and only if a LB inference was performed with $G \sqcup \forall r_1.D_1 \sqcup \ldots \sqcup \forall r_n.D_n$ as a first premise, and a set of $n-1$ clauses $\neg D_j \sqcup F_j$ as a second premise, where $2 \leq j \leq n$. We can view $\mathcal{V}^{\text{pre}}$ as follows.

$$\mathcal{V}^{\text{pre}} = \mathcal{V}_0 \cup \mathcal{V}_8 \cup \{C \sqcup D_1, E \sqcup \neg D_1, C \sqcup E\} \tag{5.286}$$

where $\mathcal{V}_0$ is a set of clauses that does not use D_1, and $\mathcal{V}_8$ is a set of clauses of the form $G \sqcup \forall r_1.D_1 \sqcup \ldots \sqcup \forall r_n.D_n$. The clauses $C \sqcup D_1$ and $E \sqcup \neg D_1$ are as above, and $C \sqcup E$ is their resolvent.

Let $\mathcal{V}^{post}$ be the snapshot of $\mathcal{V}^{q}_{ALCI}$ after removing the clauses used in resolution. We have the following.

$$\mathcal{V}^{post} = \mathcal{V}_0 \cup \mathcal{V}_8 \cup \{Ct \; Eg\} \tag{5.287}$$

Let $\mathcal{A}$ be an arbitrary ABox, $q(\vec{x})$ a conjunctive query with $\vec{x} = x_1, x_2, \ldots, x_k$, and $\vec{a} = a_1, a_2, \ldots, a_k$ an answer to $q(\vec{x})$ where $k \geq 0$. Our aim is to show that $\mathcal{V}^{pre} \equiv_{sig(\mathcal{O}) \cap F} \mathcal{V}^{post}$, which requires proving the following.

$$(\mathcal{V}^{pre}, \mathcal{A}) \models q(\vec{a}) \quad \text{implies} \quad (\mathcal{V}^{post}, \mathcal{A}) \models q(\vec{a}) \tag{5.288}$$

$$(\mathcal{V}^{post}, \mathcal{A}) \models q(\vec{a}) \quad \text{implies} \quad (\mathcal{V}^{pre}, \mathcal{A}) \models q(\vec{a}) \tag{5.289}$$

The following lemma shows that (5.289) is always true.

Lemma 5.4.20. Let $\mathcal{V}^{pre}$, $\mathcal{V}^{post}$, $\mathcal{A}$, and $q(\vec{x})$ as described above. Then, if $(\mathcal{V}^{post}, \mathcal{A}) \models q(\vec{a})$ implies $(\mathcal{V}^{pre}, \mathcal{A}) \models q(\vec{a})$, where $\vec{a}$ is an answer to $q(\vec{x})$.

Proof. Suppose $(\mathcal{V}^{post}, \mathcal{A}) \models q(\vec{a})$. Also suppose $\mathcal{I}$ is a model of $(\mathcal{V}^{pre}, \mathcal{A})$ such that $q(\vec{a})$ is false in $\mathcal{I}$. Since $\mathcal{I} \models (\mathcal{V}^{pre}, \mathcal{A})$ we have that $\mathcal{I}$ is a model of $\mathcal{V}^{pre}$, and a model of $\mathcal{A}$. From the construction of $\mathcal{V}^{pre}$ and $\mathcal{V}^{post}$, we can see the following.

$$\mathcal{V}^{post} \subseteq \mathcal{V}^{pre} \tag{5.290}$$

The subsumption (5.290) implies that $\mathcal{I}$ is also a model of $\mathcal{V}^{post}$, because removing clauses does not invalidate the models. We now have $\mathcal{I} \models \mathcal{V}^{post}$ and $\mathcal{I} \models \mathcal{A}$. So, $\mathcal{I}$ is a model of $(\mathcal{V}^{post}, \mathcal{A})$. The two following contradicting remarks are true about $\mathcal{I}$.

1. $\mathcal{I} \not\models q(\vec{a})$ by assumption.
2. $\mathcal{I} \models q(\vec{a})$ because $\mathcal{I} \models (\mathcal{V}^{post}, \mathcal{A})$ and $(\mathcal{V}^{post}, \mathcal{A}) \models q(\vec{a})$.

From the contradiction we get that if $(\mathcal{V}^{post}, \mathcal{A}) \models q(\vec{a})$ and $\mathcal{I} \models (\mathcal{V}^{pre}, \mathcal{A})$ then $\mathcal{I} \models q(\vec{a})$. □

Next, we show that (5.288) is true. Suppose $(\mathcal{V}^{pre}, \mathcal{A}) \models q(\vec{a})$, and let $\mathcal{I}$ be a model of $(\mathcal{V}^{post}, \mathcal{A})$ where $q(\vec{a})$ is false. First, we highlight cases where $\mathcal{I}$ can be transformed to a model $\mathcal{J}$, such that $\mathcal{J} \models (\mathcal{V}^{pre}, \mathcal{A})$ and $\mathcal{I}$ and $\mathcal{J}$ are bisimilar with respect to $sig(\mathcal{O}) \cap F$ and $ind(\mathcal{A})$. In these cases the following contradicting observations would occur.

1. $\mathcal{J} \models q(\vec{a})$ because $\mathcal{J} \models (\mathcal{V}^{pre}, \mathcal{A})$ and $(\mathcal{V}^{pre}, \mathcal{A}) \models q(\vec{a})$.

2. $J \not\models q(a)$, because I and J are bisimilar with respect to $sig(O) \cap F$ and $ind(A)$, and $I \not\models q(a)$.

For the other cases where I cannot be transformed to a model of (V^{pre}, A), we transform I to J such that I and J are bisimilar with respect to $sig(O) \cap F$ and $ind(A)$, and we show that there is a map p with $p(x_i) = a_i$ for $1 \leq i \leq k$, and for every anonymous variable y in q, p maps y to an element in D^J such that the following is true.

1. If $A(u)$ is an atom in q, then $p(u) \in A^J$.
2. If $r(u,v)$ is an atom in q, then $(p(u), p(v)) \in r^J$.

A contradiction would be obtained as the following will occur.

1. $J \not\models q(a)$ because I and J are bisimilar with respect to $sig(O) \cap F$ and $ind(A)$, and $I \not\models q(a)$.
2. $J \models q(a)$ by construction of the map p described above.

The two contradictions imply that if $(V^{pre}, A) \models q(a)$ and I is a model of (V^{post}, A), then $I \models q(a)$, which proves (5.288).

The following definitions support the discussion below.

Definition 5.4.21. Let K be a knowledge base, I a model of K, and $b \in D^I$ a domain element.

1. By G_D we denote the set of clauses $\forall x.G \rightarrow \forall r_1.D_1 \vee \forall r_2.D_2 \vee \cdots \vee \forall r_n.D_n$ of K.
2. By b_D we denote the set of concepts G occurring in the clauses G_D.
3. By $g_{b,D}$ we denote the set of domain elements $a \in \Delta^I$ where $(a,b) \in r_1^I$ and $a \notin G^I$ for any concept G from b_D.

Let I be as before as a model of (V^{post}, A). We transform I to another model I_0 of (V^{post}, A) where for every $b \in D^{I_0}$ we have $b \in D_1^{I_0}$ if and only if $b \in D_1^I$ and $g_{b,D_1} \neq \emptyset$. Intuitively, the transformation removes b from the interpretation of D_1 if g_{b,D_1} is empty.

Lemma 5.4.22. Let I and I_0 be as above. Then $I_0 \models (V^{post}, A)$ and $I_0 \not\models q(a)$.

Proof. From (5.287), $I_0 \models V^{post}$ if and only if $I_0 \models V_0$, $I_0 \models V_8$, and $I_0 \models C \sqsubseteq E$ where V_0 is the subset of V^{pre} that does not contain D_1 and V_8 is the subset of V^{pre} whose clauses has the form $C \sqsubseteq \forall r_1.D_1 \vee \cdots \vee \forall r_n.D_n$.

$I_0 \models V_0$ because I_0 was obtained from I by changing the interpretation of D_1, and D_1 does not occur in V_0.

$I_0 \models V_8$ if and only if for every domain element $b \in D^{I_0}$ such that $b \in D_1^I$, $b \notin D_1^{I_0}$, we have $a \in C^{I_0}$ for every $a \in g_{b;D_1}$ and every $C \in G_{D_1}$. However, from the construction of I_0 we have $b \in D_1^I$ and $b \notin D_1^{I_0}$ implies $g_{b;D_1} = \emptyset$. So, we have $I_0 \models V_8$.

To show that $I_0 \models Ct\ E$, it suffices to show that for every element $b \in D_1^I$ if $b \notin D_1^{I_0}$ then $b \in (Ct\ E)_0^I$. Recall from the construction of I_0 that b is removed from $D_1^{I_0}$ only if $g_{b;D_1}$ is empty. Therefore, $b \in (\forall r_1 : G)^{I_0}$ for every $G \in b_{D_1}$. This observation, and the discussion above regarding the form of the concept C imply that $b \in C^{I_0}$. So, $b \in (Ct\ E)^{I_0}$.

From the above we see that $I_0 \models V^{post}$. Since I_0 is constructed by removing elements from the interpretation of D_1, we have $I_0 \models A$ because $D_1 \notin sig(A)$. Therefore $I_0 \models (V^{post}; A)$.

The two models I and I_0 are bisimilar with respect to $sig(V^{post}) \cap \{D_1\}$ and $ind(A)$, because I_0 differs from I only on the interpretation of D_1. Since $I \not\models q(a)$, and $sig(q) \subseteq sig(V^{post}) \cap \{D_1\}$, we have from Lemma 5.3.13 that $I_0 \not\models q(a)$. □

Next, we discuss cases where I_0 can be transformed to the model J of $(V^{pre}; A)$ described before. The construction of V^{post} and V^{pre} implies that J is a model of $(V^{pre}; A)$ if the following three conditions are satisfied: $J \models (V^{post}; A)$, $J \models Ct\ D_1$, and $J \models Et : D_1$. Let $b \in D^{I_0}$ be a domain element. Since $Ct\ E$ is a clause in V^{post}, and $I_0 \models V^{post}$, one of the following must be true.

$$b \in E^I \quad \text{and} \quad b \notin C^I \tag{5.291}$$

$$b \notin E^I \quad \text{and} \quad b \in C^I \tag{5.292}$$

$$b \in E^I \quad \text{and} \quad b \in C^I \tag{5.293}$$

We have the following cases.

Case 1: We perform the following transformation if (5.291) is true, or (5.293) is true. Let J be the model that extends I_0 by adding b to the interpretation of D_1. We have:

1. J is a model of $(V^{post}; A)$, because adding elements to the interpretation of D_1 only impacts clauses where D_1 occurs with negative polarity. Since D_1 does not occur with negative polarity in V^{post}, we have that J is a model of V^{post}. Moreover, $J \models A$, because $D_1 \notin sig(A)$. So J is a model of the knowledge base $(V^{post}; A)$.

2. I_0 and J are bisimilar with respect to $sig(V^{post}) \cap \{D_1\}$ and $ind(A)$, because only the interpretation of D_1 is modified by the transformation.

Case 2: We assume (5.292) is true. Assume that for every clause $C = G \sqcup \forall r_1.D_1 \sqcup \ldots \sqcup \forall r_n.D_n$ in G_{D_1}, and every $a \in g_{b,D_1}$ such that $a \notin G^{I_0}$, the following is true.

$$a \in (\forall r_j.D_j)^{I_0} \text{ for some } j \text{ with } 2 \leq j \leq n \tag{5.294}$$

We transform I to J by removing b from the interpretation of D_1.

First, we show that J is a model of (V^{post}, A). Consider all clauses of V^{post} where D_1 occurs positively. Since D_1 occurs positively only in the clauses as below role restrictions, we need to verify that for every $a \in g_{b,D_1}$ we have $a \in C^J$. But this is always true because of the condition in (5.294).

Second, we can see that I_0 and J are bisimilar with respect to $sig(O) \cap F$ and $ind(A)$, because only the interpretation of D_1 is modified and $D_1 \notin sig(O) \cap F$.

Case 3: We assume (5.292) is true, and suppose there is a clause $C \in G_{D_1}$ where $C = G \sqcup \forall r_1.D_1 \sqcup \ldots \sqcup \forall r_n.D_n$, and a domain element $a \in g_{b,D_1}$ such that (5.295), (5.296), and (5.297) below are true.

$$a \notin G^{I_0} \tag{5.295}$$

$$a \in (\forall r_1.D_1)^{I_0} \tag{5.296}$$

$$a \notin (\forall r_j.D_j)^{I_0} \text{ for every } j \text{ with } 2 \leq j \leq n. \tag{5.297}$$

then removing b from the interpretation of D_1 makes C not valid in J. Suppose (5.295), (5.296), and (5.297) are true. We observe the following.

1. From (5.297), there exist domain elements b_j, such that $(a, b_j) \in r_j^{I_0}$, and $b_j \notin D_j^{I_0}$ for every j where $2 \leq j \leq n$.

2. By assumption we have $a \in C^{I_0}$, where C is the conjunction of concepts of the form $\forall r_1.(H \sqcup \bigsqcup_{i=2}^{n} \forall r_i.F_i)$, $H \in b_{D_1}$, and $\neg D_i \sqcup F_i$ are clauses in V^{post} with $2 \leq i \leq n$.

3. One of the conjuncts of C is the concept $\forall r_1.(G \sqcup \bigsqcup_{i=2}^{n} \forall r_i.F_i)$ obtained from a LB inference whose first premise is G.

4. From the previous point, we have $a \in \forall r_1.(G \sqcup \bigsqcup_{i=2}^{n} \forall r_i.F_i)^{I_0}$, and $a \in (G \sqcup \bigsqcup_{i=2}^{n} \forall r_i.F_i)^{I_0}$ because $(a, b) \in r_1^{I_0}$.

5. From (5.295) and the previous point, we have $a \in (\bigsqcap_{i=2}^{n} \forall r_i.F_i)^{\mathcal{I}_0}$.

6. From the previous point and Item 1, there exists a subset of the concepts $D_2, \ldots, D_n$ where for each concept D_j ($2 \leq j \leq n$) there is a domain element $b_j \in D_j$, such that $(a, b_j) \in r_j$, and $b_j \in F_j$. For the other concepts D_j, $b_j \notin F_j$ and $b_j \notin D_j$.

Suppose s has only one concept D_k with $2 \leq k \leq n$ where D_k occurs in the disjunct $\forall r_k.D_k$ in C. Then, there are no r_k-successors of a in the interpretation of $\forall r_k.(G \sqcup \bigsqcap_{l=1, l \neq k}^{n} \forall r_l.F_l)$, because $a \in G$, and from Item 6 above, every successor is not in F_l. We have the following

1. If D_k has been eliminated in an earlier resolution iteration, then $\mathcal{V}^{post}$ must have a clause $C_k \sqcup E_k$ where $\forall r_k.(G \sqcup \bigsqcap_{l=1, l \neq k}^{n} \forall r_l.F_l)$ is a conjunct in C_k. A r_k-successor b_k of a is not in the interpretation of C_k because it is not in the interpretation of the just mentioned conjunct. Furthermore, $b_k \in E_k^{\mathcal{I}_0}$ because $b_k \in (C_k \sqcup E_k)^{\mathcal{I}_0}$ and $b_k \notin C_k^{\mathcal{I}_0}$. We transform $\mathcal{I}_0$ to $\mathcal{J}$ such that $a \in \forall r_k.D_k$, and remove b from the interpretation of D_1.

2. If D_k has not been eliminated in an earlier resolution iteration, then $\mathcal{V}^{post}$ must have a clause $D_k \sqcup C_k$ by an LB inference. Since every r_k-successor of a is not in the interpretation of C_k, we have $a \in \forall r_k.D_k$. This case cannot happen because it contradicts the assumption (5.297).

Case 4: Assume the conditions in Case 3 are true, except which we assume has two or more concepts. This case indicates the existence of a set of models of $(\mathcal{V}^{pre}, A)$, each uses a concept D from s such that the domain elements in the interpretation of $\forall r.D$.

Definition 5.4.23. Let $\mathcal{I}_0$ be a model of $(\mathcal{V}^{post}, A)$ as before. Let $\mathcal{I}$ be a model of $(\mathcal{V}^{pre}, A)$. We say $\mathcal{I}$ is extendable from $\mathcal{I}_0$ if $\Delta^{\mathcal{I}_0} \subseteq \Delta^{\mathcal{I}}$, and for every concept name C and role r, we have:

1. $C^{\mathcal{I}_0} \subseteq C^{\mathcal{I}}$, and

2. $r^{\mathcal{I}_0} \subseteq r^{\mathcal{I}}$.

The set $\mathbb{M}$ denote the set of models of $(\mathcal{V}^{pre}, A)$ that are extendable from $\mathcal{I}_0$.

Let D be as in the previous paragraph, and denote the set of r-successors of the domain element a that are not in the interpretation of D. Suppose one of the following is true.

1. V^{post} does not have a clause of the form $D(t) \leftarrow D'(t) \wedge C$, or
2. $d \notin D'^{I_0}$ implies $d \in C^{I_0}$ for every $d \in \Delta$.

where D' is a definer.

We extend I_0 to $J \in M$ by adding all elements occurring in the set to the interpretation of D, and remove every $y \in D^J$ satisfying (5.292) from the interpretation of D_1.

First, We show that J is a model of (V^{post}, A).

1. Since the transformation adds successors of the domain element to the interpretation of D, we need only to consider clauses where D occurs negatively. Let $\neg D(t) \vee C$ be a clause in V^{post}, and d a domain element added to the interpretation of D by the transformation. Since $D \in s$, we have $d \in C^{I_0}$. So, $d \in (\neg D(t) \vee C)$. Suppose $D(t) \leftarrow D'(t) \wedge C$ is a clause in V^{post}, the conditions above require that $d \notin D'^{I_0}$ or $d \in C^{I_0}$. So, $d \in (\neg D(t) \vee D'(t) \wedge C)^J$.

2. Since the transformation removes b from the interpretation of D_1, we need to show that $a \in (G(t) \wedge \forall r:D(t) \wedge \forall r_1:D_1(t, \ldots) \wedge \ldots \wedge \forall r_n:D_n(t, \ldots))^J$. Observe that adding domain elements to the interpretation of D makes $a \in (\forall r:D)^J$. So, $a \in (G(t) \wedge \forall r:D(t, \ldots) \wedge \forall r_1:D_1(t, \ldots) \wedge \ldots \wedge \forall r_n:D_n(t, \ldots))^J$ even though b is removed from the interpretation of D_1.

Second, the two models I_0 and J are bisimilar with respect to $sig(O) \cap F$ and $ind(A)$ because we only change the interpretations of definers.

Case 5: Assume the conditions in Case 3 are true, except which we assume has two or more definers. Further, assume no definer from s satisfy the two conditions in Case 4.

Let D_p and D_q be any two definers from s, b_p be a successor of a via r_p, and b_q a successor of a via r_q. Assume $b_p \in D_{p;1}^{I_0}$ where $D_{p;1}$ is a definer, $\neg D_{p;1}(t) \vee D_p(t) \wedge C_p$ is a clause in V^{post}, and $b_p \notin C_p^{I_0}$. In parallel, assume $b_q \in D_{q;1}^{I_0}$ where $D_{q;1}$ is a definer, $\neg D_{q;1}(t) \vee D_q(t) \wedge C_q$ is a clause in V^{post}, and $b_q \notin C_q^{I_0}$.

Since V^{post} only has Type 2 definers, $D_{p;1}$ must be a Type 2 definer. Furthermore, $D_{p;1}$ must be a universal definer because existential definers cannot occur in V^{post} with negative polarity in clauses where two negative definers occur. V^{post} must have a clause $G_p(t) \wedge \forall s_1:D_{p;1}(t, \ldots) \wedge \ldots \wedge \forall s_u:D_{p;u}(t, \ldots)$ in V^{post}, and a domain element a_p such that $(a_p, b_p) \in s_1^{I_0}$, $a_p \notin G_p^{I_0}$ and $a_p \in (\forall s_1:D_{p;1})^{I_0}$. In the same way, V^{post} must have a clause $G_q(t) \wedge \forall t_1:D_{q;1}(t, \ldots) \wedge \ldots \wedge \forall t_v:D_{q;v}(t, \ldots)$ in V^{post}, and an element a_q such that $(a_q, b_q) \in t_1^{I_0}$, $a_q \notin G_q^{I_0}$ and $a_q \in (\forall t_1:D_{q;1})^{I_0}$. In this case we can prove the following lemma.

Lemma 5.4.24. There exist two models I_p and I_q of (V^{pre}, A) in M where:

1. $b_p \in D^{I_p}_{p;1}$ and $b_p \in C^{I_p}_p$.

2. $b_q \in D^{I_q}_{q;1}$ and $b_q \in C^{I_q}_q$.

Proof. Recall $a_p \in G^{I_0}_p$, where $G_p \sqcap \forall s_1.D_{p;1} \sqcap \cdots \sqcap \forall s_u.D_{p;u}$ is a clause in $\mathcal{V}^{post}$.

We note that $u \neq 2$, because if $u = 1$, then we have $(\mathcal{V}^{post}) \models D_{p;1} \sqcap \forall s_1.G_p$, and we would have that $\mathcal{V}^{post} \models \neg D_p \sqcap \forall s_1.G_p \sqcap C_p$. By the assumption made in 6 in Case 3, we get that $b_p \in C^{I_0}_p$ which contradicts our assumption.

Consider all models of $(\mathcal{V}^{pre}, \mathcal{A})$ from $\mathcal{M}$ which are extended from I_0. Suppose in every model $I \in \mathcal{M}$ we have $a_p \in (\forall s_1.D_{p;1})^I$ and $a_p \notin (\forall s_i.D_{p;i})^I$ with $2 \leq i \leq u$. In this case, we must have $(\mathcal{V}^{pre}, \mathcal{A}) \models D_{p;i} \sqcup C_{p;i}$, and domain elements $s_{p;i}$ such that $(a_p, b_{p;i}) \in s^I_i$ and $b_{p;i} \notin C^I_{p;i}$. Then $\mathcal{V}^{pre} \models D_{p;1} \sqcap \forall s_1.(G_p \sqcap \bigsqcup_{i=2}^{u} \forall s_i.C_{p;i})$. In this derivation we did not rely on clauses where D_p occur. So, by construction we have $\mathcal{V}^{post} \models D_{p;1} \sqcap \forall s_1.(G_p \sqcap \bigsqcup_{i=2}^{u} \forall s_i.C_{p;i})$. We also have $\mathcal{V}^{post} \models \neg D_p \sqcap \neg D_{p;1} \sqcap C_p$ by assumption. So, $\mathcal{V}^{post} \models \neg D_p \sqcap \forall s_1.(G_p \sqcap \bigsqcup_{i=2}^{u} \forall s_i.C_{p;i}) \sqcap C_p$. From 6 we get that $b_p \in (\forall s_1.(G_p \sqcap \bigsqcup_{i=2}^{u} \forall s_i.C_{p;i}) \sqcap C_p)^{I_0}$. Since $a_p \notin (G_p \sqcap \bigsqcup_{i=2}^{u} \forall s_i.C_{p;i})^{I_0}$ we get that $b_p \in C^{I_0}_p$ which contradicts the assumption that $b_p \notin C^{I_0}_p$.

We conclude from the above that there are models $I \in \mathcal{M}$ where a_p is in the interpretation of $\forall s_i.D_{p;i}$ for some values of i with $2 \leq i \leq u$. We transform all such models by removing every s_1 successor of a_p from the interpretation of $D_{p;1}$, and denote the transformed models by $\mathcal{N}$. We can see that the models of $\mathcal{N}$ are models of $(\mathcal{V}^{pre}, \mathcal{A})$ because in each model $I \in \mathcal{N}$ we have $a_p \in (\forall s_i.D_{p;i})^I$ for at least one value of i with $2 \leq i \leq u$.

If b_p is in the interpretation of C_p in every model from $\mathcal{N}$, then one of the following must be true.

1. $b_p \in \mathrm{ind}(\mathcal{A})$, and $\mathcal{A} \models C_p(b_p)$.

2. $\mathcal{V}_0 \models \neg C'_p \sqcap C_p$, and for every model I of $(\mathcal{V}_0, \mathcal{A})$ we have $C'_p(b_p)$.

3. $(a_p, b_p) \in s^I_i$ for some value of i with $2 \leq i \leq u$ and every $I \in \mathcal{N}$, where: $D_p \sqcap \neg D_{p;i} \sqcap C_{p;i}$ are clauses in $\mathcal{V}^{pre}$, and $\mathcal{V}_0 \models C_{p;i} \sqcup C_p$.

The conditions 1 and 2 imply that $b_p \in C^{I_0}_p$ which contradicts the earlier assumption that $b_p \notin C^{I_0}_p$. When the side conditions of 3 are satisfied, we must have had the following conclusion obtained from a UB inference.

$$\neg D_p \sqcap \forall s_1.G_p \sqcap \cdots \sqcap \forall s_u.G_p \sqcap C_p \sqcap C_{p;2} \sqcap \cdots \sqcap C_{p;u} \tag{5.298}$$

The clause (5.298) must be in V^{post}, because we require ordering of the inferences that computes it before performing the LB inference whose conclusion has as a top level positive literal. (5.298) implies that $b_p \in C_p^{I_0}$, which contradicts our initial assumption.

Therefore, there must exist a model I_p of V^{pre}, A where $b_p \notin C_p^{I_p}$. The same argument shows that there is a model I_q of (V^{pre}, A) over D^{I_0} where $b_q \notin C_q^{I_q}$. □

The result established by Lemma 5.4.24 is that some but not all models of (V^{pre}, A) have b_p in the interpretation of C_p. In the same way some but not all models of (V^{pre}, A) have b_q in the interpretation of C_q. We use this result in the following Lemma to show that if $(V^{pre}, A) \models q(\bar{a})$ then $I_0 \models q(\bar{a})$. With this we will have the contradiction described before De nition 5.4.21 established.

Lemma 5.4.25. Let $q(\bar{x})$ be a conjunctive query, and $\bar{a}$ an answer to the query in (V^{pre}, A). That is, $(V^{pre}, A) \models q(\bar{a})$. We have $I_0 \models q(\bar{a})$.

Proof. Assume $I_0 \not\models q(\bar{a})$. Let J be the model obtained from I_0 by performing the transformations in Cases 1, 2, 3, and 4 when the relevant conditions are true. When the conditions of Case 5 are true for a domain element b, we remove b from the interpretation of D_1, and add every r_p successor of a to the interpretation of D_p. Denote by a the r_p successors of a which are added to the interpretation of D_p, and are not in the interpretation of C_p. The models I_0 and J are bisimilar with respect to $sig(O) \cap F$ and $ind(A)$, because all applied transformations change the interpretations of de ners which are not in the set $sig(O) \cap F$. So, we have that $J \not\models q(\bar{a})$. If $a = \emptyset$, then only the transformations in Cases 1, 2, 3, and 4 were performed, and we have J a model of (V^{pre}, A). Moreover, the following contradicting remarks would be true about J:

1. $J \models q(\bar{a})$, because $J \models (V^{pre}, A)$ and $(V^{pre}, A) \models q(\bar{a})$.
2. $J \not\models q(\bar{a})$, because I_0 and J are bisimilar with respect to $sig(O) \cap F$ and $ind(A)$ as explained above.

The contradiction implies that if $a = \emptyset$ then $I_0 \models q(\bar{a})$.

Suppose $a \neq \emptyset$. The model J is not a model of (V^{pre}, A) because the domain elements added to the interpretation of D_p are not in the interpretation of C_p. We can nevertheless show that $J \models q(\bar{a})$.

Recall that $(V^{pre}, A) \models q(\bar{a})$, where $q(\bar{x})$ is a conjunctive query, $\bar{x} = x_1, \ldots, x_k$ are the answer variables, and $\bar{a} = a_1, a_2, \ldots, a_k$ is an answer to the query. There must be a mapp with $p(x_i) = a_i$ for $1 \leq i \leq k$, and for every anonymous variable y in q, and

every model I of (V^{pre}, A), p maps y to an element in Δ^I such that the following is true.

1. If $A(u)$ is an atom in q, then $p(u) \in A^I$.
2. If $r(u,v)$ is an atom in q, then $(p(u), p(v)) \in r^I$.

From the construction of J we have $J \models q(a)$ implies the following conditions are true.

1. u is a variable in q, and
2. For every model $I \in M$, there is a map p_I as above that maps u to $b \in a$.

We prove the above implication by contradiction. Let I be a model from M. We have $I \models q(a)$, because $I \models (V^{pre}, A)$. Therefore, there must be a map p that maps x_i to a_i for every $1 \leq i \leq k$, such that the conditions noted above about p are true. Suppose that for every variable u occurring in q, we have $p(u) \notin a$. Then, p maps every variable in q to elements in Δ^J. That is, $J \models q(a)$ which contradicts the assumption above.

Now, Lemma 5.4.24 shows that there exist a model M in where $a = \emptyset$. Thus, from the contrapositive of the above implication, we get that $J \not\models q(a)$.

Since I_0 and J are bisimilar with respect to $sig(O) \cap F$ and $ind(A)$, we have $I_0 \models q(a)$. □

It remains to consider the elimination of the clauses from $D^{\cup}$ where two or more negative definers occur. This step is performed after all resolution inferences have been performed. Recall that clauses used in resolution inferences are removed when resolution has been performed. Therefore, the clauses removed by this step have not been used in previous resolution inferences. Suppose a clause $C = D_1 \sqcup \cdots \sqcup D_n \sqcup C'$ is removed. No LB inferences could have been performed to obtain clauses where the definers $D_1, \ldots, D_n$ occur positively as top-level literals. If such clauses were obtained, then resolution would have been performed, and C would have been removed.

A clause $C = G \sqcup \forall r_1.D_1 \sqcup \cdots \sqcup \forall r_n.D_n$ cannot be used in LB inferences if one of the following is true.

1. A negative existential definer D occurs in G.
2. There are two definers, say D_1 and D_2, where $V^q_{ALC} \not\models D_1 \sqsubseteq C_1$ and $V^q_{ALC} \not\models D_2 \sqsubseteq C_2$ for any concepts $C_1 \neq \top$ and $C_2 \neq \top$.

When the clauses which use two or more negative de ners have been eliminated all remaining de ners would be simple de ners. Consequently, they will be eliminated by the simple de ner elimination process. If Condition 2 is satis ed, then the two de ners D_1 and D_2 would be eliminated using puri cation, and C will be eliminated as it becomes a tautology. If Condition 1 is satis ed, and for every de ner D_i, $\mathcal{V}^q_{\mathrm{ALCI}}$ has a clause $\neg D_i \sqcup C$ where $1 \leq i \leq n$, then the de ners will be eliminated by Non-cyclic De ner Elimination inferences.

Denote by $\mathbf{D}_e$ the set of Type 2 universal de ners occurring positively in a clause $C = G \sqcup \neg\exists r_1.D_1 \sqcup \cdots \sqcup \forall r_n.D_n$ satisfying Condition 1. We assume that the elimination of clauses from $\mathcal{D}^u$ and the elimination of simple de ners have been performed in two consecutive steps.

The rst step removes the clauses of $\mathcal{D}^u$ that use de ners from $\mathbf{D}_e$ with negative polarity, and eliminates simple de ners immediately after the clauses have been removed. Let $\mathcal{V}_e$ be the snapshot of the ontology obtained after eliminating the simple de ners.

The second step removes the remainder of the clauses in $\mathcal{D}^u$, and puri es the de ners. Let $\mathcal{V}_>$ be the snapshot of $\mathcal{V}^q_{\mathrm{ALCI}}$ obtained after the step has completed.

We consider the rst step. Let $\mathcal{V}_1$ be the snapshot of $\mathcal{V}^q_{\mathrm{ALCI}}$ before removing a clause from $\mathcal{D}^u$ that uses a de ner from $\mathbf{D}_e$ with negative polarity, and $\mathcal{V}_2$ be the snapshot after removing the clause. We show that $\mathcal{V}_1$ and $\mathcal{V}_2$ are query inseparable with respect to $\mathrm{sig}(\mathcal{O}) \cap \mathbf{F}$ by showing the following.

1. Every model $\mathcal{I}$ of $(\mathcal{V}_1, \mathcal{A})$ is a model of $(\mathcal{V}_2, \mathcal{A})$.

2. Every model $\mathcal{I}$ of $(\mathcal{V}_2, \mathcal{A})$ can be transformed to a model $\mathcal{J}$ of $(\mathcal{V}_1, \mathcal{A})$ such that $\mathcal{I}$ and $\mathcal{J}$ are bisimilar with respect to $\mathrm{sig}(\mathcal{O}) \cap \mathbf{F}$ and $\mathrm{ind}(\mathcal{A})$.

To prove Item 1, let $\mathcal{I}$ be a model of $(\mathcal{V}_1, \mathcal{A})$. So, $\mathcal{I} \models \mathcal{V}_1$ and $\mathcal{I} \models \mathcal{A}$. Since $\mathcal{V}_2 \subseteq \mathcal{V}_1$ we have $\mathcal{I} \models \mathcal{V}_2$. Hence $\mathcal{I} \models (\mathcal{V}_2, \mathcal{A})$.

To prove Item 2, let $\mathcal{I}$ be a model of $(\mathcal{V}_2, \mathcal{A})$. We must have the following clauses in $\mathcal{V}_1$.

$$G \sqcup \exists r.D \tag{5.299}$$

$$\neg D \sqcup H \sqcup \forall r_1.D_1 \sqcup \cdots \sqcup \forall r_n.D_n \tag{5.300}$$

$$\neg D_1 \sqcup \neg D_1' \sqcup C_1 \tag{5.301}$$

$$\neg D_i \sqcup \neg D_i' \sqcup C_i \tag{5.302}$$

$$\vdots \tag{5.303}$$

$$: D_n \sqcup : D_n' \sqcup C_n \tag{5.304}$$

$$\tag{5.305}$$

where $1 \leq i \leq n$, $G, H, C_1, \ldots, C_n$ are $\mathcal{ALC}$ concepts, $D_1, \ldots, D_n$ and $D_1', \ldots, D_n'$ are complex universal de ners, and D is an existential de ner. Suppose V_2 is obtained from V_1 by removing (5.307). If for every $e \in D_i^{\mathcal{I}}$ we have $e \notin D_i^{\mathcal{J}}$ or $e \in C_i^{\mathcal{I}}$, then $\mathcal{I} \models : D_i \sqcup : D_i' \sqcup C_i$, and $\mathcal{I} \models (V_1, A)$. Otherwise, we must have $d_1, d_2, e_i \in D^{\mathcal{I}}$ where:

$$d_1 \notin G^{\mathcal{I}} \tag{5.306}$$

$$d_2 \in D^{\mathcal{I}} \tag{5.307}$$

$$(d_1, d_2) \in r^{\mathcal{I}} \tag{5.308}$$

$$d_2 \in (\forall r_i . D_i)^{\mathcal{I}} \tag{5.309}$$

$$(d_2, e_i) \in r_i^{\mathcal{I}} \tag{5.310}$$

$$e_i \notin C_i^{\mathcal{I}} \tag{5.311}$$

$$e_i \in D_i^{\mathcal{J}} \tag{5.312}$$

We extend $\mathcal{I}$ to $\mathcal{J}$ such that $D^{\mathcal{J}} = D^{\mathcal{I}} \cup \{d_2', e_i'\}$, and the following is true.

1. $(d_1, d_2') \in r^{\mathcal{J}}$,
2. $d_2' \in D^{\mathcal{J}}$
3. $d_2' \in A^{\mathcal{J}}$ if and only if $d_2 \in A^{\mathcal{I}}$ for every concept name A,
4. $e_i' \in A^{\mathcal{J}}$ if and only if $e_i \in A^{\mathcal{I}}$ for every concept name A,
5. $d_2' \in D^{\mathcal{J}}$ if and only if $d_2 \in D^{\mathcal{I}}$ for every de ner D,
6. $e_i' \in D^{\mathcal{J}}$ if and only if $e_i \in D^{\mathcal{I}}$ for every de ner $D \neq D_i'$,
7. $(d_2', e) \in s^{\mathcal{J}}$ if and only if $(d_2, e) \in s^{\mathcal{I}}$, for every domain element $e \neq e_i$ and role name s,
8. $(d_2', e_i') \in r_i^{\mathcal{J}}$,
9. $(e_i', f) \in s^{\mathcal{J}}$ if and only if $(e_i, f) \in s^{\mathcal{I}}$, for every domain element f and role name s, and
10. $d_2 \notin D^{\mathcal{J}}$.

The transformation above copies d_2 to d_2', and e_i to e_i'. Then, it removes d_2 from the interpretation of D, and e_i' from the interpretation of D_i'. We can see that $J \models (V_1; A)$. Moreover, J and I are bisimilar with respect to $sig(O) \cap F$ and $ind(A)$, because the transformation creates copies of elements which preserves the bisimulation, and changes the interpretations of D and D_i' which are not in $sig(O) \cap F$.

From the above discussion we get that V_1 and V_2 are query inseparable with respect to $sig(O) \cap F$. Since we eliminate simple de ners from V_2, and the elimination of simple de ners preserves query inseparability with respect to $sig(O) \cap F$, we get from Lemma 3.5.7 the required result.

We consider the second step that removes the remainder of clauses that do not use de ners from D_e, and eliminates the simple de ners. Let V_1 be the snapshot of V_e before removing the clauses D_e. We can view V_1 as follows.

$$V_1 = V_0 \cup V_8 \cup V_D \cup V_D^2 \tag{5.313}$$

where:

1. V_0 is the set of clauses that do not use de ners.

2. V_8 is the set of clauses of the form $\top \sqsubseteq \forall\, r_1.D_1 \sqcap \cdots \sqcap \forall\, r_n.D_n$ where $D_1, \ldots, D_n$ are de ners.

3. V_D is the set of clauses of the form $D \sqsubseteq C$ where D is a de ner, and C is a concept that does not use de ners.

4. V_D^2 is the set of clauses of the form $D_1 \sqcap D_2 \sqsubseteq C$ where D_1 and D_2 are de ners, and C is a concept.

The output of this step is V_0 because V_D^2 will be removed, and the de ners, whose negative occurrences are eliminated by the removal of V_D^2 clauses, are puri ed. This will consequently make the clauses of V_8 where the puri ed de ners occur tautologous. We obtain the same output in a different way that we show to preserve query inseparability between V_1 and V_0 with respect to $sig(O) \cap F$.

Recall that V_1 and the input ontology O are query inseparable with respect to $F(O) \cap F$. Let N be the set of de ners satisfying the conditions in 2. These would be the de ners not occurring in the clauses of V_D. Assume there are n de ners in N. Let N_t be a set of n concept names such that $N_t \cap N_c = \emptyset$, $N_t \cap N_d = \emptyset$, and $N_t \cap N_r = \emptyset$.

Consider the ontology $\mathcal{V}_2 = \mathcal{V}_1 \cup \mathcal{D}_t$, where $\mathcal{D}_t$ is defined as follow.

$$\mathcal{D}_t = \{\ \exists\, D_i.t \sqsubseteq S_i \mid D_i \in \mathbf{N}_C, S_i \in \mathbf{N}_t \text{ and } 1 \leq i \leq n\} \tag{5.314}$$

We show that $\mathcal{V}_1$ and $\mathcal{V}_2$ are model inseparable with respect to $sig(\mathcal{V}_1)$. To prove the statement we show that the following are true.

1. Every model of $\mathcal{V}_2$ is a model of $\mathcal{V}_1$.
2. Every model $\mathcal{I}$ of $\mathcal{V}_1$ can be extended to a model $\mathcal{J}$ of $\mathcal{V}_2$ such that $\mathcal{I}$ and $\mathcal{J}$ $sig(\mathcal{V}_1)$-coincide.

Let $\mathcal{I}$ be a model of $\mathcal{V}_2$, then $\mathcal{I}$ is a model of $\mathcal{V}_1$ because removing axioms does not invalidate models.

Let $\mathcal{I}$ be a model of $\mathcal{V}_1$, and $\mathcal{J}$ the model extended from $\mathcal{I}$ such that $S_i^{\mathcal{J}} = D^{\mathcal{J}}$ for every $S_i \in \mathbf{N}_t$ with $1 \leq i \leq n$. We have $\mathcal{J} \models \mathcal{V}_2$. Moreover, we have that $\mathcal{I}$ and $\mathcal{J}$ $sig(\mathcal{O}) \cap \mathbf{F}$-coincide, because the interpretations of the symbols in $\mathcal{V}_1$ did not change in $\mathcal{J}$.

Since $\mathcal{V}_1$ and $\mathcal{V}_2$ are model inseparable with respect to $sig(\mathcal{O}) \cap \mathbf{F}$, we get from Lemma 3.5.10 that the two ontologies are query inseparable with respect to $sig(\mathcal{O}) \cap \mathbf{F}$.

A difference between $\mathcal{V}_2$ and $\mathcal{V}_1$ is that LB inferences can be performed on the clauses of $\mathcal{V}_2$. Let $G_i.t \sqsubseteq \forall\, r_1.D_1 \sqcup \cdots \sqcup \forall\, r_n.D_n$ be a clause in $\mathcal{V}_8$. A set of LB inferences would be performed to obtain conclusions of the form $D_i.t \sqsubseteq E$, where E is a concept of the form $\forall r.C$, and $\forall r.S$ is a disjunct in C, where $r \in \{r_1, \ldots, r_n\}$ and $S \in \mathbf{N}_t$. With the LB conclusions obtained, we resolve the obtained conclusions and the clauses in $\mathcal{D}^u$. When resolution has been exhaustively performed, the premises of the resolution inferences are removed. This will remove the clauses in the two sets $\mathcal{V}_D$ and $\mathcal{V}_D^2$.

Each computed resolvent would have a disjunct of the form $\forall r.C$ as described above. Let $\mathcal{V}_3$ be the ontology obtained, and denote by $\mathcal{V}_0$ the set of the clauses computed by resolution. $\mathcal{V}_3$ can be viewed as follows.

$$\mathcal{V}_3 = \mathcal{V}_0 \cup \mathcal{V}_8 \cup \mathcal{V}_q \tag{5.315}$$

As we only perform definer elimination, the two ontologies $\mathcal{V}_2$ and $\mathcal{V}_3$ are query inseparable with respect to $sig(\mathcal{O}) \cap \mathbf{F}$.

Next, we forget the symbols $S \in \mathbf{N}_t$ from $\mathcal{V}_3$, and purify the definers occurring in $\mathcal{V}_8$. The definers are eliminated using definer purification because they occur only positively in $\mathcal{V}_8$. When the definers are purified the clauses in $\mathcal{V}_8$ are removed as they

become tautologies. The symbols $S \subseteq N_t$ are eliminated by puri cation because they occur only positively in V_q. The puri cation of the symbols S makes the clauses of V_q tautologies. To see this consider the clauses obtained by resolution. As discussed above each resolvent uses a disjunct of the form $\forall r.C$ where $\forall r.S$ is a disjunct in C. If $\top$ is substituted for S by puri cation, the concept $\forall r.S$ will be equivalent to $\top$. Furthermore, the disjunct $\forall r_i.C$ will be equivalent to $\top$, and the resolvent will be a tautology.a Let V_4 be the ontology obtained. We have $V_4 = V_0$ because V_8 and V_q are removed by the puri cation inferences, and V_0 is not impacted because de ners and symbols from N_t do not occur in V_0. Since puri cation preserves model inseparability, we get from Lemma 3.5.10 that V_3 and V_4 are query inseparable with respect to $sig(O) \cap F$.

We now have the following.

1. V_1 and V_2 are query inseparable with respect to $sig(O) \cap F$.
2. V_2 and V_3 are query inseparable with respect to $sig(O) \cap F$.
3. V_3 and V_4 are query inseparable with respect to $sig(O) \cap F$.

From Lemma 3.5.7 we get that V_1 and V_4 are query inseparable with respect to $sig(O) \cap F$. Since V_0 and V_4 are logically equivalent, we get the required result that V_0 and V_1 are query inseparable with respect to $sig(O) \cap F$. This completes the proof for the fth component in the elimination process of Type 2 and consequently the proof of Theorem 5.4.19.

We have now presented a query customization that is pluggable to our ne-grained forgetting framework. The customization and the framework together provide a rst query forgetting method for $\mathcal{ALC}$ ontologies. The starting point of this customization is the method presented in Section 4.8 whereby ne-grained forgetting views in-between the deductive and the semantic forgetting views are extracted. It thus asserts the granularity of $\mathcal{O}^{int}$, and shows one example of how our ne-grained framework can provide exciting opportunities. So far we have been considering the content of the forgetting view. In the next chapter we move on to a different challenge centered around the syntactic representation of the semantic forgetting view, a qualitative feature that is rather important to users who maintain strict syntactic guidelines for their ontologies.

Chapter 6

Semantic Forgetting

In Chapter 4 we introduced a fine-grained forgetting framework which computes a representation of the semantic forgetting view of the input ontology with respect to the forgetting signature. The computed representation is expressed in a normal form that allows for extracting fine-grained forgetting views with customized content.

When the representation of the semantic forgetting view is intended to be used without further customization to its content, a different forgetting method can be used to obtain extra benefits. In this chapter, we present a new resolution-based forgetting method that computes a representation of the semantic forgetting views of the input ontology with respect to the forgetting signature. The main contribution of the forgetting method is preserving the syntactic structure of the input ontology in the computed representation.

As discussed in Chapter 1, preserving the syntactic structure of the input ontology is an important feature to users, because it improves the readability of the forgetting views, and enables the users to maintain their syntactic guidelines. Not preserving the syntactic structure is a common criticism to resolution based methods. The content of this chapter is the first treatment of this topic in the context of description logics.

Let us recall the definition of semantic forgetting from Chapter 3. Semantic forgetting is based on the notion of model inseparability. Two ontologies are model inseparable with respect to a set S of symbols, if the models of the two ontologies coincide on the interpretations of the symbols from S. Let $\mathcal{O}$ and $\mathcal{V}^s$ be two ontologies. $\mathcal{V}^s$ is a semantic forgetting view of $\mathcal{O}$ with respect to a forgetting signature F, if $\mathcal{V}^s$ uses only symbols from $sig(\mathcal{O}) \cap F$, and $\mathcal{O}$ and $\mathcal{V}^s$ are model inseparable with respect to $sig(\mathcal{O}) \cap F$. The formal definition of model inseparability is given in Definition 3.5.4, and the formal definition of semantic forgetting is given in Definition 3.5.16.

Similar to the previous two chapters, we use de ners to write the input ontology in a suitable form for the rules of our calculus. When the forgetting symbols have been eliminated, we eliminate the introduced de ners. The elimination of all de ners may not be possible, which is expected given that semantic forgetting views may not exist in a rst-order language as we explained in Chapter 3. If the output of the method uses de ners, then the output is a representation of the semantic forgetting view that is model inseparable from the input ontology with respect to the non-forgotten signature. If the output does not use de ners, then it is a semantic forgetting view.

The chapter is structured as follows. In Section 6.1 we explain what we mean by the syntactic structure of an ontology. In Section 6.2 we outline the three main stages of the forgetting method, and explain them in Sections 6.3, 6.4, and 6.5. Finally, we compare in Section 6.6 the forgetting method presented in this chapter with the method explained in Chapter 4, and discuss how it preserves the syntactic structure of the input ontology.

6.1 The Syntactic Structure of an Ontology

We begin by intuitively explaining what we mean by the syntactic structure of an ontology.

The syntactic structure of an $\mathcal{ALC}$ concept C denotes the concepts and logical operators used in C, their nesting, their polarities, and their orderings. For instance, consider the two concepts C_1 and C_2 below.

$$C_1 = \exists r.(B \sqcap E) \tag{6.1}$$

$$C_2 = (\exists r.B) \sqcap (\forall r.E) \tag{6.2}$$

The two concepts C_1 and C_2 are logically equivalent. However, they have different syntactic structures. The concept C_2 uses the concepts $\exists r.B$ and $\forall r.E$, which are not used in C_1, and the concept C_1 uses the concept $B \sqcap E$ which is not used in C_2. Further, the concept C_1 is a disjunction of a literal A, and a conjunction $B \sqcap E$, whereas the concept C_2 is a conjunction of two disjunctions.

Consider the four concepts below.

$$C_3 = \exists r.A \tag{6.3}$$

$$C_4 = \neg\forall r.\neg A \tag{6.4}$$

$$C_5 = A \sqcap B \sqcap E \tag{6.5}$$

$$C_6 = A \sqcap E \sqcap B \tag{6.6}$$

The concepts C_3 and C_4 are logically equivalent. However, they are syntactically different. The concept C_3 is an existential role restriction. The role filler in C_3 is the concept A. The concept C_4 is a nesting of a negation and a universal role restriction. The role filler in C_4 is also a nesting of a negation and the concept A.

Likewise, C_5 and C_6 are logically equivalent. But, they are syntactically different because they use a different ordering of the conjuncts.

The syntactic structure of an axiom denotes the subsumption or the equivalence connective of the axiom, as well as the syntactic structure of the concepts on both sides of the connective. Consider the following two axioms.

$$a_1 = C \equiv D \tag{6.7}$$

$$a_2 = C \sqsubseteq D \tag{6.8}$$

$$a_3 = D \equiv C \tag{6.9}$$

The two axioms a_1 and a_2 are logically and syntactically different, because a_1 uses the equivalence connective and a_2 uses subsumption. The two axioms a_1 and a_3 are logically equivalent. However, they are syntactically different because C occurs on the left hand side of the equivalence in a_1, and D occurs on the right hand side, whereas C occurs on the right hand side of the equivalence in a_3, and D occurs on the left hand side.

Consider the following two axioms.

$$a_4 = A \sqcup B \sqsubseteq C \tag{6.10}$$

$$a_5 = A \sqcup C \sqsubseteq B \tag{6.11}$$

The axioms a_4 and a_5 are syntactically different because the concept on the right hand side of the subsumption connective in a_4 is syntactically different from the concept on the right hand side of the subsumption in a_5.

In the same way, the notion of syntactic structure can be lifted to ontologies. Consider the following two ontologies.

$$\mathcal{O}_1 = \{A \equiv B\} \tag{6.12}$$

$$\mathcal{O}_2 = \{A \sqsubseteq B, B \sqsubseteq A\} \tag{6.13}$$

The two ontologies are logically equivalent. But, they are syntactically different because $\mathcal{O}_1$ uses an equivalence axiom, while $\mathcal{O}_2$ uses two subsumption axioms to represent the same meaning of the equivalence axiom in $\mathcal{O}_1$.

Consider the following ontologies.

$$\mathcal{O}_3 = \{A \equiv B \sqcap C\} \tag{6.14}$$

$$\mathcal{O}_4 = \{A \equiv C \sqcap B\} \tag{6.15}$$

As before, the two ontologies are logically equivalent. However, they are syntactically different, because the two axioms $A \equiv B \sqcap C$ and $A \equiv C \sqcap B$ are syntactically different.

When forgetting symbols from an ontology, we necessarily change the syntactic structure. The aim is to change the structure in the least damaging way. For instance, consider the following ontology $\mathcal{O}$.

$$A \sqsubseteq B \sqcap C \sqcap E \tag{6.16}$$

and let $\mathbf{F} = \{C\}$ be a forgetting signature. Two forgetting views $\mathcal{V}_1$ and $\mathcal{V}_2$ of $\mathcal{O}$ with respect to $\mathbf{F}$ can be computed. The forgetting view $\mathcal{V}_1$ consists of the following axiom.

$$A \sqsubseteq C \sqcap E \tag{6.17}$$

The forgetting view $\mathcal{V}_2$ consists of the following axioms.

$$\top \sqsubseteq \neg A \sqcup C \tag{6.18}$$

$$\top \sqsubseteq \neg A \sqcup E \tag{6.19}$$

The two forgetting views are not structurally equivalent to $\mathcal{O}$, because they do not contain the symbol B. However, the syntactic structure of $\mathcal{V}_1$ is more resembling of the syntactic structure of $\mathcal{O}$ than $\mathcal{V}_2$. The method in Chapter 4 would compute $\mathcal{V}_2$. Our aim in this chapter is to devise a forgetting method that computes $\mathcal{V}_1$.

Comparing $\mathcal{O}$ to $\mathcal{V}_1$ and $\mathcal{V}_2$ above is easy because $\mathcal{O}$ contains only one axiom. When the ontology and the forgetting view contain many axioms, there is the question of which axioms to compare. In the forgetting view, there can be axioms which are copied from the input ontology because they did not use forgetting symbols. The

comparison of these axioms is relatively easy, because we can compare them to the corresponding axioms in the input ontology. There may, however, be an axiom in the forgetting view that is computed by the forgetting method. We compare the set of axioms used by the method to compute it. Consider the following ontology $\mathcal{O}$.

$$A \sqsubseteq B \sqcup C \sqcup D \tag{6.20}$$
$$D \sqsubseteq E \sqcup F \tag{6.21}$$
$$F \sqsubseteq G \tag{6.22}$$

Let $\mathbf{F} = \{D\}$ be a forgetting signature. The ontology $\mathcal{V}$ below is a semantic forgetting view of $\mathcal{O}$ with respect to $\mathbf{F}$.

$$A \sqsubseteq B \sqcup C \sqcup (E \sqcup F) \tag{6.23}$$
$$F \sqsubseteq G \tag{6.24}$$

The axiom (6.24) is copied from (6.22) because (6.22) does not use forgetting symbols. So, we compare (6.24) to (6.22). The axiom (6.23) is computed from the two axioms (6.20) and (6.21). So, we compare (6.23) to (6.20) and (6.21). The axiom (6.23) resembles (6.20) because it has the concept A on the left hand side of the subsumption, and the concept $B \sqcup C$ on the right hand side. It also resembles (6.21) because it has the concept $E \sqcup F$ on the right hand side of the subsumption. The concepts used from (6.20) and (6.21) occur with the same polarity in (6.23). Also, (6.23) does not use a concept that does not occur in (6.20) or (6.21).

6.2 Outline of the Forgetting Method

We present the outline of the forgetting method. The inputs of the method are an $\mathcal{ALC}$ ontology, and a forgetting signature comprising concept names. The output is a representation of the semantic forgetting views of the input ontology with respect to the forgetting signature. The method uses a resolution calculus, and may enhance the computed semantic view with de ners to capture it. The aim is to preserve the syntactic structure of the input ontology in the forgetting view as much as possible.

The forgetting process is made of the three consecutive stages depicted in Figure 6.1. The input of the rst stage is the ontology $\mathcal{O}$ and the forgetting signature $\mathbf{F}$. The output is an ontology $\mathcal{O}^{\vee}$ where concept symbols from the forgetting signature,

Figure 6.1: Structure of the semantic forgetting method.

that are de ned by equivalence axioms, have been eliminated. These are the concept names occurring on one side of an equivalence axiom, with all other concepts in the signature of the axiom occurring on the other side.

The second stage eliminates non-de ned forgetting symbols, that is, the forgetting symbols not occurring in an equivalent axiom as described in De nition 6.3.1. The input to this stage is the ontology $\mathcal{O}^{V}$, and the output is the ontology $\mathcal{O}^{D}$. The ontology $\mathcal{O}^{D}$ does not use any symbol from the forgetting signature $\mathbf{F}$, but may use de ners.

The third stage of the method eliminates the de ners from $\mathcal{O}^{D}$, and obtains the nal representation of the semantic view $\mathcal{V}^{S}$. As discussed earlier in Chapter 4, semantic forgetting views of $\mathcal{ALC}$ ontologies are not in general expressible in $\mathcal{ALC}$. However, it can be expressed when enhanced with de ners.

6.3 Stage 1: de nition substitution

The aim of the rst stage of the forgetting method is eliminating de ned forgetting symbols from the input ontology $\mathcal{O}$. De ned forgetting symbols are de ned by the following de nition.

De nition 6.3.1. Let $\mathcal{O}$ be an ontology, and $\mathbf{F}$ a forgetting signature. A concept symbol $B \in \mathbf{F}$ is de ned if an axiom of the form $B \equiv C$, or $C \equiv B$, exists in $\mathcal{O}$, and $B \notin sig(C)$.

Algorithm 8 describes the process of eliminating de ned forgetting symbols. The lines from 1 to 3 iterate over the forgetting symbols, and check if the iterated forgetting symbol B is de ned.

Line 4 eliminates the forgetting symbol by substituting C for B everywhere in the ontology, if an axiom of the form $B \equiv C$ or $C \equiv B$ is present. We call this step de nition substitution.

Line 5 moves the execution of the algorithm to Line 1 where a new forgetting symbol is iterated.

Algorithm 8: Elimination process of defined forgetting symbols.

Input: Ontology $\mathcal{O}$, forgetting signature $\mathbf{F}$
Output: Ontology $\mathcal{O}_v$ that does not use defined forgetting symbols
Initialize: $\mathcal{O}_v := \mathcal{O}$

```
1 for B ∈ F do
2     for C ∈ O_v do
          // Check if B is a defined forgetting symbol.
3         if C has the form B ⊑ C or C ⊑ B, and B ∉ sig(C) then
              // Substitute C for B.
4             O_v := O_v[B/C]
              // Eliminate the next defined forgetting symbol.
5             Break
```

Return: $\mathcal{O}_v$

Example 6.3.2. Consider the following ontology $\mathcal{O}$.

$$A \sqsubseteq B \tag{6.25}$$

$$B \sqsubseteq C \sqcap D \tag{6.26}$$

Let $\mathbf{F} = \{B\}$ be the forgetting signature. The concept B is defined because it occurs in the axiom $A \sqsubseteq B$. We eliminate B by definition substitution, which substitutes A for B everywhere in the ontology. The following ontology $\mathcal{O}_v$ would be obtained.

$$A \sqsubseteq A \tag{6.27}$$

$$A \sqsubseteq C \sqcap D \tag{6.28}$$

The axiom (6.27) is a tautology and can be eliminated. Since no more forgetting symbols occur in $\mathcal{O}_v$, we obtain $\mathcal{O}_s$ as the ontology consisting of the axiom (6.28).

6.4 Stage 2: Resolution

The second stage of the method eliminates the remainder of the forgetting symbols from $\mathcal{O}_v$. The output of this stage is the ontology $\mathcal{O}_D$ where all forgetting symbols have been eliminated.

The first step in this stage is purifying the forgetting symbols occurring in $\mathcal{O}_v$ with only one polarity. We purify a forgetting symbol in the same way as done in Chapter 4. If a forgetting symbol occurs only positively in $\mathcal{O}_v$, then we replace it everywhere in

$\mathcal{O}^{\mathsf{v}}$ with $> \sqsubseteq \top$. If it occurs only negatively in $\mathcal{O}^{\mathsf{v}}$, then we replace it with $\bot$.

The second step in the stage is eliminating the forgetting symbols that occur both positively and negatively. This step is iterative. It eliminates one forgetting symbol in each iteration. Suppose B is a forgetting symbol that is being eliminated in one iteration. We extract the axioms that use B and write them in a normal form that we specify next.

The Normal Form

Let $\mathcal{O}^{B}$ be the subset of axioms that use B in the snapshot of $\mathcal{O}^{\mathsf{v}}$ where all forgetting symbols occurring with just one polarity have been purified. We normalize the ontology $\mathcal{O}^{B}$ in a form ready for the resolution calculus. We denote by $\mathcal{O}^{n}$ the union of the set of the normal form of $\mathcal{O}^{B}$, and the set of axioms of $\mathcal{O}^{\mathsf{v}}$ that are not in $\mathcal{O}^{B}$.

The following rewrites are performed in sequence to normalize $\mathcal{O}^{B}$.

1. Rewrite every equivalence axiom $C \equiv D \in \mathcal{O}^{B}$ to the two subsumption axioms $C \sqsubseteq D$ and $D \sqsubseteq C$.

2. Perform structural transformation so that the concepts below role restrictions where B occurs are extracted.

Here we perform structural transformation in a different way to Chapter 4. The following two definitions explain how it is performed.

Definition 6.4.1. Let B be a forgetting symbol and C be a concept such that $B \in \mathsf{sig}(C)$. We say C is a *B-concept* if and only if C has one of the following forms.

1. A concept name B;

2. $B \sqcap E$;

3. $\neg B \sqcap E$; or

4. $E_1 \sqcap E_2$, and E_1 is a B-concept or E_2 is a B-concept,

where E, E_1, E_2 are $\mathcal{ALC}$ concepts, and $\sqcap \in \{\sqcap, \sqcup\}$. Observe that in a B-concept, B cannot occur below existential or universal role restriction.

Definition 6.4.2. Let $\mathcal{O}$ be an ontology, B a forgetting symbol, α an axiom in $\mathcal{O}$, and $C = \alpha|_i$ be a concept of the form $\exists r.E$ where E is a B-concept, r is a role name,

$Q \in \{\exists, \forall\}$, and C is a concept that occurs in a at position i. We say a is structurally transformed when replacing it with axiom $a[i/Qr.D]$, and adding the axiom $def(i, a, D)$, which is defined below, to the ontology.

$$def(i, a, D) = \begin{cases} D \sqsubseteq E & \text{if } pol(a, i) = 1; \\ E \sqsubseteq D & \text{if } pol(a, i) = -1; \end{cases}$$

where D is a fresh concept symbol, that is, $D \notin sig(\mathcal{O})$. We call D, a definer.

We say $\mathcal{O}$ is structurally transformed if the above transformation is applied exhaustively on all axioms until no forgetting symbol appears below role restriction. We use the set N_D to denote the set of definers introduced by structural transformation.

The following example explains the structural transformation operation.

Example 6.4.3. Consider the following ontology $\mathcal{O}$

$$A \sqsubseteq \exists r.(B \sqcap C) \tag{6.29}$$

$$\exists r.B \sqsubseteq E \tag{6.30}$$

$$E \sqsubseteq \neg\exists r.F \tag{6.31}$$

Let $\mathbf{F} = \{B\}$ be a forgetting signature.

We perform structural transformation on the ontology $\mathcal{O}$ by transforming each axiom a in the ontology. Suppose a is the axiom (6.29). The concept $C = B \sqcap C$ which occurs below the existential role restriction is a B-concept because it has the second form specified in Definition 6.4.1. The concept C occurs at position 2.1 in a with a positive polarity. So, a is transformed to the following two axioms:

$$A \sqsubseteq \exists r.D_1 \tag{6.32}$$

$$D_1 \sqsubseteq B \sqcap C; \tag{6.33}$$

where D_1 is a definer.

Next, we transform the axiom (6.30). The concept B occurring below the existential role restriction on the left hand side of the subsumption is a B-concept because it has the first form specified in Definition 6.4.1. It occurs at position 1.1 with negative

polarity. So (6.30) is transformed to the following two axioms.

$$\exists r{:}D_2 \sqsubseteq E \tag{6.34}$$

$$B \sqsubseteq D_2 \tag{6.35}$$

where D_2 is a de ner.

We do not transform (6.31) because it does not contain the symbol B. So the ontology consisting of the following axioms.

$$A \sqsubseteq \exists r{:}D_1$$

$$\exists r{:}D_2 \sqsubseteq E$$

$$E \sqsubseteq \exists r{:}F$$

$$B \sqsubseteq D_2$$

$$D_1 \sqsubseteq B \sqcap C$$

Lemma 6.4.4. Let $\mathcal{O}$ be an ontology, and B a forgetting symbol occurring in $\mathcal{O}$. Let $\mathcal{O}^n$ be the rewrite of $\mathcal{O}$ in the normal form described above. Then, $\mathcal{O} \equiv_{sig(\mathcal{O}) \cap F} \mathcal{O}^n$.

Proof. We can see that de nition substitution preserves model inseparability, because it replaces forgetting symbols with equivalent concepts. Puri cation has been shown previously to preserve model inseparability with respect to the non-puri ed symbols, see Lemma 4.3.3. Moreover, converting equivalence axioms to subsumption axioms preserves logical equivalence. It remains to show that structural transformation preserves model inseparability with respect to $sig(\mathcal{O}) \cap F$.

Consider an application of the transformation in De nition 6.4.2 that extracts B as a concept below role restriction. Let $\mathcal{V}$ is the ontology after applying the transformation. First, we consider the positive case where $C = \exists r{:}E$ is replaced by $Q_r{:}D$, and the axiom $D \sqsubseteq E$ is appended to the ontology, where E is a B-concept. Let $\mathcal{I}$ be a model of $\mathcal{V}$. Since $\mathcal{I} \models D \sqsubseteq E$, we have that $d \in (\exists r{:}D)^{\mathcal{I}}$ implies that $d \in (\exists r{:}E)^{\mathcal{I}}$ for every domain element $d \in \Delta^{\mathcal{I}}$. Thus, $\mathcal{I} \models \mathcal{O}$.

Let $\mathcal{I}$ be a model of $\mathcal{O}$. We construct a new model $\mathcal{J}$ such that $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}}$. We set $\mathcal{J}$ equal to $\mathcal{I}$ on all interpretations and additionally interpret Q_r as: $Q_r^{\mathcal{J}} = \{ y \in E^{\mathcal{I}} \mid (x, y) \in r^{\mathcal{I}} \}$. It follows that for every domain element $d \in \Delta^{\mathcal{J}}$ we have $d \in (Q_r{:}E)^{\mathcal{I}}$ iff $e \in (Q_r{:}D)^{\mathcal{J}}$. Hence, $\mathcal{J} \models \mathcal{O}$ and $\mathcal{J} \models \mathcal{V}$. Also, since all other symbols are interpreted the same in both models, we have $\mathcal{I}$ and $\mathcal{J}$ coincide on all symbols except Q_r.

Second, we consider the negative case where C is replaced by $Q_r{:}D$, and the axiom

$E \sqsubseteq D$ is appended to the ontology. Let $\mathcal{I}$ be a model of $\mathcal{V}$. We need to show that for every domain element d we have $d \in (\neg \exists r.D)^{\mathcal{I}}$ implies $d \in (\neg \exists r.E)^{\mathcal{I}}$. Suppose $d \in (\neg \exists r.D)^{\mathcal{I}}$, then $d \in (\bar{Q} r.\neg D)^{\mathcal{I}}$ where $\bar{Q} = \{\exists, \forall\} \setminus \{Q\}$. Since $\mathcal{I} \models \neg D \sqsubseteq \neg E$ (because $E \sqsubseteq D$ is equivalent to $\neg D \sqsubseteq \neg E$), it follows then that $d \in (\bar{Q} r.\neg E)^{\mathcal{I}}$. Hence, $d \in (\neg \exists r.E)^{\mathcal{I}}$, and so $\mathcal{I} \models \mathcal{O}$.

Let $\mathcal{I}$ be a model of $\mathcal{O}$. We construct a new model $\mathcal{J}$ such that $D^{\mathcal{J}} = D^{\mathcal{I}}$ and set $\mathcal{J}$ equal to $\mathcal{I}$ on all interpretations and additionally interpret E as: $E^{\mathcal{J}} = D^{\mathcal{J}}$. It follows directly that: $\mathcal{J} \models \mathcal{O}$, $\mathcal{J} \models \mathcal{V}$, and $\mathcal{I}$ and $\mathcal{J}$ coincide on all symbols except E.

Repeating the transformation exhaustively on $\mathcal{O}$, it follows that for every model $\mathcal{I}$ of $\mathcal{O}$, there is a model $\mathcal{J}$ of $\mathcal{O}^n$ and vice versa such that $\mathcal{I}$ and $\mathcal{J}$ coincide on all symbols except those from the set N_d.

We have shown the following.

1. Every model $\mathcal{I}$ of $\mathcal{V}$ is a model of $\mathcal{O}$ as well.
2. For every model $\mathcal{I}$ of $\mathcal{O}$, there is a model $\mathcal{J}$ such that $\mathcal{J} \models \mathcal{V}$, and $\mathcal{I}$ and $\mathcal{J}$ coincide on all symbols except the symbols from N_d.

Since $N_d \cap \mathrm{sig}(\mathcal{O}) = \emptyset$, it follows from these two points that $\mathcal{O} \equiv^{\mathcal{M}}_{\mathrm{sig}(\mathcal{O})} \mathcal{V}$. From Lemma 3.5.12 we can restrict the inseparability signature and get that $\mathcal{O} \equiv^{\mathcal{M}}_{\mathrm{sig}(\mathcal{O}) \cap F} \mathcal{V}$. □

The use of a normal form may be surprising when the aim is to preserve the syntactic structure of the input ontology. Structural transformation changes the structure of the input ontology because it introduces definers, and extracts concepts occurring below role restriction. However, the idea is to eliminate the forgetting symbols from the extracted concepts, then eliminate the definers by reversing the structural transformation operations. In this way, the structural transformation itself is not damaging because we will attempt to reverse it by the end of the forgetting process.

Converting equivalence axioms to subsumption axioms also changes the structure of the input ontology because it changes the relation symbols of the axioms. That is, the equivalence symbols. However, since defined concepts have been eliminated, the equivalence axioms that we transform would use the forgetting symbol on one side of the equivalence within nested concepts. An example of an equivalence axiom that we transform is $A \sqcap B \equiv C$ where B is the forgetting symbol. Here, the concept B is nested in the concept $A \sqcap B$. Suppose $\mathcal{O}$ is the ontology containing only the axiom $A \sqcap B \equiv C$. The result of eliminating B from $\mathcal{O}$ is $C \sqsubseteq A$. which cannot be expressed with an equivalence axiom having A and C on opposite sides of the equivalence.

There are three properties of $\mathcal{O}^n$ that we need to highlight. The three properties are central for the elimination process described in the remainder of this chapter.

1. Forgetting symbols as well as de ners occur only in subsumption axioms.

2. Forgetting symbols do not occur below role restrictions.

3. If a de ner occurs at position i_1 in an axiom a_1 not below role restriction, and at position i_2 in an axiom a_2 below role restriction, then $pol(a_1; i_1)$ and $pol(a_2; i_2)$ are opposites. That is, $pol(a_1; i_1) = \neg pol(a_2; i_2)$

The rst property is true because the equivalence axioms where a forgetting symbol occurs are converted to subsumption axioms. Also, because the axioms introduced by structural transformation are subsumption axioms. The second property is a direct consequence of structural transformation. The third property can be seen from Definition 6.4.2, because a de ner D is introduced negatively in an axiom $D \vee E$ if it occurs positively below role restriction, and positively in an axiom $E \vee D$ if it occurs negatively below role restriction.

The three properties above will be preserved by the calculus rules and the method explained in the next section.

6.4.1 Eliminating Forgetting Symbols

We eliminate B from $\mathcal{O}^n$. This process proceeds in three steps. The rst step groups the axioms of $\mathcal{O}^n$ that use the symbol B and translates them to a rst-order formula ϕ_B. This step is required because the second step eliminates B from all axioms at once, and thus needs a single formula representation of the axioms that contain B. Moreover, the calculus run in the second step temporarily produces disjunctions and conjunctions of what were originally axioms in $\mathcal{O}^n$ and this is not supported in DL syntax. The third step restores the formula obtained in the second step to $\mathcal{ALC}$ description logic syntax.

We use a standard translation of the $\mathcal{ALC}$ description logic to the rst-order logic (FOL). A description of the translation has been given in Section 2.6. To simplify the notation, we use the syntax of a subsumption axiom to mean the formula obtained by the rst-order translation of the axiom. We also allow the conjunction and disjunction of axioms to mean respectively the conjunction and disjunction of their rst-order translations. Recall the translation function π from Section 2.6 which translates a DL

Splitting

$$\frac{f_B}{f_B^{\geq} \_ f_B^{?}}$$

where f_B is a first-order formula representing the set of axioms of the ontology that contain the symbol B.

Tautology Elimination (TE)

$$\frac{p(? \sqsubseteq C)}{\top}; \quad \frac{p(C \sqsubseteq \top)}{\top}; \quad \frac{p(A \sqcap C \sqsubseteq A \sqcup D)}{\top}$$

where A, C, and D are formulas.

Distribution (Dist)

$$\frac{(A_1^1 \wedge \ldots \wedge A_n^1) \_ (A_1^2 \wedge \ldots \wedge A_m^2)}{(A_1^1 \_ A_1^2) \wedge \ldots \wedge (A_n^1 \_ A_m^2)}$$

where $A_1^1, \ldots, A_n^1, A_1^2, \ldots, A_m^2$ are formulas.

Disjunction Elimination (DElim)

$$\frac{(C_1 \sqsubseteq D_1) \_ (C_2 \sqsubseteq D_2)}{(C_1 \sqcap C_2) \sqsubseteq D_1 \sqcup D_2}$$

where C_1, D_1, C_2, and D_2 are concepts.

Figure 6.2: Forgetting Calculus

axiom to a corresponding FOL formula. The following statements are synonyms:

$$C \sqsubseteq D = p(C \sqsubseteq D) \quad (6.36)$$

$$(C_1 \sqsubseteq D_1) \wedge (C_2 \sqsubseteq D_2) = p(C_1 \sqsubseteq D_1) \wedge p(C_2 \sqsubseteq D_2) \quad (6.37)$$

$$(C_1 \sqsubseteq D_1) \_ (C_2 \sqsubseteq D_2) = p(C_1 \sqsubseteq D_1) \_ p(C_2 \sqsubseteq D_2) \quad (6.38)$$

Having f_B computed, the second step eliminates B from f_B using the splitting rule in Figure 6.2. It takes one premise, the formula f_B computed above. The rule eliminates the forgetting symbol B by taking the disjunction of $f_B^{\geq}$ and $f_B^{?}$. The notation $f_B^{\geq}$ is a shorthand for $f_B[B \Rightarrow \top]$, denoting the result of substituting $\top$ for B in f_B. In the same way, $f_B^{?} = f_B[B \Rightarrow ?]$ denotes the result of substituting $?$ for B in f_B. When the formula $f_B^{\geq} \_ f_B^{?}$ is computed, we remove the axioms that use the symbol B from $\mathcal{O}^n$.

The third step in the elimination process restores the formula $f_B^{\geq} \_ f_B^{?}$ obtained from

the second step to the description logic $\mathcal{ALC}$ syntax. We use the Tautology Elimination rule in Figure 6.2, as well as the Distribution and the Disjunction Elimination rules. The Tautology Elimination rule is performed first. It eliminates tautologous conjuncts from $f^{\geq}_B$ and $f^{?}_B$.

The Distribution and the Disjunction Elimination rules eliminate the disjunction between $f^{\geq}_B$ and $f^{?}_B$. The Distribution rule distributes the disjunction over conjunction in $f^{\geq}_B \sqcup f^{?}_B$. The Disjunction Elimination rule then converts a disjunction of two formulas into a single formula that can be expressed as a DL axiom.

The following example illustrates the elimination process.

Example 6.4.5. Consider the following ontology $\mathcal{O}$.

$$A \sqsubseteq \forall r.(B \sqcap C) \tag{6.39}$$

$$\forall r.B \sqsubseteq E \tag{6.40}$$

Let $\mathbf{F} = \{B\}$ be the forgetting signature.

Definition substitution cannot be performed because $\mathcal{O}$ consists of subsumption axioms only. So, the ontology $\mathcal{O}^{\nu}$ is $\mathcal{O}$. Also, the forgetting symbol B cannot be purified because it occurs positively in (6.39), and negatively in (6.40).

We convert $\mathcal{O}^{\nu}$ to normal form. This only requires structural transformation to be performed. The normal form $\mathcal{O}^{n}$ consists of the following axioms.

$$A \sqsubseteq \forall r.D_1 \tag{6.41}$$

$$D_1 \sqsubseteq B \sqcap C \tag{6.42}$$

$$\forall r.D_2 \sqsubseteq E \tag{6.43}$$

$$B \sqsubseteq D_2 \tag{6.44}$$

The formula f_B is $(D_1 \sqsubseteq B \sqcap C) \wedge (B \sqsubseteq D_2)$, where the abbreviation in (6.37) is used to express f_B. A Splitting inference is then performed to eliminate B from f_B. When this has been performed the axioms (6.42) and (6.44) are removed.

$$f^{\geq}_B = (D_1 \sqsubseteq C) \wedge (\top \sqsubseteq D_2) \tag{6.45}$$

$$f^{?}_B = (D_1 \sqsubseteq ?) \wedge (? \sqsubseteq D_2) \tag{6.46}$$

$$f^{\geq}_B \sqcup f^{?}_B = ((D_1 \sqsubseteq C) \wedge (\top \sqsubseteq D_2)) \sqcup ((D_1 \sqsubseteq ?) \wedge (? \sqsubseteq D_2)) \tag{6.47}$$

We use Tautology Elimination to simplify (6.47), which substitutes $\top$ for the second

conjunct $\exists r.D_2$ of the second disjunct. The formula (6.47) is simplified to the following.

$$((D_1 \sqsubseteq C) \wedge (\top \sqsubseteq \exists r.D_2)) \vee (D_1 \sqsubseteq \exists r.\top) \tag{6.48}$$

Next, we push the disjunction inwards using the Distribution rule. The following formula is obtained.

$$((D_1 \sqsubseteq C) \vee (D_1 \sqsubseteq \exists r.\top)) \wedge ((\top \sqsubseteq \exists r.D_2) \vee (D_1 \sqsubseteq \exists r.\top)) \tag{6.49}$$

We perform two Disjunction Elimination inferences to eliminate the disjunctions in (6.49). The following formula is obtained.

$$(D_1 \sqsubseteq C) \wedge (D_1 \sqsubseteq D_2) \tag{6.50}$$

Finally, (6.50) is written as the set of DL axioms $\{D_1 \sqsubseteq C, D_1 \sqsubseteq D_2\}$.

The ontology $\mathcal{O}^D$ is the ontology consisting of the two axioms obtained from (6.41), (6.43), and (6.50). We list them below.

$$A \sqsubseteq \exists r.D_1$$
$$\exists r.D_2 \sqsubseteq E$$
$$D_1 \sqsubseteq C$$
$$D_1 \sqsubseteq D_2$$

When there are several forgetting symbols, the elimination process is performed iteratively. We show this in the next example.

Example 6.4.6. Consider the following ontology $\mathcal{O}$.

$$A_1 \sqsubseteq E \sqcap \exists r.(B \sqcap C)$$
$$A_2 \sqsubseteq \exists r.\neg B$$
$$\exists r.B \sqsubseteq F$$
$$E \sqsubseteq \neg B$$
$$G \sqsubseteq F$$

Let $\mathbf{F} = \{B, E, G\}$ be a forgetting signature.

definition substitution is not applicable because $\mathcal{O}$ does not have equivalence axioms. So, $\mathcal{O}^v$ is equal to the input ontology $\mathcal{O}$. The symbol G occurs only negatively in

$\mathcal{O}$. We purify G by replacing it with $\bot$, which substitutes the axiom $\bot \sqcup F$ for $G \sqcup F$. Because it is tautologous, the axiom $\bot \sqcup F$ is removed. $\mathcal{O}^{\vee}$ now consists of the first four axioms.

Suppose we eliminate B first. Since all axioms of $\mathcal{O}^{\vee}$ use the symbol B, we transform $\mathcal{O}^{\vee}$ into normal form. The following ontology is obtained.

$$A_1 \sqcup \exists t.\top \sqcup \exists r.D_1 \tag{6.51}$$
$$A_2 \sqcup \exists r.D_2 \tag{6.52}$$
$$\exists r.D_3 \sqcup F \tag{6.53}$$
$$E \sqcup \neg B \tag{6.54}$$
$$D_1 \sqcup B \sqcup C \tag{6.55}$$
$$D_2 \sqcup \neg B \tag{6.56}$$
$$B \sqcup D_3 \tag{6.57}$$

The formula f_B is the first-order translation of the formula:

$$(E \sqcup \neg B) \sqcap (D_1 \sqcup B \sqcup C) \sqcap (D_2 \sqcup \neg B) \sqcap (B \sqcup D_3) \tag{6.58}$$

We eliminate B from f_B using a Splitting inference. We have:

$$f_B^{\top} = (E \sqcup \bot) \sqcap (D_1 \sqcup C) \sqcap (D_2 \sqcup \bot) \sqcap (\top \sqcup D_3) \tag{6.59}$$
$$f_B^{\bot} = (E \sqcup \top) \sqcap (D_1 \sqcup \bot) \sqcap (D_2 \sqcup \top) \sqcap (\bot \sqcup D_3) \tag{6.60}$$

The conclusion of the Splitting inference is the formula $f_B^{\top} \sqcup f_B^{\bot}$. As splitting has been performed, we remove the axioms (6.54), (6.55), (6.56), and (6.57) where the symbol B is used.

Tautology Elimination is then performed to simplify $f_B^{\top} \sqcup f_B^{\bot}$. The inference eliminates the first, third, and fourth conjuncts of $f_B^{\bot}$. The following formula is obtained.

$$((E \sqcup \bot) \sqcap (D_1 \sqcup C) \sqcap (D_2 \sqcup \bot) \sqcap (\top \sqcup D_3)) \sqcup (D_1 \sqcup \bot) \tag{6.61}$$

Applying the Distribution rule, we get:

$$((E \sqcup \bot) \sqcup (D_1 \sqcup \bot)) \sqcap ((D_1 \sqcup C) \sqcup (D_1 \sqcup \bot)) \sqcap ((D_2 \sqcup \bot) \sqcup (D_1 \sqcup \bot)) \sqcap ((\top \sqcup D_3) \sqcup (D_1 \sqcup \bot)) \tag{6.62}$$

Disjunction Elimination inferences eliminate the four disjunctions in (6.62), and obtain the following formula.

$$(E \sqcap D_1 \sqsubseteq \bot) \wedge (D_1 \sqsubseteq C) \wedge (D_1 \sqcap D_2 \sqsubseteq \bot) \wedge (D_1 \sqsubseteq D_3) \tag{6.63}$$

The formula (6.63) is finally restored to the following form of ALC axioms.

$$E \sqcap D_1 \sqsubseteq \bot \tag{6.64}$$
$$D_1 \sqsubseteq C \tag{6.65}$$
$$D_1 \sqcap D_2 \sqsubseteq \bot \tag{6.66}$$
$$D_1 \sqsubseteq D_3 \tag{6.67}$$

In the second iteration, we eliminate the forgetting symbol E from the axioms (6.51) and (6.64). The two axioms are already in normal form. We have $f_E^{\geq} = (A_1 \sqsubseteq E \sqcup \exists r.D_1) \wedge (E \sqcap D_1 \sqsubseteq \bot)$. A Splitting inference is performed to eliminate E from $f_E^{\geq}$ as follows.

$$f_E^{\geq} = (A_1 \sqsubseteq \top) \wedge (D_1 \sqsubseteq \bot) \tag{6.68}$$
$$f_E^{?} = (A_1 \sqsubseteq \exists r.D_1) \wedge (\bot \sqsubseteq \bot) \tag{6.69}$$
$$f_E^{\geq} \vee f_E^{?} = ((A_1 \sqsubseteq \top) \wedge (D_1 \sqsubseteq \bot)) \vee ((A_1 \sqsubseteq \exists r.D_1) \wedge (\bot \sqsubseteq \bot)) \tag{6.70}$$

The formula (6.70) is simplified by Tautology Elimination, and the formula (6.71) below is obtained.

$$(D_1 \sqsubseteq \bot) \vee (A_1 \sqsubseteq \exists r.D_1) \tag{6.71}$$

Finally, we perform Disjunction Elimination to obtain the following axiom in description logic syntax.

$$A_1 \sqcap D_1 \sqsubseteq \exists r.D_1 \tag{6.72}$$

Since no more forgetting symbols remain, we obtain as the ontology consisting of the axioms (6.52), (6.53), (6.65), (6.66), (6.67), and (6.72). We summarize it below.

$$A_2 \sqsubseteq \exists r.D_2$$
$$\exists r.D_3 \sqsubseteq \bot$$
$$A_1 \sqcap D_1 \sqsubseteq \exists r.D_1$$

$$D_1 \sqsubseteq C$$
$$D_1 \sqcap D_2 \sqsubseteq \bot$$
$$D_1 \sqsubseteq D_3$$

In the remainder of this section, we prove the correctness of the rules in Figure 6.2.

De nition 6.4.7. Suppose B is the forgetting symbol. Consider the forgetting rules in Fig. 6.2. We say that a rule is *correct* if and only if for every model $\mathcal{I}$ of the premise there is a model $\mathcal{J}$ of the conclusion, and vice versa, such that $\mathcal{I}$ and $\mathcal{J}$ coincide on all symbols except, possibly, on B.

Theorem 6.4.8. The rules in Fig. 6.2 are correct.

Proof. We consider the Distribution rule rst. Let $\mathcal{I}$ be a model of the premise of the Distribution rule, and $\mathcal{J}$ a model of the conclusion. The conclusion of the Distribution rule has the disjunction in the premise distributed on the conjunctions. So, it preserves logical equivalence, and therefore $\mathcal{I}$ and $\mathcal{J}$ coincide on the interpretations of all symbols.

Second, we consider the Disjunction Elimination rule. In the FOL translation, the premise of the rule is $p(C_1 \sqsubseteq D_1) \vee p(C_2 \sqsubseteq D_2)$. Let $\mathcal{I}$ be a model of the premise of the rule, and $\mathcal{J}$ a model of the conclusion. We have the following.

$$\mathcal{I} \models p(C_1 \sqsubseteq D_1) \vee p(C_2 \sqsubseteq D_2) \tag{6.73}$$
$$(p_x(C_1) \rightarrow p_x(D_1)) \vee (p_x(C_2) \rightarrow p_x(D_2)) \tag{6.74}$$
$$(p_x(\neg C_1) \vee p_x(D_1)) \vee (p_x(\neg C_2) \vee p_x(D_2)) \tag{6.75}$$
$$(p_x(\neg C_1) \vee p_x(\neg C_2)) \vee (p_x(D_1) \vee p_x(D_2)) \tag{6.76}$$
$$(p_x(\neg (C_1 \sqcap C_2))) \vee (p_x(D_1) \vee p_x(D_2)) \tag{6.77}$$
$$p_x(C_1 \sqcap C_2) \rightarrow (p_x(D_1) \vee p_x(D_2)) \tag{6.78}$$
$$p_x(C_1 \sqcap C_2) \rightarrow p_x(D_1 \sqcup D_2) \tag{6.79}$$
$$p((C_1 \sqcap C_2) \sqsubseteq (D_1 \sqcup D_2)) \tag{6.80}$$

From the above equivalences we have that $\mathcal{I}$ and $\mathcal{J}$ coincide on all symbols. Therefore, the Disjunction Elimination rule is correct.

Third, we consider the Splitting rule. Suppose the forgetting symbol B is The premise of f_B is the rst-order formula on the form $f_B = \forall x\, p(A_1) \wedge \forall x\, p(A_2) \wedge \cdots \wedge \forall x\, p(A_n)$ where $A_1, \ldots, A_n$ are axioms of the ontology that contain the symbol B. We

rewrite f_B as $\forall x.F_1(x) \wedge \forall x.F_2(x) \wedge \ldots \wedge \forall x.F_n(x)$ where each $F_i(x)$ is the formula $\rho(A_i)$ with one free variable x. We show that the Splitting rule is correct by showing the following.

1. Every model $\mathcal{I}$ of the premise is a model of the conclusion.
2. Every model $\mathcal{I}$ of the conclusion can be extended to a model $\mathcal{J}$ of the premise, such that $\mathcal{I}$ and $\mathcal{J}$ coincide on the interpretations of all symbols, except possibly on B.

First, we prove 1. Let $\mathcal{I}$ be an arbitrary model of f_B, we have:

$$\mathcal{I} \models \forall x.F_1(x) \wedge \forall x.F_2(x) \wedge \ldots \wedge \forall x.F_n(x) \tag{6.81}$$

$$\forall x.(F_1(x) \wedge \ldots \wedge F_n(x)) \tag{6.82}$$

$$\forall x((B(x) \wedge F_1(x) \wedge \ldots \wedge F_n(x))[B \Rightarrow \top] \sqcup (\neg B(x) \wedge F_1(x) \wedge \ldots \wedge F_n(x))[B \Rightarrow \bot]) \tag{6.83}$$

$$\forall x(f_B^{\geq} \sqcup f_B^{\bot}) \tag{6.84}$$

Therefore, $\mathcal{I}$ is also a model of the conclusion.

Second, we prove 2. Let $\mathcal{I}$ be an arbitrary model of the conclusion of the Splitting rule. We move to a new model $\mathcal{J}$ which coincides with $\mathcal{I}$ on everything and, additionally, interprets the predicate B as follows:

$$\mathcal{J} \models B(a) \text{ iff } \mathcal{I} \models f_B^{\geq}[x/a]$$

for every element a in $\Delta^{\mathcal{I}}$. It follows directly that $\mathcal{J}$ is a model of f_B. Also, by construction of $\mathcal{J}$, we have $\mathcal{I} \equiv_B \mathcal{J}$. Therefore, the Splitting rule is correct.

Finally, Tautology Elimination is a standard operation that preserves model inseparability with respect to $sig(\mathcal{O}^v) \setminus \{B\}$. Therefore, as shown for the Distribution and Disjunction Elimination rules, the Tautology Elimination rule is correct. □

6.5 Stage 3: Definer Elimination

We explain the third stage of the method. The input to this stage is the ontology $\mathcal{O}^D$ obtained from the previous stage. The aim in this stage is eliminating the definers from $\mathcal{O}^D$, thereby generating a representation of the semantic forgetting views of the input ontology with respect to the forgetting signature. Complete elimination of

de ners is however not guaranteed, because it is not always possible to represent the semantic forgetting view in $\mathcal{ALC}$ over the non-forgotten symbols. More details on this can be found in Chapter 4 of this thesis. Our aim, therefore, is to identify the de ners that can be eliminated, and explain their elimination process. As done in Section 4.5, we eliminate de ners by reversing the structural transformation process. However, the elimination process presented here is more complex relative to the process in Section 4.5, because the normal form $\mathcal{O}^{\mathcal{P}}$ is more liberal than the normal form used in Chapter 4.

The following de nitions allow for the characterization of the de ners that can be eliminated, and the de ners that cannot be eliminated.

De nition 6.5.1. Let D be a de ner. We denote by $\mathit{def}(D)$ the set of all axioms that can be put in the forms $D \sqsubseteq C$ or $C \sqsubseteq D$ where C is an $\mathcal{ALC}$ concept.

De nition 6.5.2. Let D be a de ner and consider any axiom $\alpha \in \mathit{def}(D)$. Then, α is improper for D if one of the following is true:

1. α can be put equivalently into the forms $C \sqcap \bar{D} \sqsubseteq C$ or $C \sqsubseteq D \sqcup \bar{D}$ for any $\bar{D} \in \mathit{Nd}$ where $\bar{D} \neq D$ and C is an $\mathcal{ALC}$ concept.

2. α can be put equivalently into the forms $\forall r.D \sqsubseteq C$ or $C \sqsubseteq D$ and $Qr.D$ occurs in C where $Q \in \{\exists, \forall\}$, and r is any role name.

Otherwise, α is proper for D.

The set $\mathit{def}(D)$ is proper for D if all axioms $\alpha \in \mathit{def}(D)$ are proper for D, otherwise, $\mathit{def}(D)$ is improper for D.

We explain De nitions 6.5.1 and 6.5.2 by the following example.

Example 6.5.3. Recall the axioms (6.65), (6.66), (6.67), and (6.72) from Example 6.4.6, repeated here:

$$D_1 \sqsubseteq C \tag{6.65}$$
$$D_1 \sqcap D_2 \sqsubseteq \bot \tag{6.66}$$
$$D_1 \sqsubseteq D_3 \tag{6.67}$$
$$A \sqcap D_1 \sqsubseteq \exists r.D_1 \tag{6.72}$$

The symbols D_1, D_2, and D_3 are de ners. The set $\mathit{def}(D_1)$ consists of the four axioms above. The set $\mathit{def}(D_2)$ consists of the axiom (6.66). The set $\mathit{def}(D_3)$ consists of the axiom (6.67).

The axiom (6.65) is proper for D_1, and (6.67) is proper for D_1 and D_3. The axiom (6.66) is improper for D_1 and D_2, because it satisfies the first condition of Definition 6.5.2. The axiom (6.72) is improper for D_1 because it satisfies the second condition of Definition 6.5.2.

It may not be possible to eliminate a definer D if at least one axiom from $def(D)$ is improper for D. In the above example, the definer D_1 cannot be eliminated because the two axioms (6.66) and (6.72) from $def(D_1)$ are improper for D_1. The definer D_2 cannot be eliminated because the axiom (6.66) is improper for D_2. The definer D_3 can be eliminated because $def(D_3)$ has only one axiom (6.67) which is proper for D_3.

Suppose D is a definer, and a is an improper axiom for D because it satisfies the second condition Definition 6.5.2. Then, the definer D is a cyclic definer. To see this, recall the third property discussed at the end of Section 4.2 regarding the polarity of the definer. This property is preserved by the method explained in this section because the calculus rules preserve the positions and polarities of concepts. If a satisfies the second condition in Definition 6.5.2, then the definer D occurs both positively and negatively in one axiom. From Definition 4.5.1, we can see that D is a cyclic definer.

In Chapter 4 we saw another type of definers which cannot be eliminated. These are the definers occurring in the difference set Δ^d. If we compute the intermediate ontology $\mathcal{O}^{int}$ of the ontology $\mathcal{O}$ with respect to $\mathbf{F}$ from Example 6.4.6, we will find that every definer in $\mathcal{O}^{int}$ occurs in Δ^d. The ontology $\mathcal{O}^{int}$ would comprise the following clauses.

$$: A_2 \sqsubseteq \exists r.D_2' \tag{6.85}$$

$$\mathbf{F} \sqsubseteq \forall r.D_3' \tag{6.86}$$

$$: D_1' \sqsubseteq C \tag{6.87}$$

$$: D_1' \sqsubseteq \neg D_2' \tag{6.88}$$

$$: D_1' \sqsubseteq \neg D_3' \tag{6.89}$$

$$: A_1 \sqsubseteq \neg D_1' \sqsubseteq \exists r.D_1' \tag{6.90}$$

There are three definers D_1', D_2', and D_3' in $\mathcal{O}^{int}$. The set Δ^d contains the two clauses (6.88) and (6.89). The three definers D_1', D_2', and D_3' occur in Δ^d. Therefore, they cannot be eliminated from $\mathcal{O}^{int}$. In $\mathcal{O}^D$, however, only two definers cannot be eliminated.

A definer D for which $def(D)$ is proper can be eliminated. The elimination rules of D are presented in Fig. 6.3. The NDef and the PDef rules groups together the axioms of $def(D)$ in one axiom.

NDef

$$\frac{\mathcal{O}[\{(D \sqcap C_1 \sqsubseteq E_1), \ldots, (D \sqcap C_n \sqsubseteq E_n)\}]}{\mathcal{O}[\{D \sqsubseteq (\neg C_1 \sqcup E_1) \sqcap \ldots \sqcap (\neg C_n \sqcup E_n)\}]}$$

where D is a definer, $\mathit{def}(D)$ is proper, and $D \sqcap C_i \sqsubseteq E_i$ are axioms in $\mathit{def}(D)$ with $1 \leq i \leq n$.

PDef

$$\frac{\mathcal{O}[\{(C_1 \sqsubseteq D \sqcup E_1), \ldots, (C_n \sqsubseteq D \sqcup E_n)\}]}{\mathcal{O}[\{(\neg C_1 \sqcap \neg E_1) \sqcap \ldots \sqcap (\neg C_n \sqcap \neg E_n) \sqsubseteq D\}]}$$

where D is a definer, $\mathit{def}(D)$ is proper, and the axioms $C_i \sqsubseteq D \sqcup E_i$ are axioms in $\mathit{def}(D)$ with $1 \leq i \leq n$.

Negative Non-Cyclic Definer Elimination

$$\frac{\mathcal{O}[\{(D \sqsubseteq C)\}]}{\mathcal{O}[D/C]}$$

where D is a definer and may occur in $\mathcal{O}$ only positively

Positive Non-Cyclic Definer Elimination

$$\frac{\mathcal{O}[\{(C \sqsubseteq D)\}]}{\mathcal{O}[D/C]}$$

where D is a definer and may occur in $\mathcal{O}$ only negatively.

Figure 6.3: Definer elimination rules

The Negative Non-Cyclic Definer Elimination and the Positive Non-Cyclic Definer Elimination rules eliminate D using the Ackermann approach [Ack35, Sch12]. The premise $C \sqsubseteq D$ of a Positive Non-Cyclic Definer Elimination inference is usually obtained from a PDef inference, and the premise $D \sqsubseteq C$ of a Negative Non-Cyclic Definer Elimination inference is often obtained from an NDef inference. The Negative Non-Cyclic Definer Elimination and the Positive Non-Cyclic Definer Elimination correspond to the Non-cyclic Definer Elimination rules used in Section 4.5.

Lemma 6.5.4. Let $\mathcal{O}_1$ be an ontology and $\mathcal{O}_2$ the ontology after eliminating a definer D using the rules in Fig. 6.3. Then, for every model $\mathcal{I}$ of $\mathcal{O}_1$ there is a model $\mathcal{J}$ of $\mathcal{O}_2$, and vice versa, such that $\mathcal{I}$ and $\mathcal{J}$ coincide on all symbols except D.

Proof. The result of the NDef (PDef) rule is a syntactic variant of the premises.

Since the Negative Non-Cyclic De ner Elimination and the Positive Non-Cyclic De-ner Elimination rules achieve the reverse of structural transformation, it follows by the the proof of Lemma 6.4.4 that the Negative Non-Cyclic De ner Elimination and the Positive Non-Cyclic De ner Elimination rules preserve the interpretations of all symbols up to the eliminated de ners. □

Theorem 6.5.5. Let $\mathcal{O}$ be an ontology, $\mathbf{F}$ a forgetting signature, and $\mathcal{V}$ be the semantic forgetting view computed as described above. For every model $\mathcal{I}$ of $\mathcal{O}$ there is a model $\mathcal{J}$ of $\mathcal{V}$, and vice versa, such that $\mathcal{I}$ and $\mathcal{J}$ coincide on all symbols except the symbols from $\mathbf{F} \cup \mathbf{N}_d$, where $\mathbf{N}_d$ is the set of de ners in $\mathcal{V}$.

Proof. Let $\mathcal{O}_1$ be the ontology $\mathcal{O}$ after structural transformation, $\mathcal{O}_2$ the result of applying the calculus in Fig. 6.2 exhaustively on $\mathcal{O}_1$, and $\mathcal{V}$ the result of eliminating de ners from $\mathcal{O}_2$ using the rules in Fig. 6.3. By Lemma 6.4.4, the interpretations of all symbols of $\mathcal{O}$ are preserved in the models of $\mathcal{O}_1$. By Theorem 6.4.8, the models of $\mathcal{O}_1$ and $\mathcal{O}_2$ may only differ on the interpretation of the symbols $\mathbf{F}$. Finally, by Lemma 6.5.4, the models of $\mathcal{O}_2$ and $\mathcal{V}$ may only differ on the interpretation of the de ners from $\mathbf{N}_d$. It follows from the above that for every model $\mathcal{I}$ of $\mathcal{O}$ there is a model $\mathcal{J}$ of $\mathcal{V}$, and vice versa, such that $\mathcal{I}$ and $\mathcal{J}$ coincide on all symbols except the symbols from $\mathbf{F} \cup \mathbf{N}_d$. □

6.6 Preserving Ontology Structure

In Chapter 4 we computed the ontology $\mathcal{O}^{int}$ which can be viewed as a representation of the semantic forgetting views of the input ontology with respect to the forgetting signature. We have seen in many examples that the output of the method presented in this chapter is more resembling to the structure of the input ontology than $\mathcal{O}^{int}$. The question of why the method in Chapter 4 does not preserve the syntactic structure of the input ontology is an important question that we answer in this section. We present a set of examples that highlight the differences between the intermediate ontology $\mathcal{O}^{int}$ and the ontology $\mathcal{V}^s$ obtained by the forgetting method presented in this chapter. We identify the root cause of these differences in the underlying forgetting methods. The root causes observed in this section are also true for the forgetting methods in [GO92, LK14, KS13c, KS14, KS15a, ZS15, ZS16, ZS18, ZS17].

Suppose we have the following axioms.

$$A \sqsubseteq B \sqcup C \sqcup D \tag{6.91}$$
$$A \sqsubseteq B \tag{6.92}$$
$$B \sqsubseteq C \sqcup D \tag{6.93}$$
$$\forall r.B \sqsubseteq C \tag{6.94}$$
$$A \sqsubseteq B \tag{6.95}$$

Let $\mathcal{O}_1, \mathcal{O}_2$, and $\mathcal{O}_3$ be three ontologies, where $\mathcal{O}_1$ consists of the axiom (6.91), $\mathcal{O}_2$ consists of the two axioms (6.92) and (6.93), and $\mathcal{O}_3$ consists of the axioms (6.94) and (6.95).

Table 6.1 compares the semantic forgetting views $\mathcal{O}^{int}$ and $\mathcal{V}^s$.

Ontology	Axioms	$\mathbf{F}$	$\mathcal{O}^{int}$	$\mathcal{V}^s$
$\mathcal{O}_1$	$A \sqsubseteq B \sqcup C \sqcup D$	$\{B\}$	$\neg At \sqsubseteq C$ $\neg At \sqsubseteq D$	$A \sqsubseteq C \sqcup D$
$\mathcal{O}_2$	$A \sqsubseteq B$ $B \sqsubseteq C \sqcup D$	$\{B\}$	$\neg At \sqsubseteq C$ $\neg At \sqsubseteq D$ $\neg C \sqcap \neg D \sqsubseteq A$	$A \sqsubseteq C \sqcup D$
$\mathcal{O}_3$	$\forall r.B \sqsubseteq C$ $A \sqsubseteq B$	$\{B\}$	$C \sqsubseteq \exists r.\neg A$	$\forall r.A \sqsubseteq C$

Table 6.1: Comparison of $\mathcal{O}^{int}$ and $\mathcal{V}^s$ obtained for the three ontologies $\mathcal{O}_1, \mathcal{O}_2$, and $\mathcal{O}_3$ with respect to $\mathbf{F}$.

The structure resemblance between $\mathcal{V}^s$ and the input ontology is clearly visible in Table 6.1 in each of the three examples. The intermediate ontologies in the three examples do not preserve the syntactic structure of the input ontologies because they use a destructive normal form. That is, the normal form used in Chapter 4 does not preserve the syntactic structure of the input ontology. This observation can also be made for the normal forms of the methods in [GO92, LK14, KS13c, KS14, KS15a, ZS15, ZS16, ZS18, ZS17].

A common property of the normal forms of these methods as well as the two methods in Chapters 4 and 5 is the transformations to negation normal form and conjunctive normal form. For example, the transformation to negation normal form is a direct cause for using the existential role restriction and the concept $\neg At$ in the intermediate ontology of $\mathcal{O}_3$ with respect to $\mathbf{F}$. Moreover, it also causes the use of the concepts $\neg C \sqcap \neg D$

and: A in the intermediate ontologies of O_1 and O_2 with respect to F.

The transformation to conjunctive normal form leads to using two clauses in the intermediate ontology of O_1 with respect to F, and three clauses in the intermediate ontology of O_2 with respect to F.

The normal form used in this chapter avoided the transformation to negation normal form and conjunctive normal form. This way, we managed to obtain an output that is more resembling of the syntactic structure of the input ontology, and more natural for users.

Chapter 7

Implementation and Evaluation

We presented a ne-grained forgetting framework in Chapter 4. The framework computes a representation of the semantic forgetting views of the input ontology with respect to the forgetting signature. The framework allows for customizing the content of the obtained representation and obtaining a ne-grained for getting views that are more informative than the deductive forgetting views and less informative than the semantic forgetting views. A customization whereby the nal forgetting view coincides in content with the deductive forgetting views was presented. In Chapter 5 we showed another customization whereby the nal forgetting view coincides with the query forgetting views of the ontology with respect to the forgetting signature.

In Chapter 6 we designed a new method that computes a different representation of the semantic forgetting views of the input ontology with respect to the forgetting signature. The aim was investigating and solving the challenges related to preserving the syntactic structure of the input ontology in the computed forgetting view.

The key motivation of our work is satisfying important requirements of the users. These being the enhancement of the content of the forgetting views, and the improvement of the syntactic representation of the forgetting views. Besides these two requirements, we are also interested in the practicality of the forgetting method. Practicality is an important requirement because the size of the forgetting view is in the worst case triple exponential in the size of the input ontology [LW11]. Thus, it is important to evaluate the forgetting methods on real-life ontologies.

In this chapter, we present results of an evaluation of the ne-grained forgetting framework, the deductive customization, and the semantic forgetting method. We compare our results to the state-of-the-art deductive forgetting tool LETHE [Koo15] and semantic forgetting tool FAME [ZS15].

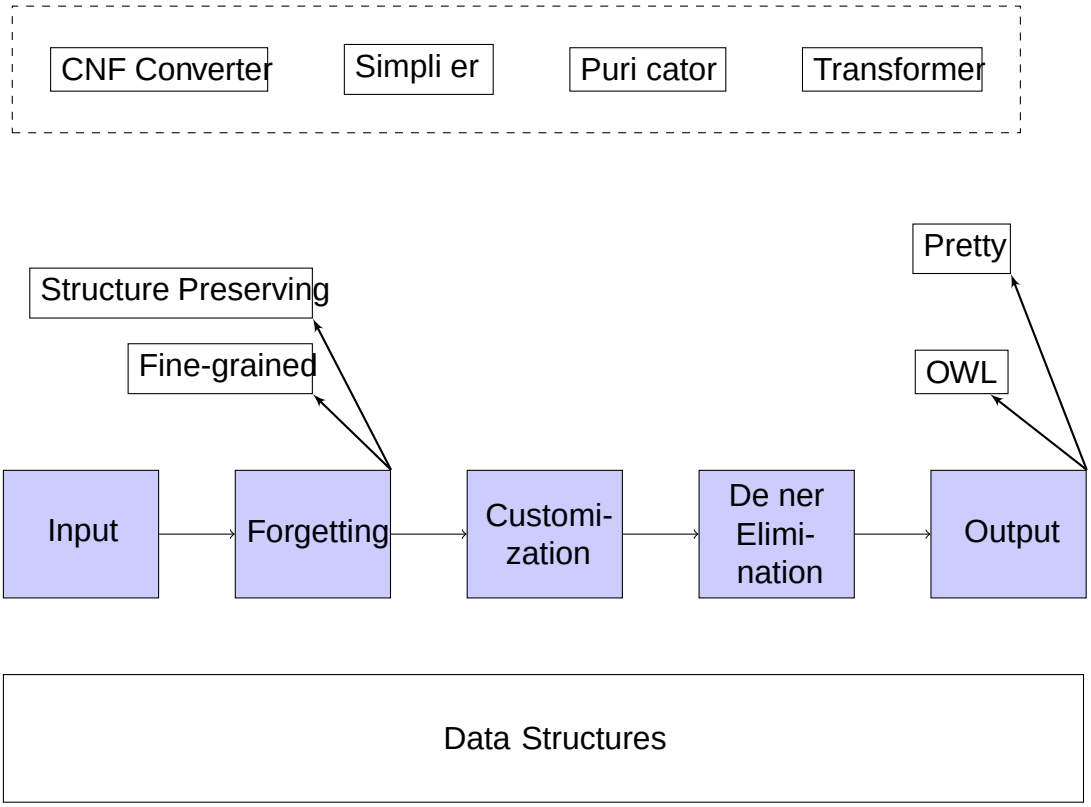

Figure 7.1: Architecture of the F3 system.

The chapter is structured as follows. Section 7.1 describes our implementations of the ne-grained forgetting framework and the semantic forgetting method. Section 7.2 describes the corpus of the ontologies used in the evaluation experiments. An evaluation of the performance of our ne-grained framework and the deductive customization is presented in Section 7.3. We also compare the results against those of LETHE. Section 7.4 presents a performance evaluation of the semantic forgetting method from Chapter 6 of this thesis, which is compared against the performance of FAME.

7.1 The Implementation

We implemented a single con gurable prototype of the ne-grained forgetting framework and the semantic forgetting method. The prototype is based on Java 12, a cross-platform object oriented programming language. Thus, the prototype can be executed on different operating systems and hardware setups. Two avors of the prototype are supported. The rst is a stand-alone application, and the second is a pluggable java library.

Figure 7.1 depicts the architecture of the implementation. We use a pipeline of five modules. In the figure, they are highlighted in blue. The first module is the Input module. It reads and parses the input ontology. The module relies on the OWL API 5.1.11 [Pal19]. Thus, ontology formats such as OWL [HPSvH03, Con12] and RDF [CHPS08] are supported.

The second module is the Forgetting module. This module is responsible for eliminating the forgetting symbols from the input ontology. The output of this module is an ontology that is semantically inseparable from the input ontology with respect to the non-forgotten symbols. We provided two implementations of this module.

1. The Fine-grained implementation.
2. The Structure Preserving implementation.

The Fine-grained implementation converts the ontology to the normal form described in Section 4.2, and performs the forgetting method described in Section 4.3. The output of this implementation is the intermediate ontology $\mathcal{O}^{int}$. The Structure Preserving implementation performs the forgetting method described in Sections 6.3 and 6.4. The output of this implementation is the ontology $\mathcal{O}^{D}$.

The third module is the Customization module. When this module is used, the Fine-grained implementation of the Forgetting module must also be used. Taking the ontology $\mathcal{O}^{int}$ from the Fine-grained implementation, the module customizes its content and computes the two ontologies $\mathcal{O}^{d}$ and $\mathcal{D}^{d}$. Recall that $\mathcal{O}^{d}$ is concept inseparable from $\mathcal{O}^{int}$ with respect to the non-forgotten symbols, and $\mathcal{D}^{d}$ is a set of axioms that indicates the content difference between $\mathcal{O}^{int}$ and $\mathcal{O}^{d}$. A configuration of the prototype that just aims to obtain a representation of the semantic forgetting views, turns this module off.

The fourth module is the Definer Elimination module. It eliminates the definers from the input ontology of the module. The output of the module is the final forgetting view. The module implements the definer elimination process described in Section 6.5. We observe that when input ontology of this module is in the normal form presented in Section 4.2, the outputs of the definer elimination methods in Sections 4.5 and 6.5 coincide. Thus, having the definer elimination process from Section 6.5 is sufficient to support all configurations of the prototype. We will explain the different configurations of the prototype shortly.

The fifth module is the Output module. This module is responsible for store the final forgetting view computed by the previous module to file format. There are two

implementations of this module.

1. The OWL implementation.

2. The Pretty implementation.

The OWL implementation writes the forgetting view in OWL format. The Pretty implementation writes the output in a user friendly format. When the stand-alone version of the prototype is used, the Pretty implementation is more suitable to express the forgetting view. When the output is meant to be shared with other users or applications for further processing, the OWL format is more suitable.

The implementation uses a library of the following supporting components.

1. CNF Converter: The input of this component is a set of clauses in negation normal form. The output is a set comprising the translation of the input clauses to the conjunctive normal form.

2. Simplifier: The input of this component is a set of axioms. The output is the set consisting of the input axioms where tautologous concepts and axioms have been eliminated.

3. Purificator: The component takes two inputs, a set of axioms, and a set of concept names. It iterates over the concepts from the second set, and purifies them in the first set if they occur with only one polarity.

4. Transformer: The component takes two inputs, a set of axioms, and a set of concept names. The component performs structural transformation exhaustively until no symbol from the second set occurs below role restriction in the first set.

Observe that the above library does not include a component to convert a set of axioms to clauses in negation normal form, instead the OWL built-in negation normal form transformer is used when such a transformation is required.

We use our own data structures to store ontologies, axioms, clauses, and concepts to allow for optimizing the memory access operations.

Our prototype is called F3, an abbreviation of Fine-grained Forgetting Framework.

There are three configurations of F3. The first configuration uses the fine-grained implementation of the Forgetting module, and turns the Customization module on. This configuration is an implementation of the deductive forgetting method obtained from the fine-grained framework by using the deductive customization. The second

Size of Ontology			#Concept Names		
Avg.	Median	Max	Avg.	Median	Max
22663	4351	133290	10152	3390	73390

Table 7.1: Average, median, and maximum number of axioms and concept names in our ontology corpus.

configuration uses the Fine-grained implementation of the Forgetting module, and turns the Customization module off. This configuration is an implementation of the semantic forgetting method obtained from the fine-grained framework. The third configuration uses the Structure Preserving implementation of the Forgetting module, and turns the Customization module off. This configuration is an implementation of the semantic forgetting method explained in Chapter 6.

7.2 The Corpus

We evaluated the performance of our prototype on a corpus of real-world ontologies from the NCBO BioPortal repository [WNS+11, MNS+11, MBP13]. The NCBO BioPortal is a large repository of biomedical ontologies. It has over 1000 ontologies covering most branches of biological and medical fields. It has been an important resource to health care professionals as well as researchers, attracting over seventy five thousand visits in September 2023 [Bio23]. It has also been used as a benchmark repository in the evaluation of the state-of-the-art forgetting tools Lethe [Koo15], and Fame [Zha18].

Fifty ontologies were randomly selected from the NCBO BioPortal. Table 7.1 summarizes key statistics of the corpus. The median of the sizes of the ontologies was 4351 axioms. The higher average of 22663 axioms indicates that the corpus also included many large-scale ontologies with a maximum of 133290 axioms.

The table also shows a diversity in the number of concept names that occurred in the ontologies, which can be used as an indicator on the complexity of the modeled information in the ontologies. The median number of concept names that occurred in the ontology was 3390, indicating a medium complexity for most ontologies. The high average of 10152 concept names in the ontologies indicated that many ontologies in the corpus used a large number of concept names up to a maximum of 73390. This means that our corpus had many ontologies that modeled a large amount of information.

One performed evaluation of the semantic forgetting method presented in Chapter 6 compared the performance of the method against the semantic forgetting method FAME [ZS15]. In this evaluation we needed to construct a set of handmade ontologies because FAME could not handle the sizes of many of the ontologies of our corpus. The ontologies are constructed such that:

1. They are constructed from real life ontologies; and

2. They model one or several related topics.

The first requirement was important to build realistic data that resembles real-life ontologies. The second requirement was important to achieve integrity in the constructed ontologies. A handmade ontology was constructed in the following way:

1. A set S of random concept names were collected from the Interlinking Ontology for Biological Concepts (IOBC) ontology [KKK+19][1]. The IOBC is a large biomedical ontology which describes over 80,000 biological concepts such as biological phenomena, diseases, molecular functions, gene products, chemicals, and medical cares. The concepts of the IOBC are structured by 35 role names, for example, 'has function', 'has role', 'has quality', and 'is participant in'.

2. The ontologies from the NCBO Bioportal that use symbols from S were retrieved.

3. The axioms from the retrieved ontologies, where symbols from S occur, were extracted and combined into a single ontology.

The construction guaranteed that each generated ontology contained information that described the concept names (topics) extracted from the IOBC ontology. Since these concepts were extracted from the same ontology, i.e., the IOBC ontology, they were semantically related. In total, 28 ontologies were constructed using the method described above. In each construction, at least 25% of the input ontologies were expressed in $\mathcal{ALC}$ or a more expressive logic. Axioms in an input ontology that used more expressive constructs than allowed in $\mathcal{ALC}$ were removed.

[1] Interlinking Ontology for Biological Concepts © National Bioscience Database Center licensed under CC Attribution-NonCommercial 4.0 International

7.3 The Fine-Grained Forgetting Framework

This section presents the evaluation of the fine-grained forgetting framework, and the customization whereby the final forgetting views coincides with the deductive forgetting views of the input ontologies with respect to the forgetting signatures. First we explain the design of the performed experiments, then we present and analyze the obtained findings.

7.3.1 The Experiments

We performed two evaluations that are different in regards to the selection method of the forgetting signature. The selection method of the first evaluation, Evaluation 1, sorted the concept names of the tested ontology by name, and used a sequence from the beginning of the sorted list as the forgetting signature Σ. For instance, suppose we use an ontology $\mathcal{O}$ in a forgetting experiment, and the following concept names are used in $\mathcal{O}$.

1. Abdomen.
2. Bone-Marrow.
3. Abdomen-Pain.
4. Cancer.
5. Bone.
6. Abdomen-Operation.

Sorting the above concepts by name reorganizes them as follows.

1. Abdomen.
2. Abdomen-Operation.
3. Abdomen-Pain.
4. Bone.
5. Bone-Marrow.
6. Cancer.

A forgetting signature of size three would have the symbols Abdomen, Abdomen-Operation, and Abdomen-Pain in it, because the three concepts occur at the top of the sorted list.

The rational behind this signature selection method was to simulate the use case of extracting the knowledge of a single topic, for example, the digestive system, from a large biomedical ontology.

The signature selection method of the second evaluation, Evaluation 2, selected the concept names that occurred most frequently under role restrictions. For instance, in an input ontology $\mathcal{O}$, suppose Bone-Marrow is used twelve times below role restrictions, Abdomen is used nine times below role restrictions, Bone is used two times below role restrictions, and the remainder of the concepts are not used below role restrictions. A forgetting signature of two symbols would consist of Bone-Marrow and Abdomen. The other concept names would not be in the forgetting signature because they occurs less frequently below role restrictions than Bone-Marrow and Abdomen.

The aim of the signature selection method of the second evaluation was to maximize the number of definers being introduced in $\mathcal{O}^{int}$. When the ontology $\mathcal{O}$ from above was transformed to the normal form, structural transformation was performed twenty-one times, twelve times to extract role fillers where Bone-Marrow is used, and nine times to extract the role fillers where Abdomen was used. In total, twenty-one definers were introduced. When the number of definers is large, more clauses are likely to occur in the set $\mathcal{D}^{d}$, and the two benefits below are expected.

1. The total time consumed in computing the clauses of $\mathcal{D}^{d}$ and definer elimination would increase, and the performance of our three stage forgetting framework would be tested in extreme conditions.

2. The increase in the number of clauses of $\mathcal{D}^{d}$ would give insight on the amount of information that is not preserved by the deductive forgetting views of real-life ontologies.

In these two evaluations of the fine-grained forgetting framework we used the real-life ontologies obtained directly from the Bioportal repository, but not the hand-made ontologies. A total of 300 experiments were performed in the two evaluations. Every ontology was used in three experimental settings in each evaluation. 10% of the concept names used in the ontology were forgotten in the first setting, 30% were forgotten in the second setting, and 50% were forgotten in the third setting. The average size

Table 7.2: Number of timeouts of F3 and Lethe

	Evaluation 1			Evaluation 2		
	10%	30%	50%	10%	30%	50%
F3	2	3	5	2	3	3
Lethe	2	7	8	1	6	7

of the forgetting signature is 1015 concept names in the 10% setting, 3046 in the 30% setting, and 5076 in the 50% settings.

Suppose an ontology $\mathcal{O}$ is used in the three experiment settings. Let $\mathbf{F}_1$ be the forgetting signature of the 10% setting, $\mathbf{F}_2$ the forgetting signature of the 30% setting, and $\mathbf{F}_3$ the forgetting signature of the 50% setting. We have the following relations.

$$\mathbf{F}_1 \quad \mathbf{F}_2 \quad \mathbf{F}_3 \tag{7.1}$$

The relations in (7.1) are true in the two evaluations.

Each experiment compared the performance of F3 with the performance of Lethe (version 2.11-0.026)[2][KS13a]. Lethe is an implementation of the deductive forgetting method in [KS13c]. Each tool was allocated 2GB of memory and the timeout was five hours. All experiments were run on a x64-based processor Intel(R) Core(TM) i5 CPU @ 2.7GHz with a 64-bit operating system (macOS Catalina 10.15.7).

7.3.2 The Analysis

Table 7.2 shows the number of timeouts of F3 and Lethe. Fewer timeouts were observed for F3 than for Lethe in both evaluations. Also, F3 was less affected by increasing the size of $\mathbf{F}$ from 10% to 30% to 50% of $\mathbf{N_c}$, indicating that F3 is more reliable and scalable to harder problems than Lethe.

We compared the execution times of F3 and Lethe to compute the deductive view. We first computed the time gained by using F3 over Lethe with the following formula.

$$(T_L \quad T_N)/T_L \tag{7.2}$$

where T_L is the time consumed by Lethe, and T_N is the time consumed by F3. Second, we computed the averages and standard deviations of the G values. In line with

[2] https://lat.inf.tu-dresden.de/~koopmann/LETHE/

Figure 7.2: Normal Distribution of the Gain values. Range of X-axes is Average 3 Standard Deviations

standard data analysis methods, outliers were excluded. These were experiments with extreme Gain values compared to the rest of the experiments. In Evaluation 1 we excluded two experiments in the 10% setting and one in the 50% setting, whereas in Evaluation 2 we excluded one experiment in the 10% setting and one from the 50% setting.

The averages and standard deviations in Evaluation 1 were (0.58, 0.78) in the 10% setting, (0.60, 0.68) in the 30% setting, and (0.66, 0.68) in the 50% setting. The averages and standard deviations in Evaluation 2 were (0.65, 0.46) in the 10% setting, (0.74, 0.50) in the 30% setting, and (0.74, 0.67) in the 50% setting. Figure 7.2 shows the normal distributions of the Gain values in the three settings in the two evaluations. The graphs re ect the attained positive averages stated above, and compare the gain values across the three settings in each evaluation, also allowing the two evaluations to be compared.

Normal distributions are known to have bell shape curves. The x-axis represents the gain values of using F3 over Lethe, the y-axis represents the probability of attaining these values in the performed experiments. The peak of the curve is at the average gain value. The stretch of the curve indicates the standard deviation in the gain values. When the curve is stretched, high probabilities of less gains would be indicated. The narrower the curve, the more representative the average is of the data. 68.2% of the experiments results are represented by the section of the curve within plus or minus one standard deviation from the average. 95.4% of the experiments results are represented by the section of the curve within plus or minus two standard deviations from the average. 99.7% of the experiments results are represented by the section of the curve within plus or minus three standard deviations from the average. In the gures, the graphs cover the section within plus or minus three standard deviations from the

average. As such, 99.7% of the experiments are represented.

Surprisingly, a better and more consistent performance was found in Evaluation 2 over Evaluation 1, against our expectation that Evaluation 2 would represent the worst case scenario for F3. This is indicated by the higher peaks indicating higher probability of achieving the average gain, and the narrower curves indicating less variation in the results.

The performance improvement happens due to the forgetting method itself not the corpus. While Lethe translates the input ontology to a clausal form that is similar to ours, it disallows clauses with two or more negative de ners. To compensate for this restriction, Lethe introduces de ner symbols as part of the forgetting calculus, and builds a subsumption hierarchy between the de ners. This hierarchy forces extra resolution inferences to be performed. The following example illustrates dynamic introduction of de ners in Lethe.

Example 7.3.1. Consider the following ontology $\mathcal{O}$

$$A_1 \sqsubseteq \exists r.\top \sqsubseteq B \tag{7.3}$$

$$A_2 \sqsubseteq \forall r.B \tag{7.4}$$

Let $F = \{B\}$ be a forgetting signature. Lethe converts the input ontology to the following normal form.

$$\neg A_1 \sqcup \exists r.D_1 \tag{7.5}$$

$$\neg A_2 \sqcup \forall r.D_2 \tag{7.6}$$

$$\neg D_1 \sqcup \neg B \tag{7.7}$$

$$\neg D_2 \sqcup B \tag{7.8}$$

where D_1 and D_2 are de ners. Instead of resolving (7.7) and (7.8) on B to compute the clause $\neg D_1 \sqcup \neg D_2$, Lethe introduces a new de ner D_3 and appends the following clauses to the ontology.

$$\neg A_1 \sqcup \neg A_2 \sqcup \exists r.D_3 \tag{7.9}$$

$$\neg D_3 \sqcup D_1 \tag{7.10}$$

$$\neg D_3 \sqcup D_2 \tag{7.11}$$

The clauses (7.7) and (7.10) are resolved on D_1 to obtain the clause $\neg D_3 \sqcup \neg B$. In the

same way, the clauses (7.8) and (7.11) are resolved on D_2 to obtain the clause $D_3 \sqcup B$. The two new clauses are then resolved on B, and the clause D_3 would be obtained.

Next, all clauses where B occurs are removed, and we get the following ontology.

$$\top \sqsubseteq A_1 \sqcup \exists r.D_1 \tag{7.12}$$

$$\top \sqsubseteq A_2 \sqcup \forall r.D_2 \tag{7.13}$$

$$\top \sqsubseteq \neg A_1 \sqcup \neg A_2 \sqcup \exists r.D_3 \tag{7.14}$$

$$\top \sqsubseteq D_3 \tag{7.15}$$

The definers D_1, D_2, and D_3 are eliminated in a similar way to the method described in Section 4.5. The two definers D_1 and D_2 are each eliminated by a purification inference, and D_3 is eliminated by a Non-Cyclic Definer Elimination inference. and we obtain the following ontology.

$$\top \sqsubseteq A_1 \sqcup \exists r.\top \tag{7.16}$$

$$\top \sqsubseteq A_1 \sqcup \neg A_2 \tag{7.17}$$

Our normal form is more flexible as it allows several negative definers to appear in a clause, thus avoids the extra resolution inferences performed by Lethe. Let us count the number of inferences performed when F3 is used, and compare them to those performed by Lethe. F3 resolves (7.7) and (7.8) on B to compute the clause $\top \sqsubseteq \neg D_1 \sqcup \neg D_2$. Then, it removes the clauses (7.7) and (7.8). Since D_1 is an existential definer and D_2 is universal, a Role Propagation inference is performed, and computes the following clause.

$$\top \sqsubseteq \neg A_1 \sqcup \neg A_2 \tag{7.18}$$

The clause $\top \sqsubseteq \neg D_1 \sqcup \neg D_2$ computed earlier is removed by the Reduction rule of the deductive customization. Definers are then eliminated using two purification inferences. In total Lethe performed seven inferences. One role propagation. Two resolution inferences on D_1 and D_2. One resolution inference on B. Two purification inferences to eliminate the D_1 and D_2. And, one inference to eliminate D_3. F3 performed only four inferences. One resolution inference on B. One role propagation inference. And, two purification inferences to eliminate D_1 and D_2. The inferences performed to translate the input ontology to the normal form were not counted because they are common to the two methods. Thus, F3 performed approximately half of the inferences performed by Lethe, which explains the performance improvements indicated by the graphs in

Figure 7.2.

We measured the size of the extracted set. In Evaluation 1, $\mathcal{D}^d$ was on average 0.01% of the size of the original ontology in the 10% setting, 0.66% in the 30% setting, and 18.3% in the 50% setting. In Evaluation 2, $\mathcal{D}^d$ was on average 0.23% of the size of the original ontology in the 10% setting, 0.88% in the 30% setting, and 7.13% in the 50% setting. The results are surprising, because when 10% and 30% of the concept names were forgotten, the number of clauses in $\mathcal{D}^d$ was less that 1% of the original ontology. When 50% of the concept names were forgotten the number of clauses in $\mathcal{D}^d$ increased significantly.

We also measured the number of definers in $\mathcal{D}^d$ as a ratio to the forgetting signature. In Evaluation 1, they were on average 0.03% in the 10% setting, 0.63% in the 30% setting, and 1.5% in the 50% setting. In Evaluation 2, they were on average 0.04% in the 10% setting, 0% in the 30% setting, and 1.4% in the 50% setting. These ratios indicate that our fine-grained method is feasible since appending (the deductively reduced ontology obtained by removing the $\mathcal{D}^d$ clauses from the intermediate ontology and performing Role Propagation inferences) with axioms from $\mathcal{D}^d$ introduced few definers relative to the size of $\mathbf{F}$.

Finally, we measured the size of the deductive view relative to the original ontology. In Evaluation 1 it was on average 114% of the size of the original ontology in the 10% setting, 103% in the 30% setting, and 117% in the 50% setting. In Evaluation 2, it was on average 113% of the size of the original ontology in the 10% setting, 95% in the 30% setting, and 103% in the 50% setting.

7.4 The Semantic Forgetting Method

In this section we used the configuration of F3 which uses Structure Preserving implementation of the Forgetting module, and turns the Customization module off. As noted earlier, this configuration is an implementation of the semantic forgetting method explained in Chapter 6.

7.4.1 The Experiments

We performed two evaluations, Evaluation 3 and Evaluation 4. In Evaluation 3, we used the real-life ontologies from the corpus described in Section 7.2. Experiments ran in the three settings, where 10%, 30%, and 50% of the concept names are forgotten.

Table 7.3: Statistics of the constructed ontologies and forgetting signatures.

	Maximum	Minimum	Average	Median
Ontology Size (axioms)	1422	78	302	201
Forgetting Signature Size (concepts)	324	1	27	10

The signature selection method of Evaluation 1 was used.

In Evaluation 4, the handmade ontologies where used, but not the real-life ontologies. The experiments compare the performance of F3 against the performance of the Fame(Q) tool [ZS19]. Fame(Q) is a Java implementation of the method of [ZS18]. The method of [ZS18] is a semantic forgetting method. It captures the semantic information that is representable in the description logic $\mathcal{ALCOQ}^{-,t,u}$. Comparison with Fame(Q) was not possible in Evaluation 3, because when tried, 52 experiments failed with memory issues. This happened more often in experiments with large forgetting signatures and ontologies. The findings would have been unreliable and representative of small ontologies and forgetting signatures only.

We used the handmade ontologies in three different experimental settings: Low, Moderate, and High with a total of 84 experiments. Each setting corresponded to a different choice of the forgetting signature. The Low setting meant that at least half of the axioms of the ontology contained a forgetting symbol, and the probability of two forgetting symbols appearing in the same axiom was low. The Moderate setting meant that all axioms of the ontology contained a forgetting symbol, and at least one axiom contained two or more forgetting symbols. The High setting meant that all axioms of the ontology contained a forgetting symbol, and at least half of the axioms contained two or more forgetting symbols.

The occurrence of several forgetting symbols in the same axioms makes the axioms generated from one forgetting round input to subsequent rounds. This made the experiments more computationally challenging for both methods, and compensated for the drawbacks of reducing the size of the input ontologies.

Statistics about the ontologies and the forgetting signatures are presented in Table 7.3.

7.4.2 Analysis of Evaluation 3

Statistics gathered from the experiments are shown in Fig. 7.3. Chart A of Fig. 7.3

Figure 7.3: Chart A: Average execution times (seconds) of F3. Chart B: Breakdown of the execution time into the time consumed eliminating forgetting symbols and the time consumed eliminating de ners. Chart C: Average number of de ners introduced by structural transformation Chart D: Average number of de ners remaining in the semantic forgetting view.

shows the average running time of the experiments across the three settings of forgetting 10%, 30%, and 50% of the signature. The chart shows a polynomial increase in the average running times (369, 699, and 1091 seconds in the 10%, 30%, and 50% settings respectively).

A breakdown of the average running times is shown in chart B of Fig. 7.3. The chart shows the average time consumed for eliminating the forgetting symbols using the Resolution, Tautology Elimination, Distribution, and Disjunction Elimination rules of the forgetting calculus in Fig. 6.2, and the average time consumed for eliminating the de ner using the de ner elimination rules in Fig. 6.3, relative to the average execution time of the whole forgetting process. We found that the time consumed for eliminating the de ners occupied the majority of the time of the whole forgetting process. However, we noticed a decreasing trend in the time consumed for eliminating the de ners. Execution of the forgetting calculus rules in Fig. 6.1 consumed on average 13%, 33%, and 40% of the time of the whole forgetting process in the 10%, 30%, and

50% settings respectively, whereas execution of the definer elimination rules in Fig. 6.3 consumed on average 87%, 67%, and 60% of the time. This suggests that definers not only simplify the forgetting method, but also the cost of eliminating them becomes less significant when forgetting large subsets of the vocabulary.

We measured in chart C of Fig. 7.3 the trend of introducing definers by structural transformation. We observed a linear increase in the average number of introduced definers. On average there were 1530, 2750, and 3708 introduced definers in the 10%, 30%, and 50% settings respectively. This increase was expected given the increase in the number of forgetting symbols. We also measured the number of introduced definers as a ratio to the forgetting signature. On average, the definers introduced by the structural transformation process were 107%, 56%, and 47% of the forgetting signature in the 10%, 30%, and 50% settings respectively. We may conclude that a factor of the downward trend in the time consumed by the elimination of definers is the relative decrease in the number forgetting symbols which caused new definers to be introduced.

We observed that in some cases the above factor also reduced the number of definers that remain in the semantic forgetting view. The following example shows the idea.

Example 7.4.1. Consider the following ontology $\mathcal{O}$.

$$A_1 \sqsubseteq C \sqcap \exists r.B \tag{7.19}$$

$$A_2 \sqsubseteq \exists r.\neg B \tag{7.20}$$

Let $\mathbf{F}_1 = \{B\}$, and $\mathbf{F}_2 = \{B, C\}$ be two forgetting signatures. The semantic forgetting view of $\mathcal{O}$ with respect to $\mathbf{F}_1$ is the ontology $\mathcal{V}_1$ consisting of the following axioms.

$$A_1 \sqsubseteq C \sqcap \exists r.D_1 \tag{7.21}$$

$$A_2 \sqsubseteq \exists r.D_2 \tag{7.22}$$

$$D_1 \sqcap D_2 \sqsubseteq \bot \tag{7.23}$$

The semantic forgetting view of $\mathcal{O}$ with respect to $\mathbf{F}_2$ is the ontology $\mathcal{V}_2$ comprising:

$$A_2 \sqsubseteq \exists r.\top \tag{7.24}$$

The concept name C does not appear under role restriction in $\mathcal{O}$, and does not result in introducing a definer. If only B is forgotten from $\mathcal{O}$, then the definers D_1 and D_2 appear

Figure 7.4: Chart A: Execution time(seconds) comparison with Fame(Q). Chart B: Average number of de ners introduced by structural transformation. Chart C: Average number of the introduced de ners as a ratio to the number of forgetting symbols.

in the semantic forgetting view. Forgetting additionally the concept name C leads to the elimination of the de ners D_1 and D_2 in the semantic forgetting view.

The observation illustrated by the example explains chart D in Fig. 7.3, which shows the average number of de ners in the final forgetting view. Recall that for an ontology evaluated in the 10%, 30%, and 50% settings, the forgetting signature used in the 10% setting is a subset of the forgetting signature used in the 30% setting, and the forgetting signature used in the 30% setting is a subset of the forgetting signature used in the 50% setting. When only 10% of the concept names were eliminated, there were on average 80 de ners in the forgetting view. Eliminating more de ners in the 30% and the 50% settings has led to eliminating most of the introduced de ners that were introduced by eliminating the symbols of the 10% setting as explained in the example. As such, only three de ners were used on average in the 30% setting, and 2 de ners in the 50% setting.

7.4.3 Analysis of Evaluation 4

Results gathered from the experiments are shown in Fig. 7.4. Chart A in Figure 7.4 compares the average execution times of Fame(Q) and F3. In general we found the execution time of Fame(Q) stable across the Low, Moderate, and High settings. The average execution times of Fame(Q) were 12, 14, 13 seconds respectively in the three settings. In contrast, the time consumed by F3 grew polynomially over the different settings. This growth was expected since more semantic information was captured by F3 as the forgetting problem becomes harder. Since Fame(Q) is not complete, we do not see the same increase in the time consumed by Fame. We noticed in the Low and Moderate settings that F3 consumes less time than Fame, which was not expected since it should still preserve more information than Fame. We found that Fame(Q) used a

reasoner to remove redundancies in the output, which resulted in consuming more time than F3.

We measured the number of introduced definers across the three settings in the same way as before. Chart B in Fig. 7.4 shows an increase in the number introduced definers as we moved to harder settings. On average, 149, 173, and 223 definers were introduced by structural transformation in the Low, Moderate, and High settings respectively. This increase is expected since the number of forgetting symbols increase. However, as shown in Chart C of Fig. 7.4, the average number of introduced definers as a ratio to the number of forgetting symbols decreased as we go to harder settings. This agreed with the ratios of 107%, 56%, and 47% that were found before when forgetting was done directly on the original BioPortal ontologies.

Chapter 8

Conclusions

Forgetting is an important engineering task that allows for extracting relevant modules from given ontologies. It has been shown crucial for important yet dif cult applications such as ontology creation, information hiding, abduction, and agent communication.

In this thesis we studied the deductive, semantic, and query forgetting notions in the context of the expressive description logic $\mathcal{ALC}$. We developed a forgetting framework that is capable of computing forgetting views for the three notions, as well as ne-grained forgetting views in between. The framework comprises two main processes. The rst is the forgetting process which eliminates forgetting symbols from the input ontology, and produces an intermediate ontology which coincides in content with the semantic forgetting view relative to the non-forgetting symbols. The second process is a customization process whereby the content of the intermediate ontology is ltered and customized to obtain the nal forgetting view. In this way, the framework segregates the elimination of the forgetting symbols from the customization of the content of the nal forgetting view, which makes forgetting suitable to solve more use cases than were considered before.

We designed a deductive customization whereby the deductive forgetting view, or a representation of it, is computed. A second output of the customization is a set of axioms that indicate the differences between the content of the input ontology relative to the non-forgetting symbols, and the content of the deductive forgetting view. This allows us to compare the semantic forgetting view of the ontology with the deductive forgetting view, and understand the information entailed by the former but not the latter. We designed an algorithm that uses this content difference to compute ne-grained forgetting views in-between the deductive forgetting view and the semantic forgetting view. With this algorithm, users can enhance the content of the deductive

forgetting view according to their requirements.

Extending on this customization idea, we presented another customization to our forgetting framework that computes the query forgetting view of the input ontology. The customization computes two representations of the query forgetting view. The first is an $\mathcal{ALC}$ representation which uses auxiliary symbols, or definers. The second is an $\mathcal{ALCI}$ representation. Under some acyclicity conditions, the $\mathcal{ALCI}$ representation does not use definers. In this way we give the user the flexibility of using either of the two representations according to his or her requirements. The first representation of the query forgetting view was obtained by enhancing the content of the deductive forgetting view in the way mentioned above. Therefore, it serves as a query forgetting view and a deductive forgetting view. The $\mathcal{ALC}$ representation also comes with a set of axioms that indicate the difference between its content and the content of the semantic forgetting view. This allows comparing the contents of the semantic, the deductive, and the $\mathcal{ALC}$ representation of the query forgetting view.

Next we investigated the problem of preserving the syntactic structure of the input ontology in the forgetting view. A common concern of most resolution based methods, is the use of structure damaging normal forms that do not preserve the syntactic structure of the input ontology. We developed a semantic forgetting method that uses only structure preserving operations in its normal form transformation. The method performs forgetting by first translating the relevant axioms to formulas in first-order logic, then eliminating the forgetting symbols from these formulas, and finally restoring the forgetting view to description logic syntax. We showed that with this method the forgetting view is more resembling the syntactic structure of the input ontology than other forgetting methods in the literature.

The fine-grained forgetting framework extended with the deductive customization, and the semantic forgetting method were both evaluated in a novel evaluation framework. Two different methods to select the forgetting symbols were used in the evaluations, and each was used in sets of experiments where 10%, 30%, and 50% of the concept names of the input ontologies were forgotten. In total this amounted to six sets of experiments. The time performance of our fine-grained forgetting framework was compared to the performance of the deductive forgetting system Lethe. In all six sets, the deductive customization of our forgetting framework performed significantly better than Lethe. In most experiments our deductive customization successfully terminated in less than half of the time consumed by Lethe, despite computing more information. In some experiments this even went down to almost 25% of the time consumed by

Lethe.

The performance of our semantic forgetting method was evaluated on its own, and in comparison with the semantic forgetting system Fame. The evaluations also suggested that the de ners occurring in the semantic forgetting view, if any, are often signi cantly fewer than the number of forgetting symbols. At most, the number of de ners was 8% of the number of forgetting symbols. In two evaluations out of three, our semantic forgetting method terminated successfully in less than 42 % of the time consumed by Fame.

8.1 Future Work

In future we intend to extend our work in the following directions.

Productization. We implemented a prototype, called F3, that showed promising results in comparison to state-of-the-art forgetting systems. However, F3 remains a prototype that is implemented as a proof of concept for our ideas. It can bene t from modern advances in the eld of software engineering such as parallelization and cloud computing. In addition, implementation of techniques such as subsumption deletion and other optimizations from the eld of automated theorem proving can enhance the performance of the forgetting methods, and result in more compact forgetting views.

Extending to other description logics. The normal form of the forgetting framework uses a language extended with de ners. As noted in Chapter 4, this language can be seen as a second-order language. Ontologies expressed in more expressive description logics can in principle be translated to such an extended language. As such, I believe the forgetting framework may be capable of working with more expressive description logics. However, this requires further investigation. The elimination of the de ners and restoring the results back to the source logic can be rather challenging.

Extending the framework with role forgetting and concept forgetting. Our ne-grained forgetting framework is capable of forgetting concept names. One way to extend the framework is allowing role forgetting. Role forgetting is insuf ciently studied in the contexts of query forgetting and expressive description logics. In other words, what information should be preserved when forgetting roles from ontologies in a query forgetting context? Another way to extend our framework is forgetting complex concepts. For instance, forgetting concepts such as $\exists r.C \sqcup D$. To the best of our knowledge, this sort of forgetting has not been investigated before.

Preserving the syntactic structure of the input ontology. We developed a semantic forgetting method that preserves the structure of the input ontology. Using the ideas of this forgetting method in the ne-grained forgetting framework can lead to more readable forgetting views, and attract users who are reluctant to use forgetting due to the syntactic structure of the forgetting views.

Bibliography

[Ack35] Wilhelm Ackermann. Untersuchungen über das Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 110:390–413, 1935.

[AS19] Ruba Alassaf and Renate A Schmidt. DLS-Forgetter: An Implementation of the DLS Forgetting Calculus for First-Order Logic. In Diego Calvanese and Luca Iocchi, editors, *GCAI 2019. Proceedings of the 5th Global Conference on Artificial Intelligence*, volume 65 of *EPiC Series in Computing*, pages 127–138. EasyChair, 2019.

[ASDG21] Ghadah Alghamdi, Renate A. Schmidt, Warren Del-Pinto, and Yongsheng Gao. Upwardly abstracted definition-based subontologies. In A. L. Gentile and R. Gonçalves, editors, *K-CAP'21: Knowledge Capture Conference*, pages 209–216. ACM, 2021.

[ASS22] Ruba Alassaf, Renate A. Schmidt, and Uli Sattler. Saturation-based uniform interpolation for multi-agent modal logics. In D. Fernández-Duque, A. Palmigiano, and S. Pinchinat, editors, *Advances in Modal Logic, Volume 14*, pages 37–58, London, 2022. College Publications.

[BCM+07] Franz Baader, Diego Calvanese, Deborah Mcguinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2nd edition, 2007.

[Bie16] Meghyn Bienvenu. Ontology-mediated query answering: Harnessing knowledge to get more from data. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 4058–4061. AAAI Press, 2016.

[Bio23] BioPortal. Bioportal visits https://bioportal.bioontology.org/visits, September 2023.

[BKL+17] Elena Botoeva, Boris Konev, Carsten Lutz, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. Inseparability and conservative extensions of description logic ontologies: A survey. In Reasoning Web 2016, pages 27–89. Springer, 2017.

[BKR+15] Elena Botoeva, Roman Kontchakov, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. When are description logic knowledge bases indistinguishable? In IJCAI International Joint Conference on Artificial Intelligence, volume 2015-Jan, pages 4240–4246, 2015.

[BKR+16] Elena Botoeva, Roman Kontchakov, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. Games for query inseparability of description logic knowledge bases. Artificial Intelligence, 234:78–119, 2016.

[BLR+19] Elena Botoeva, Carsten Lutz, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. Query inseparability for ALC ontologies. Artificial Intelligence, 272:1–51, 2019.

[BN07] Franz Baader and Werner Nutt. The Description Logic Handbook: Theory, Implementation and Applications, chapter Basic Description Logic, pages 47–104. Cambridge University Press, 2nd edition, 2007.

[BO15] Meghyn Bienvenu and Magdalena Ortiz. Ontology-mediated query answering with data-tractable description logics. In Reasoning Web. Web Logic Rules: 11th International Summer School 2015, Berlin, Germany, Jul 31- Aug 4, 2015, Tutorial Lectures, pages 218–307. Springer, 2015.

[Boo54] George Boole. An Investigation of the Laws of Thought: On Which Are Founded the Mathematical Theories of Logic and Probabilities. Cambridge University Press, 1854.

[Bor96] Alex Borgida. On the relative expressiveness of description logics and predicate logics. Artificial Intelligence, 82(1-2):353–367, Apr 1996.

[BOvX13] Meghyn Bienvenu, Magdalena Ortiz, Mantas Simkus, and Guohui

Xiao. Tractable queries for lightweight description logics. In Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI '13, page 768–774. AAAI Press, 2013.

[Bra77] Ronald J Brachman. What's in a concept: structural foundations for semantic networks. International Journal of Man-Machine Studies, 9(2):127–152, 1977.

[BS85] Ronald J Brachman and James G Schmolze. An overview of the KL-ONE Knowledge Representation System. Cognitive Science, 9(2):171–216, 1985.

[BUL01] Matthias Baaz, Egly Uwe, and Alexander Leitsch. Normal form transformations. In Handbook of Automated Reasoning, pages 275 – 332. North-Holland, 2001.

[BvBW06] Patrick Blackburn, Johan van Benthem, and Frank Wolter. Handbook of modal logic, volume 3 of Studies in logic and practical reasoning. Elsevier, 1st edition, 2006.

[CAS+19] Jieying Chen, Ghadah Alghamdi, Renate A Schmidt, Dirk Walther, and Yongsheng Gao. Ontology Extraction for Large Ontologies via Modularity and Forgetting. In Proceedings of the 10th International Conference on Knowledge Capture, K-CAP '19, pages 45–52, New York, NY, USA, 2019. Association for Computing Machinery.

[CDL+05] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. DL-Lite: Tractable Description Logics for Ontologies. In Proceedings of the National Conference on Artificial Intelligence, volume 2, pages 602–607, 2005.

[CEO14] Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Answering regular path queries in expressive Description Logics via alternating tree-automata. Information and Computation, 237:12–55, 2014.

[CG10] Bernardo Cuenca Grau. Privacy in ontology-based information systems: A pending matter. Semantic Web, 1(1, 2):137–141, 2010.

[CGL+18] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Encyclopedia of Database Systems,

chapter Ontology-Based Data Access and Integration, pages 2590–2596. Springer, New York, 2018.

[CGM10] Bernardo Cuenca Grau and Boris Motik. Pushing the limits of reasoning over ontologies with hidden content. In *Proc. KR 2010*, pages 214–224. AAAI Press, 2010.

[CHPS08] Jeremy Carroll, Ivan Herman, and Peter F. Patel-Schneider. *OWL 2 Web Ontology Language: RDF-Based Semantics*. ERCIM Research Center for Information Technology, 2nd edition, 2008. Last modified 2012.

[Con12] World Wide Web Consortium. Web ontology language (owl). https://www.w3.org/OWL/, December 2012.

[Con23] World Wide Web Consortium. W3C. https://www.w3.org/, Mar 2023.

[Cra57] William Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *Symbolic Logic*, 22(3):269–285, 1957.

[DHLM08] Hans van Ditmarsch, Andreas Herzig, Jérôme Lang, and Pierre Marquis. Introspective forgetting. In *Proceedings of the 21st Australasian Joint Conference on Artificial Intelligence*, pages 18–29. Springer, 2008.

[DL59] M A E Dummett and E J Lemmon. Modal Logics Between S 4 and S 5. *Mathematical Logic Quarterly*, 5(14-24):250–264, 1959.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A Machine Program for Theorem-Proving. *Communications of the ACM*, 5(7):394–397, Jul 1962.

[D S97] Patrick Doherty, Witold Łukaszewicz, and Andrzej Szałas. Computing Circumscription Revisited: A Reduction Algorithm. *Journal of Automated Reasoning*, 18(3):297–336, 1997.

[D S01] Patrick Doherty, Witold Łukaszewicz, and Andrzej Szałas. Computing strongest necessary and weakest sufficient conditions of first-order formulas. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence, IJCAI'01*, pages 145–151. Morgan Kaufmann, 2001.

[dlT92] Thierry de la Tour. An Optimality Result for Clause Form Translation. Journal of Symbolic Computation, 14(4):283–301, Oct 1992.

[DP60] Martin Davis and Hilary Putnam. A Computing Procedure for Quantification Theory. J. ACM, 7(3):201–215, Jul 1960.

[DP22] Warren Del-Pinto. Computing Spaces of Independent Explanations via Abductive Reasoning. PhD thesis, University of Manchester, 2022.

[DPS19a] Warren Del-Pinto and Renate A. Schmidt. ABox abduction via forgetting in ALC. In Proc. AAAI 2019, pages 2768–2775. AAAI Press, 2019.

[DPS19b] Warren Del-Pinto and Renate A. Schmidt. Extending forgetting-based abduction using nominals. In A. Herzig and A. Popescu, editors, Proceedings of the 12th International Symposium on Frontiers of Combining Systems (FroCoS 2019), volume 11715 of Lecture Notes in Artificial Intelligence, pages 185–202. Springer, 2019.

[DPSG+23] Warren Del-Pinto, Renate A. Schmidt, Yongsheng Gao, Ghadah Alghamdi, A. Lopez Osornio, and S. Roy. International patient summary terminology. In MEDINFO 2023: Proceedings of the 19th World Congress on Medical and Health Informatics, Studies in Health Technology and Informatics. IOS Press, 2023.

[DSG22] Warren Del-Pinto, Renate A. Schmidt, and Yongsheng Gao. Extracting subontologies from SNOMED CT. In P. Groth, A. Rula, J. Schneider, I. Tiddi, E. Simperl, P. Alexopoulos, R. Hoekstra, M. Alam, A. Dimou, and M. Tamper, editors, The Semantic Web: ESWC 2022 Satellite Events, volume 13384 of Lecture Notes in Computer Science, pages 291–294. Springer, 2022.

[EGOS08] Thomas Eiter, Georg Gottlob, Magdalena Ortiz, and Mantas Simkus. Query answering in the description logic horn-SHIQ. In Steffen Hölldobler, Carsten Lutz, and Heinrich Wansing, editors, Logics in Artificial Intelligence, pages 166–179. Springer, 2008.

[EHEVGJ21] Wafaa El Husseini, Cheikh Brahim El Vaigh, François Goasdoué, and Hélène Jaudoin. Ontology-Mediated Query Answering: Performance

Challenges and Advances. In International Semantic Web Conference (ISWC) CEUR, October 2021.

[EKI19] Thomas Eiter and Gabriele Kern-Isberner. A Brief Survey on Forgetting from a Knowledge Representation and Reasoning Perspective. KI - Künstliche Intelligenz, 33(1):9–33, 2019.

[Eng96] T. Engel. Quanti er elimination in second-order predicate logic. Diplomarbeit, Fachbereich Informatik, Univ. des Saarlandes, Saarbrücken, Germany, Oct 1996.

[EOS12] Thomas Eiter, Magdalena Ortiz, and Mantas Simkus. Conjunctive query answering in the description logic SH using knots. Journal of Computer and System Sciences, 78(1):47–85, 2012.

[GBIGJ17] Víctor Gutiérrez-Basulto, Yazmín Ibáñez-García, and Jean Christoph Jung. On Query Answering in Description Logics with Number Restrictions on Transitive Roles. In Proceedings of the 30th International Workshop on Description Logics, Montpellier, 2017. CEUR.

[GHH+23] Oliver Görlitz, Daniel Hausmann, Merlin Humml, Dirk Pattinson, Simon Prucker, and Lutz Schröder. Cool 2 – a generic reasoner for modal fixpoint logics (system description). In Automated Deduction – CADE 29, pages 234–247. Springer, 2023.

[GHKS07] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. A Logical Framework for Modularity of Ontologies. In Proceedings of the 20th International Joint Conference on Arti cial Intelligence, IJCAI'07, pages 298–303, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[GHKS08] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Modular Reuse of Ontologies: Theory and Practice. J. Artif. Int. Res., 31(1):273–318, Feb 2008.

[GHLS07] Birte Glimm, Ian Horrocks, Carsten Lutz, and Uli Sattler. Conjunctive query answering for the description logic SHIQ. In IJCAI International Joint Conference on Arti cial Intelligence, 31(1):399–404, Jan 2007.

[GHS04] Lilia Georgieva, Ullrich Hustadt, and Renate A. Schmidt. A Survey of Decidable First-Order Fragments and Description Logics. Journal of Relational Methods in Computer Science, 1:251–276, 2004.

[GKW14] William Gatens, Boris Konev, and Frank Wolter. Lower and Upper Approximations for Depleting Modules of Description Logic Ontologies. In Proceedings of the Twenty-First European Conference on Arti cial Intelligence, ECAI'14, pages 345–350, NLD, 2014. IOS Press.

[GLW06] Silvio Ghilardi, Carsten Lutz, and Frank Wolter. Did I damage my ontology? a case for conservative extensions in description logic. In Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning, KR'06, pages 187–197. AAAI Press, 2006.

[GO92] Dov Gabbay and Hans Jürgen Ohlbach. Quanti er elimination in second-order predicate logic. In Proc. KR 1992, pages 425–435. Morgan Kaufmann, 1992.

[Göd30] Kurt Gödel. Die Vollständigkeit der Axiome des logischen Funktionenkalküls. Monatshefte für Mathematik und Physik, 37(1):349–360, 1930.

[GSS08] Dov Gabbay, Renate A. Schmidt, and Andrzej Szałas. Second-Order Quanti er Elimination. College Publications, 2008.

[Hen63] Leon Henkin. An Extension of the Craig-Lyndon Interpolation Theorem. The Journal of Symbolic Logic, 28(3):201–216, 1963.

[HM08] Andreas Herzig and Jérôme Mengin. Uniform interpolation by resolution in modal logic. In Proceedings of the 11th European Conference on Logics in Arti cial Intelligence, volume 5293 LNAI, pages 219–231, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[Hoo91] John N Hooker. Logical Inference and Polyhedral Projection. In Proceedings of the 5th Workshop on Computer Science Logic, CSL '91, pages 184–200, Berlin, Heidelberg, 1991. Springer-Verlag.

[HPS11] Ian Horrocks and Peter F Patel-Schneider. Handbook of Semantic Web Technologies, chapter KR and Reasoning on the Semantic Web: OWL, pages 365–398. Springer, Berlin, 2011.

[HPSvH03] Ian Horrocks, Peter F Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: the making of a Web Ontology Language. Journal of Web Semantics, 1(1):7–26, 2003.

[Int23] SNOMED International. Snomed members. https://www.snomed.org/members, Sep 2023.

[KHM99] Juerg Kohlas, Rolf Haenni, and Serafín Moral. Propositional information systems. Journal of Logic and Computation, 9(5):651–681, 1999.

[KK16] Zurab Khasidashvili and Konstantin Korovin. Predicate elimination for preprocessing in first-order theorem proving. In Nadia Creignou and Daniel Le Berre, editors, Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings, volume 9710 of Lecture Notes in Computer Science, pages 361–372. Springer, 2016.

[KKK+19] Tatsuya Kushida, Kouji Kozaki, Takahiro Kawamura, Yuka Tateisi, Yasunori Yamamoto, and Toshihisa Takagi. Interconnection of biological knowledge using Nikkaji RDF and interlinking ontology for biological concepts. New Generation Computing, 37:1–25, 09 2019.

[KKL+11] Boris Konev, Roman Kontchakov, M. Ludwig, Thomas. Schneider, Frank Wolter, and Michel Zakharyaschev. Conjunctive query inseparability of OWL 2 QL TBoxes. In Proceedings of the National Conference on Artificial Intelligence, volume 1, pages 221–226, San Francisco, 2011. AAAI Press.

[KLWW09] Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization, chapter Formal Properties of Modularisation, pages 25–66. Springer, Berlin, 2009.

[KLWW13] Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. Model-theoretic inseparability and modularity of description logic ontologies. Artificial Intelligence, 203:66–103, 2013.

[Koo15] Patrick Koopmann. Practical Uniform Interpolation for Expressive Description Logics. PhD thesis, University of Manchester, 2015.

[KRMZ13] Roman Kontchakov, Mariano Rodríguez-Muro, and Michael Zakharyaschev. Ontology-based data access with databases: A short course. In *Lecture Notes in Computer Science*, volume 8067 LNAI, pages 194–229, Berlin, 2013. Springer.

[KS13a] P. Koopmann and R. A. Schmidt. Implementation and evaluation of forgetting in ALC-ontologies. In C. Del Vescovo, T. Hahmann, D. Pearce, and D. Walther, editors, *Proceedings of the 7th International Workshop on Modular Ontologies (WoMo 2013)*, volume 1081 of *CEUR Workshop Proceedings*, pages 1–12. CEUR, 2013.

[KS13b] Patrick Koopmann and Renate A. Schmidt. Forgetting concept and role symbols in $\mathcal{ALCH}$-ontologies. In *Proc. LPAR 2013*, volume 8312 of *Lecture Notes in Computer Science*, pages 552–567. Springer, 2013.

[KS13c] Patrick Koopmann and Renate A. Schmidt. Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints. In *Proc. FroCoS 2013*, volume 8152 of LNAI, pages 87–102. Springer, 2013.

[KS14] Patrick Koopmann and Renate A. Schmidt. Count and forget: Uniform interpolation of $\mathcal{SHQ}$-ontologies. In *Proc. IJCAR 2014*, volume 8562 of LNAI, pages 434–448. Springer, 2014.

[KS15a] Patrick Koopmann and Renate A. Schmidt. Saturation-based forgetting in the description logic $\mathcal{SIF}$. In Diego Calvanese and Boris Konev, editors, *Proceedings of the 28th International Workshop on Description Logics (DL-2015)*, volume 1350 of *CEUR Workshop Proceedings*, pages 439–452, Germany, 2015. CEUR.

[KS15b] Patrick Koopmann and Renate A. Schmidt. Uniform interpolation and forgetting for $\mathcal{ALC}$ ontologies with ABoxes. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 175–181. AAAI Press, 2015.

[KWW08] Boris Konev, Dirk Walther, and Frank Wolter. The Logical Difference Problem for Description Logic Terminologies. In *Proceedings of the 4th*

International Joint Conference on Automated Reasoning, IJCAR 2008, pages 259–274. Springer, 2008.

[KWW09] Boris Konev, Dirk Walther, and Frank Wolter. Forgetting and uniform interpolation in large-scale description logic terminologies. In Proc. IJCAI 2009, page 830–835. Morgan Kaufmann, 2009.

[KWZ08] Roman Kontchakov, Frank Wolter, and Michael Zakharyaschev. Can You Tell the Difference Between DL-Lite Ontologies?. In Principles of Knowledge Representation and Reasoning: Proceedings of the 11th International Conference, KR 2008, pages 285–295, 2008.

[KWZ10] Roman Kontchakov, Frank Wolter, and Michael Zakharyaschev. Logic-based ontology comparison and module extraction, with an application to DL-Lite. Arti cial Intelligence, 174(15):1093–1141, 2010.

[Lin00] Fangzhen Lin. On Strongest Necessary and Weakest Suf cient Conditions. In Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning, KR'00, pages 167–175, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[LK14] Michel Ludwig and Boris Konev. Practical uniform interpolation and forgetting for alc tboxes with applications to logical difference. In Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning, KR'14, pages 318–327. AAAI Press, 2014.

[LLA+21] Zhao Liu, Chang Lu, Ghadah Alghamdi, Renate A Schmidt, and Yizheng Zhao. Tracking Semantic Evolutionary Changes in Large-Scale Ontological Knowledge Bases. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, CIKM '21, pages 1130–1139, New York, NY, USA, 2021. Association for Computing Machinery.

[LLM03] Jérôme Lang, Paolo Liberatore, and Pierre Marquis. Propositional Independence - Formula-Variable Independence and Forgetting. Journal of Arti cial Intelligence Research, 18:391–443, jun 2003.

[LPW10] Carsten Lutz, Robert Piro, and Frank Wolter. $\mathcal{EL}$-Concepts go Second-Order: Greatest Fixpoints and Simulation Quantifiers. In Volker Haarslev, David Toman, and Grant E Weddell, editors, *Proceedings of the 23rd International Workshop on Description Logics*, volume 573 of CEUR Workshop Proceedings, pages 43–55. CEUR, 2010.

[LR94a] Fangzhen Lin and Ray Reiter. Forget it! In Russell Greiner and Devika Subramanian, editors, *Working Notes of AAAI Fall Symposium on Relevance*, pages 154–159. AAAI, 1994.

[LR94b] Fangzhen Lin and Ray Reiter. How to progress a database (and why) I. logical foundations. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, KR 1994, pages 425 – 436. Morgan Kaufmann, 1994.

[LW10] Carsten Lutz and Frank Wolter. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *Symbolic Computation*, 45:194–228, 02 2010.

[LW11] Carsten Lutz and Frank Wolter. Foundations for uniform interpolation and forgetting in expressive description logics. In *Proc. IJCAI 2011*, page 989–995, 2011.

[MBP13] Nicolas Matentzoglu, Samantha Bail, and Bijan Parsia. A snapshot of the OWL web. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz, editors, *Proceedings of the 12th International Semantic Web Conference, ISWC 2013*, pages 331–346, Berlin, 2013. Springer.

[MNS+11] Mark Musen, Natasha Noy, Nigam Shah, Patricia Whetzel, Christopher Chute, Margaret-Anne Story, and Barry Smith. The national center for biomedical ontology. *Journal of the American Medical Informatics Association : JAMIA*, 19:190–5, 11 2011.

[MS04] Gerome Miklau and Dan Suciu. A Formal Analysis of Information Disclosure in Data Exchange. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages

575–586, New York, NY, USA, 2004. Association for Computing Machinery.

[NS98] Andreas Nonnengart and Andrzej Szałas. A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. In Ewa Orlowska, editor, *Logic at Work. Essays Dedicated to the Memory of Helena Rasiowa*, pages 89–108. Springer, 1998.

[NW01] Andreas Nonnengart and Christoph Weidenbach. Computing small clause normal forms. In *Handbook of Automated Reasoning*, pages 335 – 367. North-Holland, 2001.

[OCE08] Magdalena Ortiz, Diego Calvanese, and Thomas Eiter. Data Complexity of Query Answering in Expressive Description Logics via Tableaux. *Journal of Automated Reasoning*, 41(1):61–98, 2008.

[Ort13] Magdalena Ortiz. Ontology Based Query Answering: The Story So Far. In Loreto Bravo and Maurizio Lenzerini, editors, *Proceedings of the 7th Alberto Mendelzon International Workshop on Foundations of Data Management, Puebla/Cholula, Mexico, May 21-23, 2013*, volume 1087 of *CEUR Workshop Proceedings*, pages 1–14. CEUR, 2013.

[OŠ12] Magdalena Ortiz and Mantas Šimkus. Reasoning and Query Answering in Description Logics. In Thomas Eiter and Thomas Krennwallner, editors, *Reasoning Web. Semantic Technologies for Advanced Query Answering: 8th International Summer School 2012, Vienna, Austria, September 3-8, 2012. Proceedings*, pages 1–53. Springer, Berlin, 2012.

[OSE08] Magdalena Ortiz, Mantas Simkus, and Thomas Eiter. Worst-case Optimal Conjunctive Query Answering for an Expressive Description Logic without Inverses. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 504–510. AAAI, 2008.

[Pal19] Ignazio Palmisano. Owl api 5.1.11. `https://mvnrepository.com/artifact/net.sourceforge.owlapi/owlapi-distribution/5.1.11`, 2019.

[Pan21] Tadeusz Pankowski. Modeling and Querying Data in an Ontology-Based Data Access System. *Procedia Computer Science*, 192:497–506, 2021.

[PG86] David A Plaisted and Steven Greenbaum. A Structure-Preserving Clause Form Translation. *Journal of Symbolic Computation*, 2(3):293–304, Sep 1986.

[Pir13] Robert Piro. *Model Theoretic Characterisations of Description Logics*. PhD thesis, Department of Computer Science, University of Liverpool, 2013.

[RG10] Sebastian Rudolph and Birte Glimm. Why infinity is your friend. *J. Artif. Intell. Res.*, 39:429–481, 2010.

[Sat07] Ulrike Sattler. Reasoning in description logics: Basics, extensions, and relatives. In *Proceedings of the Third International Summer School Conference on Reasoning Web*, RW'07, page 154–182, Berlin, Heidelberg, 2007. Springer-Verlag.

[Sch91] Klaus 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.

[Sch94] Klaus Schild. Terminological Cycles and the Propositional μ-Calculus. In Jon Doyle, Erik Sandewall, and Pietro Torasso, editors, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94). Bonn, Germany, May 24-27, 1994*, pages 509–520. Morgan Kaufmann, 1994.

[Sch12] Renate A. Schmidt. The Ackermann approach for modal logic, correspondence theory and second-order reduction. *Journal of Applied Logic*, 10(1):52–74, 2012.

[SPSW01] M Q Stearns, C Price, K A Spackman, and A Y Wang. SNOMED clinical terms: overview of the development process and project status. *Proceedings. AMIA Symposium*, pages 662–666, 2001.

[SS89] Manfred Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In Ronald J Brachman, Hector J Levesque, and Raymond Reiter, editors, *Proc. of the 1st Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'89)*, pages 421–431. Morgan Kaufmann, 1989.

[SS21] Mostafa Sakr and Renate A. Schmidt. Semantic forgetting in expressive description logics. In *Frontiers of Combining Systems: 13th International Symposium, FroCoS 2021*, page 118–136, Berlin, 2021. Springer.

[SS22] M. Sakr and Renate A. Schmidt. Fine-grained forgetting for the description logic ALC. In *Proceedings of the 35th International Workshop on Description Logics*. CEUR, 2022.

[SSS91] Gert Smolka and Manfred Schmidt-Schauß. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[SSZ09] Ulrike Sattler, Thomas Schneider, and Michael Zakharyaschev. Which Kind of Module Should I Extract? In Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, and Ulrike Sattler, editors, *Proceedings of the 22nd International Workshop on Description Logics (DL 2009)*, 2009.

[Sti98] Colin Stirling. The joys of bisimulation. In Luboš Brim, Jozef Gruska, and Jiří Zlatuška, editors, *Mathematical Foundations of Computer Science 1998*, volume 1450 of *Lecture Notes in Computer Science*, pages 142–151. Springer, 1998.

[Swe02] Latanya Sweeney. K-Anonymity: A Model for Protecting Privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, Oct 2002.

[Tar55] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific journal of Mathematics*, 5(2):285–309, 1955.

[TCMV06] Balder Ten Cate, Willem Conradie, Maarten Marx, and Yde Venema. Definitorially complete description logics. In *Proceedings of the International Workshop on Temporal Representation and Reasoning*, pages 79–89, 2006.

[TSP22] David Toluhi, Renate Schmidt, and Bijan Parsia. Concept Description and Definition Extraction for the ANEMONE System. In Natasha Alechina, Matteo Baldoni, and Brian Logan, editors, *Engineering Multi-Agent Systems*, pages 352–372, Cham, 2022. Springer International Publishing.

[vHLP08] Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter. *Handbook of Knowledge Representation*. Elsevier, San Diego, CA, USA, 2008.

[Vis96] Albert Visser. *Uniform interpolation and layered bisimulation*, volume 6 of *LNL*, page 139–164. Springer, 1996.

[Wer12] Christoph Wernhard. Projection and scope-determined circumscription. *Symbolic Computation*, 47(9):1089–1108, Sep 2012.

[WNS+11] Patricia L. Whetzel, Natasha Noy, Nigam Haresh Shah, Paul R. Alexander, Csongor Nyulas, Tania Tudorache, and Mark A. Musen. Bioportal: enhanced functionality via new web services from the national center for biomedical ontology to access and use ontologies in software applications. *Nucleic Acids Research*, 39:W541 – W545, 2011.

[WWT+09] Zhe Wang, Kewen Wang, Rodney Topor, Jeff Z. Pan, and Grigoris Antoniou. Uniform Interpolation for ALC Revisited. In *AI 2009: Advances in Artificial Intelligence*, pages 528–537, Melbourne, Australia, 2009.

[WWT+14] Kewen Wang, Zhe Wang, Rodney Topor, Jeff Z. Pan, and Grigoris Antoniou. Eliminating Concepts and Roles from Ontologies in Expressive Description Logics. *Computational Intelligence*, 30(2):205–232, May 2014.

[WWTP08] Zhe Wang, Kewen Wang, Rodney Topor, and Jeff Z Pan. Forgetting Concepts in DL-Lite. In *Proceedings of the 5th European Semantic Web Conference on The Semantic Web: Research and Applications*, ESWC'08, pages 245–257, Berlin, Heidelberg, 2008. Springer-Verlag.

[XCK+18] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. Ontology-Based Data Access: A Survey. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5511–5519. International Joint Conferences on Artificial Intelligence Organization, 2018.

[ZFA+18] Yizheng Zhao, Hao Feng, Ruba Alassaf, Warren Del-Pinto, and Renate A. Schmidt. The FAME family: A family of reasoning tools for forgetting in expressive description logics. In Magdalena Ortiz and Thomas Schneider, editors, *Proceedings of the 31st International Workshop on Description Logics*, volume 2211 of *CEUR Workshop Proceedings*. CEUR, 2018.

[Zha18] Yizheng Zhao. *Automated Semantic Forgetting For Expressive Description Logics*. PhD thesis, University of Manchester, 2018.

[ZS15] Yizheng Zhao and Renate A. Schmidt. Concept forgetting in $\mathcal{ALCOI}$-ontologies using an Ackermann approach. In M. Arenas, O. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. T. Groth, M. Dumontier, J. Heflin, K. Thirunarayan, and S. Staab, editors, *The Semantic Web, 14th International Semantic Web Conference, ISWC 2015*, volume 9366 of *Lecture Notes in Computer Science*, pages 587–602. Springer, 2015.

[ZS16] Yizheng Zhao and Renate A. Schmidt. Forgetting concept and role symbols in $\mathcal{ALCOIH}\mu^{+}(\tilde{\mathrm{N}}; u)$-ontologies. In S. Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI 2016)*, pages 1345–1352. AAAI Press/IJCAI, 2016.

[ZS17] Yizheng Zhao and Renate A. Schmidt. Role forgetting for $\mathcal{ALCOQH}(\tilde{\mathrm{N}})$-ontologies using an Ackermann approach. In C. Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI'17)*, pages 1354–1361. AAAI Press/IJCAI, 2017.

[ZS18] Yizheng Zhao and Renate A. Schmidt. On concept forgetting in description logics with qualified number restrictions. In J. Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI'18)*, pages 1984–1990. AAAI Press/IJCAI, 2018.

[ZS19] Yizheng Zhao and Renate A. Schmidt. FAME(Q): An automated tool for forgetting in description logics with qualified number restrictions. In Pascal Fontaine, editor, *The 27th International Conference on Automated Deduction–CADE-27*, volume 11716 of *Lecture Notes in Artificial Intelligence*. Springer, 2019.

[ZZ09] Yan Zhang and Yi Zhou. Knowledge forgetting: Properties and applications. *Artificial Intelligence*, 173:1525–1537, 2009.

[ZZ10] Yan Zhang and Yi Zhou. Forgetting revisited. In *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning*, KR'10, page 602–604. AAAI Press, 2010.

[ZZ11] Yi Zhou and Yan Zhang. Bounded forgetting. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, page 280–285. AAAI Press, 2011.